RC 21103(94395)(19FEB98)
Computer Science/Mathematics

IBM Research Report

The Volume Algorithm: producing primal solutions with a subgradient method

Francisco Barahona, Ranga Anbil

IBM Research Division
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York

Research Division
Almaden · T.J. Watson · Tokyo · Zurich

The Volume Algorithm: producing primal solutions with a subgradient method

Francisco Barahona[1]
Ranga Anbil[1]

October 1997

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

Abstract

We present an extension to the subgradient algorithm to produce primal as well as dual solutions. It can be seen as a fast way to carry out an approximation of Dantzig-Wolfe decomposition. This gives a fast method for producing approximations for large scale linear programs. It is based on a new theorem in linear programming duality. We present successful experience with linear programs coming from set partitioning, set covering, max-cut and plant location.

Key words. Subgradient algorithm, Dantzig-Wolfe decomposition, large scale linear programming.

AMS subject classifications. 90C05, 90C06.

1. Introduction

Since the work of Held and Karp [19, 20] and Held, Wolfe and Crowder [21], in the early seventies, the subgradient algorithm has been used in many different contexts to produce lower bounds for large-scale linear programs, see e. g. [25] for more references about it. This procedure is attractive because of its low computational cost. The literature contains many experiences in which it has produced very good approximations to optimal dual solutions. One drawback is that it does not produce values for the primal variables, and then a different procedure has to be applied for their computation. This is usually done based on the complementary slackness conditions, in some cases it can be done in a combinatorial way, in others one has to solve a smaller linear program. Another drawback of the subgradient algorithm is it has not a well defined stopping criterion. In most practical cases, the stopping criterion is just based on a limit for the number of iterations or the number of steps without an improvement.

The question of producing primal solutions with a subgradient algorithm has been studied in [28], [22], [2] and [27]. They give proofs of convergence for different choices of the step length. We do not know whether these methods have been tested with large-scale problems.

We present an extension to the subgradient algorithm that will produce an approximation to a primal solution. Additionally, this extension, while maintaining the same low computational cost per iteration, it gives a much better stopping criterion. In general, it produces a primal vector as well as a dual vector that can be used by themselves or as the starting points for a more exact method. The convergence properties of subgradient algorithms are not well understood; versions that converge in theory are too slow computationally, and for the versions that are implemented there is not a known proof of convergence. Our algorithm falls in this category, it mimics convergent approaches, but the emphasis is in numerical computations. We present successful experiments with linear programs coming from combinatorial problems like set partitioning, set covering, max-cut, and plant location. We have not done any experiments with

linear programs with a more general structure, these might turn out to be more difficult for this type of method. When dealing with the dual variables, our algorithm has similarities with the *Conjugate Subgradient method* proposed in [29, 23]. What seems to be new is the way of dealing with the primal variables. There we try to estimate the volume below the faces that are active at an optimal dual solution. These volumes provide the primal solution.

This paper is organized as follows. In Section 2 we review Dantzig-Wolfe decomposition, lagrangean relaxation and the subgradient algorithm. In Section 3 we present a theorem on linear programming duality that justifies the algorithm. In Section 4 we present the extended subgradient algorithm. Section 5 contains some implementation issues. In Section 6 we present experiments with different types of problems.

2. Lagrangean relaxation and subgradients

In this section we present an overview of Dantzig-Wolfe decomposition, lagrangean relaxation, and the subgradient algorithm. Consider the linear program

$$\left.\begin{array}{l} \text{minimize } cx \\ Ax = b \\ Dx = e \\ x \geq 0. \end{array}\right\} \tag{2.1}$$

and assume that

$$\{x | Dx = e, x \geq 0\} = \{x | x = \sum \lambda_i g_i, \sum \lambda_i = 1, \lambda \geq 0\}.$$

Dantzig-Wolfe decomposition [14] relies on the fact that (2.1) is equivalent to

$$\left.\begin{array}{l} \text{minimize } \sum (cg_i)\lambda_i \\ \sum (Ag_i)\lambda_i = b \\ \sum \lambda_i = 1, \\ \lambda \geq 0. \end{array}\right\} \tag{2.2}$$

Then (2.2) is solved with a column generation approach. Each new column is obtained by solving

$$\left.\begin{array}{l} \text{minimize } (c - \bar{\pi} A)\, x \\ Dx = e \\ x \geq 0, \end{array}\right\} \tag{2.3}$$

where $\bar{\pi}$ are the dual values associated with the current basis in (2.2). There are cases where this procedure might take a large number of iterations. In [5] there is empirical evidence that this procedure can be accelerated by mixing it with the subgradient method. The method that we describe in this paper can be seen as a fast way to approximate Dantzig-Wolfe decomposition.

Now we describe the subgradient method. Problem (2.2) is equivalent to

$$\left.\begin{array}{l} \text{minimize } \sum cg_i\lambda_i \\ \sum (Ag_i - b)\lambda_i = 0 \\ \sum \lambda_i = 1, \\ \lambda \geq 0, \end{array}\right\} \tag{2.4}$$

its dual is

$$\left.\begin{array}{l} \text{maximize } z \\ z + \pi(Ag_i - b) \leq cg_i. \end{array}\right\} \tag{2.5}$$

The subgradient algorithm operates as follows:

Step 1. Given $\bar{\pi}$, find a tight inequality in (2.5), by solving

$$\left.\begin{array}{l} \text{minimize } z = (c - \bar{\pi}A)x + \bar{\pi}b \\ Dx = e \\ x \geq 0. \end{array}\right\} \tag{2.6}$$

Let $\bar{x}$ be a solution of this. Then $v = b - A\bar{x}$ is a subgradient at $\bar{\pi}$.

Step 2. Update $\bar{\pi}$ as $\bar{\pi} \leftarrow \bar{\pi} + sv$. Here s is the step size, and the usual formula for it is

$$s = f\frac{UB - \bar{z}}{\|v\|^2}, \tag{2.7}$$

where f is a number between 0 and 2, and UB is an upper bound for the optimal value.

If some of the constraints in $Ax = b$ are inequalities, instead of equations, and their dual variables are required to be nonnegative, then one should take

$$\max\{\bar{\pi}_i, 0\}$$

for those components, in Step 2.

Usually f is decreased after a certain number of iterations without an improvement. The usual stopping criterion is given by a bound on the total number of iterations, or by a bound on the number of iterations without an improvement.

If the subproblem (2.6) can be solved efficiently, then each iteration has a very low computational cost. There are many cases in the literature in which this procedure has been very effective. Although there are convergence theorems, see [26, 7, 24], the convergence properties are not well understood. In many cases, the subgradient algorithm produces a good approximation to an optimal solution of (2.5), however it does not produce a primal vector λ that solves (2.2) or (2.4). We are going to present an extension of this procedure to produce an approximation to an optimal solution of (2.2) or (2.4).

3. Volume and Duality

This section is devoted to a theorem in linear programming duality that leads to the computation of the primal variables (that are the dual variables in this section).

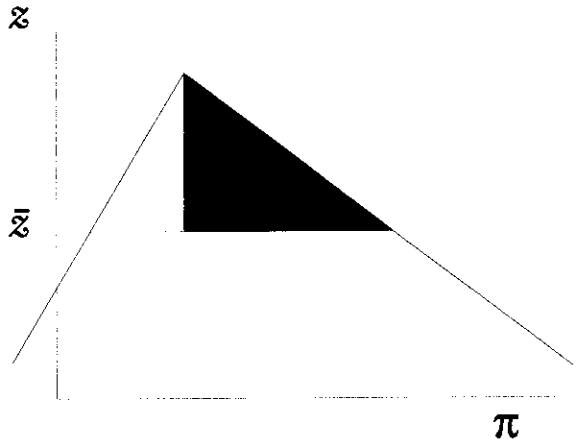

Figure 3.1

Theorem 3.2. Consider the linear program

$$\left.\begin{array}{l} \text{maximize } z \\ \text{subject to} \\ z + a_i\pi \leq b_i, \text{ for } i = 1, ..., m. \end{array}\right\} \tag{3.3}$$

where π is a vector with $n-1$ components. Let $(\hat{z}, \hat{\pi})$ be an optimal solution, and suppose that that constraints $1, ..., m'$, $m' \leq m$, are active at this point. Let $\bar{z} < \hat{z}$, and assume that

$$\left.\begin{array}{l} z + a_i\pi \leq b_i, \text{ for } i = 1, ..., m', \\ z \geq \bar{z}, \end{array}\right\} \tag{3.4}$$

defines a bounded polyhedron. For $1 \leq i \leq m'$, let γ_i be the volume between the face defined by $z + a_i\pi \leq b_i$ and the hyperplane defined by $z = \bar{z}$. The shaded region in Figure 3.1 illustrates

such a volume. Then an optimal dual solution is given by

$$\lambda_i = \frac{\gamma_i}{\sum_{j=1}^{m'} \gamma_j}.$$

Proof. Consider the polyhedron P defined by (3.4), this is a full dimensional polytope. denote by F_0 to be the face defined by $z \geq \bar{z}$, and by F_i the face defined by $z + a_i\pi \leq b_i$. Gauss' divergence theorem says that for a closed bounded region P whose boundary is a piecewise smooth orientable surface S, and a constant vector v,

$$\int_S v \cdot n \; dS = 0, \tag{3.5}$$

where n is the outer unit normal vector of S. By taking $v = e_j$ (the jth unit vector) for $1 \leq j \leq n$, we obtain

$$\int_S n \; dS = \mathbb{0}, \tag{3.6}$$

where $\mathbb{0}$ denotes a vector of all zeroes. This implies

$$\sum_{i=1}^{m'} \frac{\delta_i}{\|(1, a_i)\|}(1, a_i) - \delta_0(1, 0 \ldots 0) = \mathbb{0}$$

where δ_i is the area of F_i. Thus

$$(1, 0 \ldots 0) = \sum_{i=1}^{m'} \frac{\delta_i}{\delta_0 \|(1, a_i)\|}(1, a_i),$$

so at this point we have the gradient of the objective function written as a nonnegative linear combination of the gradients of the constraints that are active at the optimum. This gives us an optimal dual solution.

Now we shall see that

$$\gamma_i = C \, \frac{\delta_i}{\|(1, a_i)\|},$$

where C is a constant, and γ_i, as defined earlier, is the volume between F_i and the hyperplane F_0.

If $\delta_i = 0$ then $\gamma_i = 0$, so we have to consider the case when $\delta_i > 0$. For that we apply Gausses theorem again as follows. Denote by Q_i the convex hull of F_i and $(\bar{z}, \hat{\pi})$, see Figure 3.7.

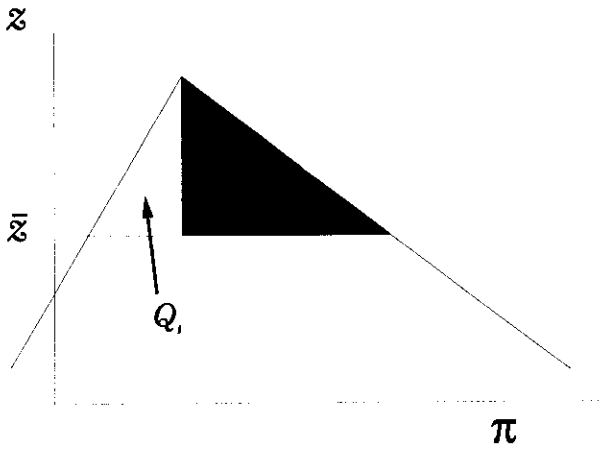

Figure 3.7

Let $\bar{F}_0 = F_0 \cap Q_i$. Notice that the faces of Q_i other than $\bar{F}_0$ and F_i are defined by inequalities like

$$c\pi \leq d,$$

where the variable z is absent. This means that the normals to these faces are orthogonal to the z axis. Then if we use formula (3.5) with $v = e_1$ on Q_i, the only faces with $v \cdot n \neq 0$ are F_i and $\bar{F}_0$. We have

$$A_i = \frac{\delta_i}{\|(1, a_i)\|},$$

where A_i is the area of $\bar{F}_0$. Let $h = \hat{z} - \bar{z}$, then the volume of Q_i is $\gamma_i = {}^1\!/_2 h\, A_i$.

Since $\delta_0 = \sum A_j$, we have

$$\frac{\delta_i}{\delta_0 \|(1, a_i)\|} = \frac{A_i}{\sum A_j} = \frac{\gamma_i}{\sum \gamma_j}.$$

□

Roughly speaking, this theorem suggests that given a vector $(\bar{z}, \bar{\pi})$ one should look at the active faces, and compute the volume of their projection over the hyperplane $z = \bar{z} - \epsilon$, on a neighborhood of $\bar{\pi}$, and for some small value of ϵ. Let λ_i be the ratio of the volume below F_i to the total. Then one should compute

$$(1, 0 \ldots 0) - \sum_{i=1}^{m'} \lambda_i (1, a_i) \tag{3.8}$$

If this is 0 we have a proof of optimality, otherwise we have a direction of improvement, see Figure 3.9. We present an algorithm in Section 4 that computes approximations to these volumes.

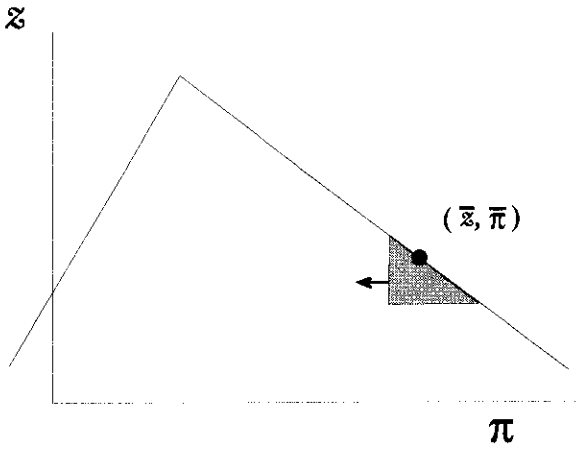

Figure 3.9

4. The volume algorithm

We describe below a modification of the subgradient algorithm to produce primal solutions. The basic idea comes from applying Theorem 3.2 to (2.5). This tell us that in a neighborhood of an optimal solution of (2.5), the probability that (2.6) will produce a face i is λ_i; this is an optimal solution of the primal problem (2.2) or (2.4). So we modify the subgradient algorithm in order to estimate these probabilities. If one looks only at the dual variables, the algorithm below has similarities with the Conjugate Subgradient method [29, 23] and the Bundle method [24].

Volume Algorithm

Step 0. We start with a vector $\bar{\pi}$ and solve (2.6) to obtain $\bar{x}$ and $\bar{z}$.

Set $x^0 = \bar{x}$, $z^0 = \bar{z}$, $t = 1$.

Step 1. Compute $v^t = b - A\bar{x}$ and $\pi^t = \bar{\pi} + sv^t$ for a step size s given by (2.7).

Solve (2.6) with π^t, let x^t and z^t be the solutions obtained. Then $\bar{x}$ is updated as

$$\bar{x} \leftarrow \alpha x^t + (1-\alpha)\bar{x}, \tag{4.1}$$

where α is a number between 0 and 1.

Step 2. If $z^t > \bar{z}$ update $\bar{\pi}$ and $\bar{z}$ as

$$\bar{\pi} \leftarrow \pi^t, \quad \bar{z} \leftarrow z^t.$$

Let $t \leftarrow t+1$ and go to Step 1.

If $x^0, \ldots, x^t$ is the sequence of vectors produced by (2.6), then

$$\bar{x} = \alpha x^t + (1-\alpha)\alpha x^{t-1} + \ldots + (1-\alpha)^t x^0.$$

So we should look at $\bar{x}$ as a convex combination of $\{x^0, \ldots, x^t\}$. Our goal is to use the coefficients $\alpha, (1-\alpha)\alpha, \ldots, (1-\alpha)^t$ as an approximation of an optimal solution λ of the master problem (2.2) in Dantzig-Wolfe decomposition.

Let us call Step 1 a minor iteration, and an update in Step 2 a major iteration. We might have a sequence of minor iterations before finding an improvement. During this stage we have a tentative set of dual values λ associated with the inequalities of (2.5), then when problem (2.6) produces an inequality i, we increase its dual value to $\alpha + (1-\alpha)\lambda_i$, and we multiply all the other dual values by $(1-\alpha)$. The idea behind this is that we are moving in a neighborhood of $(\bar{z}, \bar{\pi})$ and we are trying to estimate the volumes below the faces that are active in this neighborhood; see Theorem 3.2.

We use these dual values to define the direction v, while in the subgradient algorithm, the direction is given by only one active face. If no further improvement is found, the values λ should approximate an optimal solution of (2.2). Notice that we do not keep a vector λ explicitely, all this information is in the primal vector $\bar{x}$.

Another idea to approximate these volumes would be to just count the number of times that a face is produced by (2.6). We prefer the power series described above, thus if a face appeared only early in the procedure its weight would decrease exponentially.

We should stop when $\|v\|$ and $|c\bar{x} - \bar{z}|$ are both below a certain threshold.

The idea of taking a convex combination of the new and the old direction at each iteration of the subgradient algorithm, appears in [8, 7, 3] and maybe others with the intention of accelerating solution convergence. However, there is no mention of any means for obtaining primal solutions.

5. Implementation details

In this section we give further details on how we computed α in (4.1), and f in (2.7).

The value α could be set to a fixed value for a number of iterations and could be decreased afterwards. We used an idea from the Conjugate Subgradient method [29, 23] as follows. Let $\bar{v} = b - A\bar{x}$ and $v^t = b - Ax^t$, and let α_{max} be an upper bound. Then we would compute α_{opt} as the value that minimizes $\|\alpha v^t + (1-\alpha)\bar{v}\|$. If $\alpha_{opt} < 0$ we would set $\alpha = \alpha_{max}/10$. Otherwise we would set $\alpha = \min\{\alpha_{opt}, \alpha_{max}\}$. Typically we would start with $\alpha_{max} = 0.1$ and we would decrease its value near the end. This should increase the precision of the primal solution.

Then to set the value of f we define three types of iterations. Each time that we do not

find an improvement we call this iteration *red.* If $z^t > \bar{z}$ we compute

$$d = v^t \cdot (b - Ax^t).$$

If $d < 0$ it means that a longer step in the direction v^t would have given a smaller value for z^t, we call this iteration *yellow.* If $d \geq 0$ we call this iteration green.

At each green iteration we would multiply f by 1.1. After a sequence of 20 consecutive red iterations we would multiply f by 0.66.

6. Computational experiments

In this section we describe successful results with three types of large scale linear programs. Their only common characteristic is that all come from combinatorial problems. The stopping criterion for the volume algorithm was

$$\frac{|c\bar{x} - \bar{z}|}{\bar{z}} < 0.02, \quad \text{and} \quad \frac{\sum |v_i|}{m} < 0.01,$$

where m is the number of rows of the matrix A. We compared with the simplex method and the interior point method implemented in OSL [13].

6.1. Set partitioning problems

The linear programming relaxation of a set partitioning problem can be described as

$$\begin{aligned} &\text{minimize } cx \\ &Ax = \mathbb{1} \\ &x \geq 0, \end{aligned}$$

where A is a matrix with 0-1 coefficients, and $\mathbb{1}$ denotes a vector of ones. The instances come from applying the global approach described in [1] to a major airline's crew scheduling. The columns of A correspond to crew trips and the rows correspond to flights that have to be covered.

We relaxed all equations, and problem (2.6) became

$$\left.\begin{aligned} &\text{minimize } (c - \pi A)\, x + \pi \mathbb{1} \\ &0 \leq x \leq \mathbb{1}. \end{aligned}\right\} \tag{6.1}$$

For a problem with 2504 rows and 50722 columns, we plot in Figure 6.3 the value of $\bar{z}$ with a continuous line and the value of $c\bar{x}$ with bullets. We plot in Figure 6.2 the value of $(\sum |v_i|)/m$, where m is the number of equations.

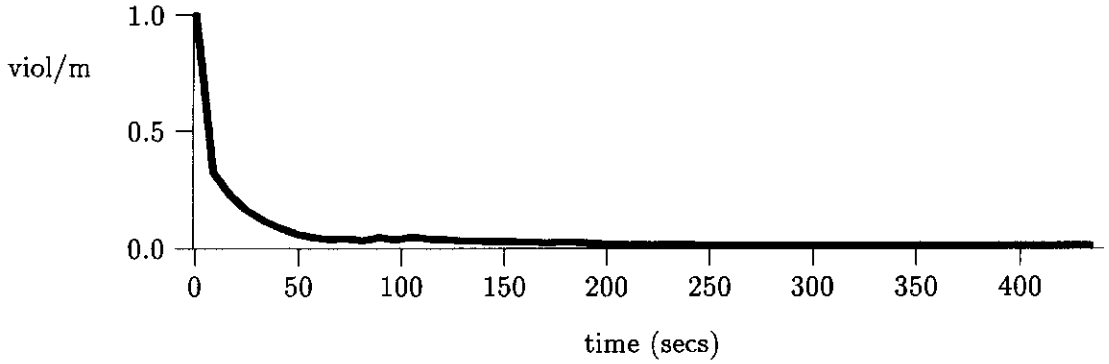

Figure 6.2

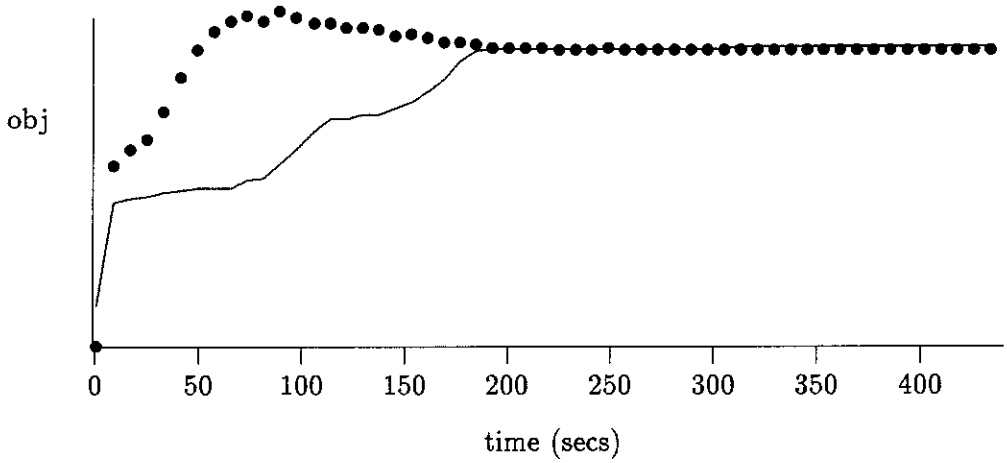

Figure 6.3

In Table 6.4 we present computing times on an IBM RS 6000/590 for 2 instances. The label V corresponds to the time taken by the volume algorithm. Given the dual values produced by the volume algorithm, we computed reduced costs, using this information and the primal values we selected a set of columns and we applied the dual simplex method to this smaller problem using the reduced costs instead of the original objective, then we priced out the remaining columns, the time taken by this part appears under the label V-D. We also applied the dual simplex method from scratch to the entire problem, this time appears under D. It is known that the dual simplex method performs better than primal simplex for set partitioning problems, and our experience confirms this. Finally we applied an interior point method, it was the primal-dual barrier algorithm with a predictor-corrector method. This time appears under B.

TABLE 6.4

name	rows	columns	V (secs.)	V-D (secs.)	D (secs.)	B (secs.)
sp6	2504	50722	440	135	3283	1299
sp7	2991	46450	895	428	5753	2048

Then we tried these instances as set covering problems, i. e. with inequalities

$$Ax \geq \mathbb{1}$$

instead of equations. The subgradient algorithm has been used to obtain dual solutions and lower bounds in [17, 4, 9, 10] and others. For problems with "small costs", a subgradient method has been used to produce primal solutions, see [10].

In Table 6.5 we display the solution times on an IBM RS 6000/590 for the set covering trials. For set covering problems, our experience is that the primal simplex method is more effective than dual simplex. We applied the volume algorithm and we used the primal vector to start the primal simplex method. The times of these two phases appear under V and V-P. Under the label P we present the time taken by the primal simplex method starting from scratch. The time taken by the interior point algorithm appears under B.

TABLE 6.5

name	rows	columns	V (secs.)	V-P (secs.)	P (secs.)	B (secs.)
sp6	2504	50722	471	134	4970	1831
sp7	2991	46450	645	171	5517	2311

6.2. Max cut problems

Given a graph $G = (V, E)$, the max-cut problem consists of finding a partition of the set of nodes into two sets that maximizes the total weight of the edges between the two sets. For a complete graph, a linear programming relaxation is given by

$$\begin{aligned}
&\text{maximize } c\,x \\
&\text{subject to} \\
&x_{ij} + x_{jk} + x_{ik} \leq 2, \\
&x_{ij} - x_{jk} - x_{ik} \leq 0, \\
&-x_{ij} + x_{jk} - x_{ik} \leq 0, \\
&-x_{ij} - x_{jk} + x_{ik} \leq 0, \\
&\text{for all } i, j, k.
\end{aligned}$$

Here x_{ij} should take the value 1 if the edge ij appears in the cut, and 0 otherwise. These are called the *triangle inequalities* and they define facets of the cut polytope, see [6]. It was pointed out in [15] that this linear program can be very difficult to solve with the simplex method, even for small graphs. So we tried it with the objective of all ones. Our first observation was that dual simplex performed better than primal simplex. For the smaller case below, primal simplex did not move from the origin after two hours of pivoting. As for set partitioning we relaxed all triangle inequalities, so problem (2.6) was of the type

$$\text{minimize } \bar{c}x + \pi b$$
$$0 \leq x \leq \mathbb{1}.$$

We tried one instance with 60 nodes and one with 80 on an IBM RS 6000/590. In Table 6.6 we present the time taken by the volume algorithm under V. The time taken by primal simplex starting from the primal vector given by our algorithm appears under V-P. The time taken by the dual simplex method starting from scratch appears under D. Because of its high storage requirements we were not able to run the interior point method on these problems.

TABLE 6.6

nodes	variables	constraints	V (secs.)	V-P (secs.)	D (secs.)
60	1770	136880	215	933	3242
80	3160	328640	339	6102	55921

6.3. Facility location

The uncapacitated facility location problem consists of selecting a subset from a set of possible locations of facilities, that should serve a set of customers. A linear programming relaxation of it is

$$\text{minimize } \sum a_i x_i + \sum c_{ij} f_{ij}$$
$$\text{subject to}$$
$$\sum_i f_{ij} = 1, 1 \leq j \leq n, \tag{6.7}$$
$$f_{ij} \leq x_i, 1 \leq i \leq m,\ 1 \leq j \leq n, \tag{6.8}$$
$$0 \leq x_i \leq 1,\ 0 \leq f_{ij} \leq 1.$$

If we relax inequalities (6.7), and we drop the index i, we obtain subproblems as below

$$\text{minimize } ax + \sum \bar{c}_j f_j$$

subject to

$$f_j \leq x, \text{ for all } j,$$

$$0 \leq x \leq 1,\ 0 \leq f_j \leq 1.$$

This can be solved as follows. Set to 0 any variable f_j with $\bar{c}_j \geq 0$. Then compute

$$\mu = a + \sum_{\bar{c}_j < 0} \bar{c}_j.$$

If $\mu < 0$ set $x = 1$, otherwise set $x = 0$. Set $f_j = x$ if $\bar{c}_j < 0$.

An algorithm based Dantzig-Wolfe decomposition was given in [18], an ascent method has been used in [16], the subgradient algorithm to obtain a lower bound has been used in [11], a primal subgradient algorithm to produce primal solutions and an upper bound has been given in [12]. We decided to try the volume algorithm, using this relaxation. We took an instance from [5] with 250 possible locations and 163 customers and conducted solution runs on an IBM RS 6000/590. In Figure 6.9 we plot the value of $(\sum |v_i|)/n$. We plot in Figure 6.10 the value of the lower bound with a continuos line and the value of of the primal vector with bullets.

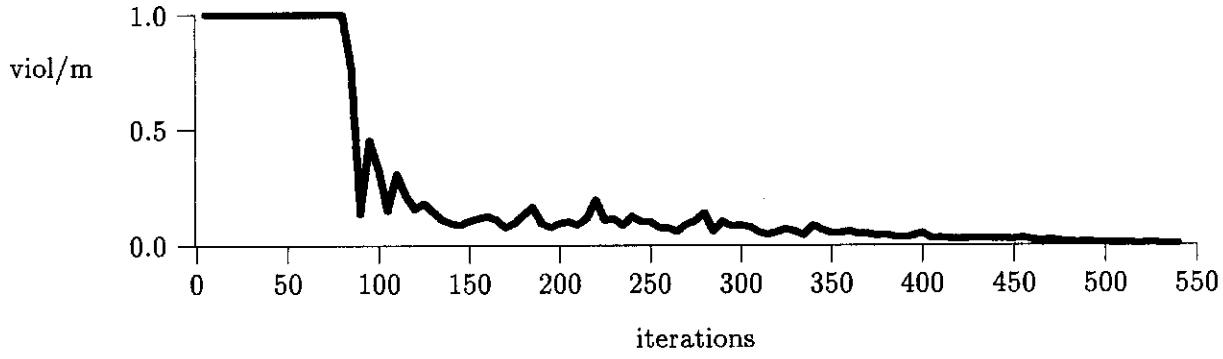

Figure 6.9

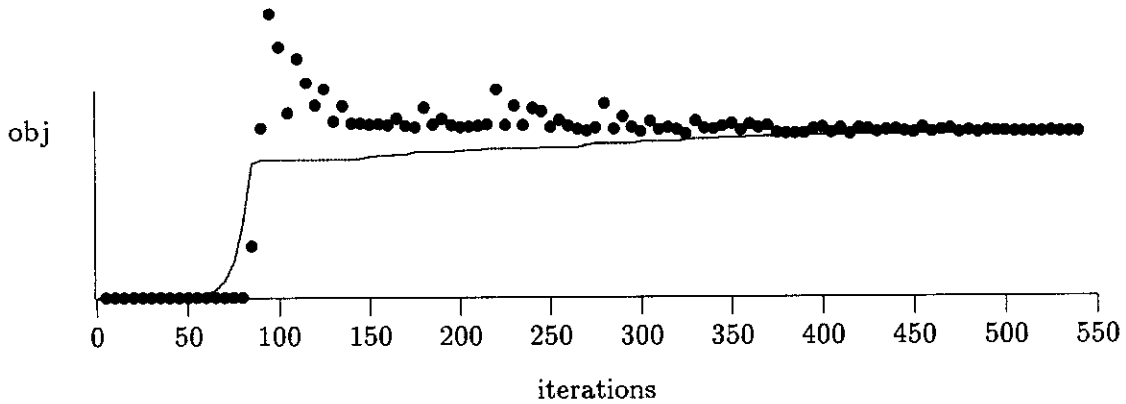

Figure 6.10

In Table 6.11 we present the computing times for two instances taken from [5]. We did not compare with the simplex method because we are taking advantage of the special structure.

TABLE 6.11

locations	customers	time (secs)
250	163	39.2
276	241	48.8

7. Final Remarks

We have presented a way to produce approximate solutions in Dantzig-Wolfe decomposition with a subgradient algorithm. The instances where we had success share the following properties:

- All variables are bounded between 0 and 1.
- All coefficients in the matrix are 0, 1 or -1.
- Problem (2.6) could be solved in linear time, with respect to the size of the input.

The first two imply that if π^t and π^{t+1} are "close", then the constraints

$$z + \pi(Ax^t - b) \leq cx^t$$

and

$$z + \pi(Ax^{t+1} - b) \leq cx^{t+1}$$

of (2.5) do not form a "sharp" angle. We believe that this is a key property necessary for this type of approximate algorithm. While combinatorial problems seem to be very good candidates, we expect problems with more general coefficients to be more difficult for this approach.

This algorithm is very easy to implement. Other than keeping the original data, the storage requirements are minimal. Since no matrix inversion is required, no numerical difficulties arise.

Another attractive feature of this algorithm is that it can be trivially parallelized. For instance, for solving (6.1) the most expensive operation is to compute $(c - \pi A)$. Here each column can be treated independently by different processors. The other expensive operation is to compute $v = b - A\bar{x}$, in Step 1. Here each row can be treated independently.

References

[1] R. ANBIL, E. L. JOHNSON, AND R. TANGA, *A global approach to crew-pairing optimization*, IBM Systems Journal, 31 (1992), pp. 71–78.

[2] K. M. ANSTREICHER AND L. WOLSEY, *On dual solutions in subgradient optimization*, Technical report, 1993.

[3] B. M. BAKER AND J. SHEASBY, *Accelerating the convergence of subgradient optimization*, tech. rep., Coventry University, 1996.

[4] E. BALAS AND A. HO, *Set covering algorithms using cutting planes, heuristics, and subgradient optimization*, Math. Programming Study, 12 (1980), pp. 37–60.

[5] F. BARAHONA AND D. JENSEN, *Plant location with minimum inventory*, Research Report RC 20367, IBM, T. J. Watson Research Center, 1996.

[6] F. BARAHONA AND A. MAHJOUB, *On the cut polytope*, Mathematical Programming, 36 (1986), pp. 157–173.

[7] U. BRÄNNLUND, *A generalized subgradient method with relaxation step*, Mathematical Programming, 71 (1995), pp. 207–219.

[8] P. M. CAMERINI, L. FRATTA, AND F. MAFFIOLI, *On improving relaxation methods by modified gradient techniques*, Mathematical Programming Study, 3 (1975), pp. 26–34.

[9] A. CAPRARA, M. FISCHETTI, AND P. TOTH, *A heuristic method for the set covering problem*, Technical Report OR-95-8, University of Bologna, 1995.

[10] S. CERIA, P. NOBILI, AND A. SASSANO, *A lagrangian based heuristic for large scale set covering problems*, Technical Report 406, IASI-CNR-Rome, 1995.

[11] G. CORNUEJOLS, M. L. FISHER, AND G. L. NEMHAUSER, *Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms*, Management Sci., 23 (1977), pp. 789–810.

[12] G. CORNUEJOLS AND J. M. THIZY, *A primal approach to the simple plant location problem*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 504–510.

[13] I. B. M. CORP., *Optimization subroutine library: Guide and reference*, (1995).

[14] G. B. DANTZIG AND P. WOLFE, *Decomposition principle for linear programs*, Oper. Res., 8 (1960), pp. 101–111.

[15] C. DE SIMONE AND G. RINALDI, *A cutting plane algorithm for the max-cut problem*, Optimization Methods and Software, 3 (1994), pp. 195–214.

[16] D. ERLENKOTTER, *A dual-based procedure for uncapacitated facility location*, Oper. Res., 26 (1978), pp. 992–1009.

[17] J. ETCHEBERRY, *The set-covering problem: A new implicit enumeration algorithm*, Operations Research, 25 (1977), pp. 760–772.

[18] R. S. GARFINKEL, A. W. NEEBE, AND M. R. RAO, *An algorithm for the m-median plant location problem*, Transportation Sci., 8 (1974), pp. 217–236.

[19] M. HELD AND R. M. KARP, *The travelling salesman problem and minimum spanning trees*, Oper. Res., 18 (1970), pp. 1138–1162.

[20] ——, *The travelling salesman problem and minimum spanning trees: Part II*, Mathematical Programming, 1 (1971), pp. 6–25.

[21] M. HELD, P. WOLFE, AND H. P. CROWDER, *Validation of subgradient optimization*, Mathematical Programming, 6 (1974), pp. 62–88.

[22] T. LARSON AND Z. LIU, *A primal convergence result for dual subgradient optimization with application to multicommodity network flows*, Research Report S-581 83, Dept. of Math, Linkoping Institute of Technology, 1989.

[23] C. LEMARÉCHAL, *An extension of Davidon methods to non differentiable problems*, Math. Programming Stud., 3 (1975), pp. 95–109.

[24] ——, *Nondifferentiable optimization*, in Optimization, Hanbooks in Operations Research, G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds., North Holland, 1989, pp. 529–572.

[25] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer and Combinatorial Optimization*, Wiley, New York, 1988.

[26] B. T. POLYAK, *Minimization of unsmooth functionals*, U.S.S.R. Comput. Math. and Math. Phys., 9 (1969), pp. 509–521.

[27] H. D. SHERALI AND G. CHOI, *Recovery of primal solutions when using subgradient optimization methods to solve lgrangian duals of linear programs*, Operations Research Letters, 19 (1996), pp. 105–113.

[28] N. Z. SHOR, *Minimization methods for nondifferentiable functions*, Springer, Berlin, 1985.

[29] P. WOLFE, *A method of conjugate subgradients for minimizing nondifferentiable functions*, Mathematical Programming Study, 3 (1975), pp. 145–173.