On some difficult linear programs coming from Set Partitioning

Francisco Barahona[1]
Ranga Anbil[1]

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

Abstract

We deal with the linear programming relaxation of set partitioning problems arising in airline crew scheduling. Some of these linear programs have been extremely difficult to solve with the traditional algorithms. We have used an extension of the subgradient algorithm, the *volume algorithm*, to produce primal solutions that might violate the constraints by at most 2%, and that are within 1% of the lower bound. This method is fast, requires minimal storage, and can be parallelized in a straightforward way.

Key words. Subgradient algorithm, volume algorithm, large scale linear programming.

AMS subject classifications. 90C05, 90C06.

1. Introduction

Set partitioning problems arise in airline crew scheduling when one has to select crew trips to cover a set of flights. The crew trips are given by a column generation procedure. Because of the large dimensions involved, one way to tackle this is to first solve a linear programming relaxation, then choose a set of variables to be fixed to 1, and generate more columns. When a large set of variables have been fixed, a traditional branch and bound procedure is applied.

These linear programs can be described as

$$\begin{aligned} &\text{minimize } cx \\ &Ax = \mathbb{1} \\ &x \geq 0, \end{aligned} \tag{1.1}$$

where A is a matrix with 0-1 coefficients, and $\mathbb{1}$ denotes a vector of ones. Each row of A corresponds to a flight leg that must be staffed, each column of A corresponds to a legal crew trip, and the components of c correspond to the costs of the trips.

Despite all the progress in linear programming, solving these LP relaxations can be a challenge. In general, the dual simplex method works better than primal simplex. Interior point algorithms tend to work better, but they might require large amounts of storage. In some cases, solving one of these LPs with the traditional methods can take more than 10 hours on a fast workstation. Due to the large size of these problems, we need a fast procedure that produces good approximate solutions. This is particularly important in the early stages of the column generation procedure, when an exact solution to the LP might not be required. Producing approximate solutions to large linear programs is an area that needs more study. Our work is one step in that direction, and other work in the same direction appears in [8, 18, 6]. The approach presented here can be applied to many other combinatorial problems.

We use subgradient techniques because the subgradient algorithm is easy to implement, fast and produces very good dual solutions. However, in its original form it does not produce primal solutions. We have extended it to the so-called *volume algorithm* that produces dual solutions as well as approximate primal solutions. This procedure is very simple, requires minimal storage, decreases the computing time dramatically, and can be parallelized in a trivial way. As we shall see, when this procedure is followed by the simplex method, it can yield a great acceleration of the latter.

Subgradient techniques have been used for set covering problems see [15, 5, 11, 10]. In these articles, different heuristics, based on the dual vectors, are used to produce primal integer solutions. We believe that the primal information produced by the volume algorithm would enhance these procedures.

In this computational study, we concentrate on producing fast approximate solutions to the LP relaxation. For this reason, we have left out all the integer programming aspects; we shall address them in subsequent publications.

This paper is organized as follows. In Section 2 we describe the instances treated and their solution with traditional methods. Section 3 is devoted to the subgradient method. In Section 4 we deal with the volume algorithm. In Section 5 we study the "cross over" problem. Some final remarks appear in Section 6.

2. The Test Set

In this section we describe our test problems, which come from the approach to airline crew scheduling described in [2]. For this study we have chosen particularly challenging instances from a larger test set. They are available from the authors for similar computational studies.

The table below shows the number of rows and columns, then the time and storage required by the dual simplex method, and then similar information for a primal-dual barrier method. The computing time of the barrier method does not include the time needed to cross over to an optimal basis. We also give the optimal values; they may be used for comparisons with the bounds presented later. All computations were done on an IBM RS 6000/590 with the OSL package [12].

TABLE 2.1

Name	Rows	Columns	Dual Simplex (hh:mm)	(MB)	Barrier (hh:mm)	(MB)	Optimum
sp6	2504	50722	1:01	70	0:21	70	157414
sp7	2991	43459	1:40	70	0:34	102	162350
sp8	4810	91123	6:28	77	1:03	187	368265
sp9	2917	50013	0:46	104	0:13	104	166704
sp12	3218	84746	4:19	107	0:54	107	248004
sp13	3928	58051	2:56	104	0:28	104	347151
sp14	3217	47214	2:55	100	0:38	100	250196
sp15	10764	207205	12:28	148	19:02	438	69371
sp16	4835	144888	31:42	200	3:48	280	490865

This table shows that the interior point method was faster than dual simplex in most cases. One drawback of the interior point method is the large amount of storage needed for larger problems. A more serious drawback is that in integer programming one needs to continually reoptimize after adding cutting planes or after fixing variables; this is not a well resolved issue for interior point methods.

3. The Subgradient Algorithm

For a vector of dual multipliers π, a *Lagrangian relaxation* of (1.1) is

$$\left.\begin{aligned} z(\pi) = \min\ & (c - \pi A)\, x + \pi \mathbb{1} \\ 0 \le\ & x \le \mathbb{1}. \end{aligned}\right\} \tag{3.1}$$

The value $z(\pi)$ is a lower bound on the optimal value of (1.1). One can try to maximize z with the subgradient algorithm. Since the work of Held and Karp [19, 20] and Held, Wolfe and Crowder [21], in the early seventies, this algorithm has been used to produce lower bounds for large-scale linear programs. The main loop of iteration $j \ge 0$ consists of the two steps below.

Step 1. Given $\bar{\pi}^j$, solve (3.1) with $\pi = \bar{\pi}^j$ to obtain its solution $\bar{x}^j$. Then $v^j = \mathbb{1} - A\bar{x}^j$ is a subgradient of the (concave) function z at $\bar{\pi}^j$.

Step 2. Compute $\bar{\pi}^{j+1} = \bar{\pi}^j + s_j\, v^j$. Here $s_j > 0$ is a *stepsize*.

If the sequence of stepsizes $\{s_j\}$ satisfies

$$s_j \to 0, \quad \sum_{j=0}^{\infty} s_j = \infty, \tag{3.2}$$

then (cf. [28]) $\limsup_{j\to\infty} z(\bar{\pi}^j) = \max z$ =the optimal value of (1.1).

If the step size is chosen as

$$s_j = \lambda_j \frac{\hat{z} - z(\bar{\pi}^j)}{\|v^j\|^2}, \tag{3.3}$$

where $\hat{z} < \max z$, and $\epsilon < \lambda_j \leq 2 - \epsilon$, for a fixed $\epsilon \in (0, 1]$ then either $z(\bar{\pi}^j) \to \hat{z}$ or a point $\bar{\pi}^j$ is found with $z(\bar{\pi}^j) \geq \hat{z}$; see [29]. Other authors have given convergence proofs for other choices of the step-size, see [14, 16, 30, 7, 17, 27, 1, 22, 25, 23, 24] for examples.

On the other hand, most practitioners use a heuristic choice of the step size proposed by Held et al [21], as follows. An overestimate is used instead of the underestimate $\hat{z}$ in (3.3). Then λ_j is chosen as a fixed value λ that is periodically decreased by some factor. This choice of the step size violates the hypothesis of (3.2) and (3.3). The choice of the step size is one aspect of subgradient optimization that is not well understood.

One drawback of this algorithm is that, as in the steepest ascent method, the direction only depends on the last point, and all the information given by the previous iterations is ignored. The second drawback is that, since it does not produce primal variables, one has no idea of the distance from optimality.

In the early seventies Crowder [13] proposed the following modification of Step 2

$$\bar{\pi}^{j+1} = \bar{\pi}^j + s_j\, d^j,$$

where the direction d^j is set to v^0 for $j = 0$, and is updated for $j \geq 1$ via

$$d^j = d^{j-1} + \theta v^j,$$

for a fixed value θ, $0 < \theta \leq 1$. He presented it as a way to avoid zig-zag, without losing the simplicity of the algorithm. Other ways to avoid zig-zag have been proposed in [9, 4].

We implemented the update

$$d^j = (1 - \alpha)d^{j-1} + \alpha v^j,$$

proposed in [4], where $\alpha \in (0, 1]$. We started with $\alpha = 0.1$, then every 100 iterations we checked if the objective had increased by at least 1%. If not, α was divided by 2, unless it was already less than 10^{-5}, in which case it was kept constant. This should be seen as an attempt to increase the precision when computing the direction. We call this method *Modified subgradient* (M-Sbg). As we discussed in the next section, our method uses a similar idea when working with the dual variables, but it also produces primal variables.

Table 3.4 shows the lower bound produced by the subgradient method and by the modification above (M-Sbg). Both methods stopped after 100 iterations without improvement. For both methods the initial vector was $\pi^0 = 0$. For the step size, we used a modification of formula (3.3) described in Section 4, see formula (4.2). In the denominator, one has to use $\|v^j\|^2$ for the subgradient algorithm and $\|d^j\|^2$ for the second method. In all cases, the second bound was much better than the one given by the original subgradient method.

TABLE 3.4

Name	Subgradient	M-Sbg
sp6	128072	155540
sp7	137725	160147
sp8	310617	364450
sp9	127082	164298
sp12	128141	245283
sp13	159886	339710
sp14	119083	247152
sp15	59996	68650
sp16	126141	416431

4. The Volume Algorithm

As described in the last section, the subgradient algorithm or its modification is computationally very attractive. Its main drawback is that it does not produce values for the primal variables. In [6] we extended the subgradient algorithm, so that with the same computational effort per iteration, it could produce primal variables as well as dual variables. This is called the *volume algorithm*. This name reflects the fact that primal values come from computing the volume below the faces of the dual problem. The direction of movement is also given by these volumes. Its convergence has been studied in [3]. Its description is below.

Volume Algorithm

Step 0. Starting with a vector $\bar{\pi}$, solve (3.1) with $\pi = \bar{\pi}$ to obtain its solution $\bar{x} = x^0$ and $\bar{z} = z(\bar{\pi})$. Set $t = 1$.

Step 1. Compute $v^t = \mathbb{1} - A\bar{x}$ and $\pi^t = \bar{\pi} + sv^t$ for a step size s given by (4.2). Solve (3.1) with $\pi = \pi^t$ to get its solution x^t and $z^t = z(\pi^t)$. Update $\bar{x}$ 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.

Notice that in Step 2 we update $\bar{\pi}$ only if $z^t > \bar{z}$, so this is an ascent method. We are trying to mimic the bundle method [26], but we want to avoid the extra effort of solving a quadratic problem at each iteration.

One difference with the subgradient algorithm is the use of the formula (4.1). If $x^0, \ldots, x^t$ is the sequence of vectors produced by problem (3.1), 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 $\{x^0, \ldots, x^t\}$. The assumption that this sequence approximates an optimal solution of (1.1) is based on a theorem in linear programming duality that appears in [6]. Notice the exponential decrease of the coefficients of this convex combination; latest vectors thus receive much larger weights than earlier ones. At every iteration the direction is being updated as in the modified subgradient method, so this is a method with "memory". Thus, it does not have the same zig-zagging behavior of the subgradient method.

Here the formula for the step size is

$$s = \lambda \frac{T - \bar{z}}{\|v^t\|^2}, \tag{4.2}$$

where λ is a number between 0 and 2, and T is a *target* value. We started with a small value for T, and each time that $\bar{z} \geq 0.95\, T$, we increased T to $T = 1.05\, \bar{z}$.

In order to set the value of λ we define three types of iterations: red, yellow and green.

- *Red.* Each time that we do not find an improvement (i.e. $z^t \leq \bar{z}$), we call this iteration red. A sequence of red iterations suggests the need for a smaller stepsize.
- *Yellow.* If $z^t > \bar{z}$ we compute

  $$d = v^t \cdot (\mathbb{1} - Ax^t).$$

  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.
- *Green.* If $d \geq 0$ we call this iteration green. A green iteration suggests the need for a larger stepsize.

At each green iteration, we multiplied λ by 1.1. If the result was greater than 2, we set $\lambda = 2$. After a sequence of 20 consecutive red iterations, we multiplied λ by 0.66, unless $\lambda < 0.0005$, in which case we kept it constant.

The value of α in (4.1) was chosen as the solution of the following 1-dimensional problem:

$$\left.\begin{array}{l} \text{minimize } \|\mathbb{1} - A(\alpha x^t + (1-\alpha)\bar{x})\| \\ \text{subject to} \\ \dfrac{u}{10} \leq \alpha \leq u. \end{array}\right\} \tag{4.3}$$

The value u was originally set to 0.1 and then every 100 iterations we checked if $\bar{z}$ had increased by at least 1%. If not, we divided u by 2, unless u was already less than 10^{-5}, in which case it was kept constant. Each time that u was decreased we noticed a decrease in the sum of the primal infeasibilities. This choice of α is very similar to the one proposed in [31]; the difference is in the bounds $u/10$ and u.

Table 4.4 shows the results given by the volume algorithm. As in the last section, the initial vector was $\bar{\pi} = 0$. First we show the lower bound, then the value of the primal vector (i.e., $c\bar{x}$). We accepted a primal vector only if each constraint was violated by at most 0.02. The algorithm terminated when this condition was satisfied, and the difference between the lower bound and the value of the primal vector was less than 1% (i.e., $|c\bar{x} - \bar{z}| < 0.01|\bar{z}|$). We also present the time and the storage required.

TABLE 4.4

name	Volume				
	l b	primal	max viol	(hh:mm)	MB
sp6	157109	158688	0.02	0:24	10
sp7	161548	162853	0.02	0:18	10
sp8	367837	371512	0.02	1:29	19
sp9	166247	166930	0.02	0:26	13
sp12	247283	249020	0.02	0:28	17
sp13	346751	349170	0.02	0:34	13
sp14	249454	251959	0.02	0:19	10
sp15	69238	69983	0.02	1:57	43
sp16	484482	489383	0.02	2:16	40

5. Crossing Over

Starting from an approximate solution, one might want to produce a primal feasible vector with low computational effort. This is a question with great practical interest, and not much research has been done on it. We describe our procedure below.

From the vectors $\bar{\pi}$ and $\bar{x}$, produced by the volume algorithm, we computed the reduced costs $\bar{c} = c - \bar{\pi}A$. We chose a set of columns S, with the 20,000 smallest reduced costs, and then from the remaining columns, we added to S those with $\bar{x}_j > 10^{-3}$.

To achieve dual feasibility, for successive $j \in S$, if $\bar{c}_j < 0$ then we computed

$$\epsilon = \frac{\bar{c}_j}{\sum_i a_{ij}},$$

and updated $\bar{\pi}$ as

$$\bar{\pi}_i \leftarrow \bar{\pi}_i + \epsilon a_{ij}.$$

After each update of $\bar{\pi}$ the reduced costs $\bar{c}$ had to be updated. Let $\bar{\bar{\pi}}$ be the final vector obtained, then $z(\bar{\bar{\pi}}) = z(\bar{\pi})$ in (3.1).

Finally we applied the dual simplex method to

$$\begin{aligned} &\text{minimize } \bar{c}x \\ &\bar{A}x = \mathbb{1} \\ &x \geq 0, \end{aligned} \tag{5.1}$$

where $\bar{A}$ consists of the columns of A in S, and $\bar{c}_j = c_j - \sum_i \bar{\pi}_i a_{ij}$, for $j \in S$.

Notice that we used the reduced costs instead of the original costs in (5.1). We have observed that this is much better for the convergence of the dual simplex algorithm. For instance, when we tried the original costs for problem sp15, it took 2:08 hours instead of 17 minutes, (see Table 5.2).

The table below contains: the number of columns in S, the time and storage needed by the dual simplex method, the total time (V+D) taken by the volume algorithm and dual simplex, and finally, the objective value obtained. Because we are considering a reduced set of columns, the objective values can be slightly higher than those in Table 2.1.

TABLE 5.2

name	columns	Dual Simplex (hh:mm)	(MB)	V+D (hh:mm)	Objective
sp6	24802	0:04	45	0:28	157414
sp7	24886	0:10	45	0:28	162350
sp8	30476	0:14	47	1:43	368268
sp9	25248	0:04	46	0:30	166704
sp12	28370	0:14	47	0:42	248004
sp13	26788	0:07	46	0:41	347185
sp14	25261	0:12	45	0:31	250199
sp15	45236	0:17	54	2:14	69372
sp16	29763	2:25	50	4:41	491014

6. Concluding Remarks

For the set partitioning instances studied, the volume algorithm produced approximate primal solutions with a maximum violation of 2% with a value within 1% of the lower bound. Then the "cross over" procedure of Section 5 produced primal feasible vectors. This approach seems appropriate for many other combinatorial problems where the linear programming relaxation is difficult to solve and only gives an approximation of an integer solution.

The procedure described in this paper is not only fast but also requires minimal storage: just the matrix A, and a few vectors. Its other attractive feature is that it can be parallelized in a straightforward way. At each iteration the two most expensive operations are the computation of $c - \pi A$ and $v = \mathbb{1} - Ax$. The first operation can be decomposed per columns, and the second one can be decomposed per rows. This compares very favorably with algorithms that require pivoting, matrix inversion, matrix multiplication or solving systems of equations at each iteration.

Acknowledgments. We are grateful to both referees whose suggestions have greatly improved the presentation.

References

[1] E. ALLEN, R. H. J. KENNINGTON, AND B. SHETTY, *A generalization of Polyak's convergence result for subgradient optimization*, Math. Programming, 37 (1987), pp. 309–317.

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

[3] L. BAHIENSE, N. MACULAN, AND C. SAGASTIZÁBAL, *On the convergence of the volume algorithm*, tech. rep., 2000.

[4] B. M. Baker and J. Sheasby, *Accelerating the convergence of subgradient optimization*, tech. rep., Coventry University, UK, 1996.

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

[6] F. Barahona and R. Anbil, *The volume algorithm: producing primal solutions with a subgradient method*, Mathematical Programming, 87 (2000), pp. 385–399.

[7] M. S. Bazaraa and H. D. Sherali, *On the choice of step size in subgradient optimization*, European J. Oper. Res., 7 (1981), pp. 380–388.

[8] D. Bienstock, *Experiments with a network design algorithm using ϵ-approximate linear programs*, tech. rep., Columbia University, 1996.

[9] P. M. Camerini, L. Fratta, and F. Maffioli, *On improving relaxation methods by modified gradient techniques*, Mathematical Programming Study, 3 (1975), pp. 26–34.

[10] A. Caprara, M. Fischetti, and P. Toth, *A heuristic method for the set covering problem*, Operations Research, 47 (1999), pp. 730–743.

[11] S. Ceria, P. Nobili, and A. Sassano, *A lagrangian based heuristic for large scale set covering problems*, Mathematical Programming, 81 (1998), pp. 215–228.

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

[13] H. Crowder, *Computational improvements for subgradient optimization*, in Symposia Mathematica, Vol XIX, Academic Press, London, 1976, pp. 357–372.

[14] Y. M. Ermoliev, *Methods of solution of nonlinear extremal problems*, Kibernetika, 4 (1966), pp. 1–17.

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

[16] J. L. Goffin, *On convergence rates of subgradient optimization methods*, Math. Programming, 13 (1977), pp. 329–347.

[17] ——, *Convergence results in a class of variable metric subgradient methods*, in Nonlinear Programming 4, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., New York, 1981, Academic Press, pp. 283–326.

[18] A. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein, *An implementation of a combinatorial approximation algorithm for minimum cost multicommodity flows*, Tech. Rep. STAN-CS-TR-97-1600, Stanford University, 1997.

[19] M. Held and R. M. Karp, *The travelling salesman problem and minimum spanning trees*, Operations Research, 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] S. Kim, H. Ahn, and S. C. Cho, *Variable target value subgradient method*, Math. Programming, 49 (1991), pp. 359–369.

[23] K. C. Kiwiel, *The efficiency of subgradient projection methods for convex optimization, part I: General level methods*, SIAM J. Control Optim., 34 (1996), pp. 677–697.

[24] K. C. Kiwiel, T. Larsson, and P. O. Lindberg, *The efficiency of ballstep subgradient level methods for convex optimization*, Math. Oper. Res., 24 (1999), pp. 237–254.

[25] A. N. Kulikov and V. R. Fazilov, *Convex optimization with prescribed accuracy*, Zh. Vychisl. Mat. i Mat. Fiz., 30 (1990), pp. 663–671.

[26] C. Lemaréchal, *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.

[27] Y. E. Nesterov, *Minimization methods for nonsmooth convex and quasiconvex functions*, Matekon, 29 (1984), pp. 519–531.

[28] B. T. Polyak, *A general method for solving extremum problems*, Soviet. Math. Dokl., 8 (1967), pp. 593–597.

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

[30] N. Z. Shor, *Minimization methods for nondifferentiable functions*, Springer, Berlin, 1985.

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