Large-Scale Linear Programming and Applications

Stephen J. Stoyan, Maged M. Dessouky*, and Xiaoqing Wang

Daniel J. Epstein Department of Industrial and Systems Engineering,

University of Southern California, Los Angeles, CA, 90089-0193

*Corresponding author (e-mail: maged@usc.edu)

1 Introduction

Linear Programming has been used as an effective tool to solve a number of applications in manufacturing, transportation, military, healthcare, and finance. In a number of applications, the size of the formulation in terms of the number of variables and constraints can be quite large in order to accurately model practical systems. For example, Leachman (1993) formulates a linear programming model of over 150,000 variables and 100,000 constraints to solve a production planning problem for semiconductors at the Harris Corporation. With such large size problems, specialized procedures such as column generation, cutting planes, and the Dantzig-Wolfe decomposition may need to be employed to solve them. The purpose of this chapter is to provide an overview of these specialized procedures, and we conclude with an example of a large-scale formulation for solving a route balancing problem.

2 Linear Programming Problem

Consider the following standard Linear Programming (LP) problem:

$$\min \quad \mathrm{c}^\top x \tag{1}$$

$$\text{s.t.} \quad \mathrm{A}x = \mathrm{b} \tag{2}$$

$$x \geq 0, \tag{3}$$

where $x \in \mathbb{R}^n$, A is a $n \times m$ matrix, and c and b are vectors of length n and m; respectively. Now suppose the problem is so large, i.e. the A matrix cannot be stored in memory, that it cannot be solved. Using the simplex method, large-scale problems can be solved by adding a few extra steps to the algorithm. Depending on the structure of the A matrix it may be possible to decompose the problem such that solving smaller subproblems provide sufficient optimality conditions. Recall from the simplex method, the problem is partitioned into Basic (B) and Nonbasic (N) variables. Partitioning (1)–(3) into basic and nonbasic elements and making the substitution into (1) gives

$$\min \quad \mathrm{c}_B^\top \mathrm{B}^{-1}\mathrm{b} + (\mathrm{c}_N^\top - \mathrm{c}_B^\top \mathrm{B}^{-1}\mathrm{N})x_N, \tag{4}$$

where $\mathrm{B} = \sum_i \mathrm{A}_i \; \forall \; i \in B$, $\mathrm{N} = \sum_j \mathrm{A}_j \; \forall \; j \in N$, and A_n refers to column n in the A matrix. The algorithm then pivots through columns which allow nonbasic elements to enter the basis in each iteration until the optimality condition is satisfied; namely

$$\mathrm{c}_j^\top - \mathrm{c}_B^\top \mathrm{B}^{-1}\mathrm{A}_j \geq 0 \quad \forall \; j \in N. \tag{5}$$

Therefore, given $n > m$, N consists of $n - m$ elements and the simplex algorithm will iteratively pivot through nonbasic variables that have a minimum reduced cost, defined as

$$\min_{j \in N} \{\mathrm{c}_j^\top - \mathrm{c}_B^\top \mathrm{B}^{-1}\mathrm{A}_j\} < 0. \tag{6}$$

Equation (6) determines whether or not the simplex algorithm will continue to the next iteration, notice that if it is nonnegative then equation (5) is satisfied and the algorithm terminates. This observation is a critical step in many large-scale algorithms. Given the current basis, only search for new variables to pivot into the basis if the minimum of $\{\mathrm{c}_j^\top - \mathrm{c}_B^\top \mathrm{B}^{-1}\mathrm{A}_j\}$ is *not* nonnegative $\forall\ j \in N$; otherwise, the current iterate is optimal. In the next sections we investigate such applications.

3 Column Generation

Many practical problems involve large-scale models where there are more variables than constraints; hence, $n > m$. For such instances, column generation is an effective solution method. The method begins by initiating the simplex algorithm as usual. Hence, a collection of m basic columns $\mathrm{A}_i\ \forall\ i \in B$ is formed and we search for variable $j \in N$ that has the minimum reduced cost, where we define the set $\Omega = \{i \in B, j\}$. In order to find variable x_j with the minimum reduced cost, solving for j in equation (6) would be most favorable; however, this may be very costly or not possible depending on the problems size. Instead, taking the

$$\min_{j \in N} \ \mathrm{c}_j \tag{7}$$

may accomplish the same result. There are other variants of reduced cost searches, however, for lack of space we refer the reader to [1, 2, 4]. In any case, a search is completed and an entering variable j is selected. We define the subproblem

$$\min \quad \sum_{\omega \in \Omega} \mathrm{c}_\omega^\top x_\omega \tag{8}$$

$$\text{s.t.} \quad \sum_{\omega \in \Omega} \mathrm{A}_\omega x_\omega = \mathrm{b} \tag{9}$$

$$x \geq 0, \tag{10}$$

which contains all of the basic columns and the entering column A_j. Therefore, Ω has a size of $m+1$ and if the original A matrix is very large then we only need enough storage to hold $m+1$ columns. Furthermore, using the current basic feasible solution and the entering element, one may perform as many simplex iterations as needed to solve the problem; allowing one nonbasic variable to enter the basis at each iteration. In the absence of degeneracy, solving (8)–(10) by adding columns in an iterative manner is guaranteed to terminate since it is simply a special implementation of the simplex algorithm. If degeneracy is present in the problem, one can use a method to prevent cycling such as the lexicographic rule. Finally, there are different techniques used to store the number of variables in Ω. In the method described above, the size of Ω is kept to a minimum of $m+1$ variables. Other methods never remove any variables that have entered the basis, and some allow Ω to grow to a certain size before removing variables.

There are also various problems where instead of solving a large set of decisions, the problem may be more efficiently approached if it is structured to generate candidate solutions (columns) until a better solution cannot be found. Column generation can be used to solve such problems, which exist in inventory, transportation, and scheduling. In Section 6 we provide a route balancing example where road crews are assigned to clean various areas of Los Angeles based on distance. As will be presented, the problem can be expressed as

$$\min \quad \mathrm{e}^\top x \tag{11}$$

$$\text{s.t.} \quad \mathrm{A}x = \mathrm{b} \tag{12}$$

$$x \geq 0, \tag{13}$$

where e is a vector of all ones. As mentioned above, matrix A may not involve the set of all possible combinations to solve the original problem and candidate solutions will be generated to do so. In this case, candidate solutions are equivalent to adding columns A_j to the problem. In the route balancing example, finding a candidate solution (column A_j)

simply requires finding an area that a crew may be allocated to. Thus, providing an initial basis to (11)–(13) involves generating m different candidate solutions or columns A_j. Now suppose that we have an initial basis B, then to solve for the reduced cost in (11)–(13) we take

$$\min_{j \in N} \{1 - \mathrm{e}^\top \mathrm{B}^{-1} \mathrm{A}_j\}. \tag{14}$$

Letting $\mu = \mathrm{e}^\top \mathrm{B}^{-1}$, the problem is optimal if

$$\min_{j \in N} \{1 - \mu \mathrm{A}_j\} \geq 0. \tag{15}$$

Notice that equation (15) is equivalent to the following inequality:

$$\max_{j \in N} \{\mu \mathrm{A}_j\} \leq 1, \tag{16}$$

Thus, the optimality of the original problem requires obtaining the largest candidate solution that satisfies (16). Depending on the number of constraints, finding the largest column that satisfies (12)–(13) may simply require solving the knapsack problem, provided decisions are purely integer variables.

4 Cutting Planes

Using the models presented in Sections 2 and 3, but investigating the dual problem defines the *Cutting Plane* method. Consider the dual of (1)–(3),

$$\max \quad \mathrm{b}^\top y \tag{17}$$

$$\text{s.t.} \quad \mathrm{A}^\top y \leq \mathrm{c}. \tag{18}$$

Since we have taken the dual of the original problem, we are now faced with a problem that has many rows or constraints. We then define a subset of (17)–(18) as follows:

$$\max \quad \mathrm{b}^\top y \tag{19}$$

$$\text{s.t.} \quad \sum_{\omega \in \Omega} \mathrm{A}_\omega^\top y_\omega \leq \mathrm{c}_\omega, \tag{20}$$

where Ω contains the elements in the basis and grows with each j added to the set, as described on page 4 of Section 3. Thus, (19)–(20) is simply a less-constrained version of (17)–(18), referred to as the relaxed dual problem. Given $z_\Omega^\star$ is an optimal solution to (19)–(20), there are two issues to consider with respect to the optimality of (17)–(18), namely

(i) If $z_\Omega^\star$ is a feasible solution to (17)–(18), then $z_D^\star = z_\Omega^\star$; where $z_D^\star$ refers to the optimal solution of (17)–(18);

(ii) If $z_\Omega^\star$ is infeasible with respect to (17)–(18), then we find the violated constraint and add it to subproblem (19)–(20). Thus, given row $\mathrm{A}_j^\top$ provides the violation, then set Ω is updated to include j and the problem is re-solved.

Figure 1 is a graphical illustration of cases (i) and (ii) above. Given the feasible region of the original problem is A_1, if constraint L_4 is not included in subproblem (19)–(20), then the feasible region is larger and contains regions $A_1 \bigcup A_2$. In this case, solving the subproblem produces a feasible optimal solution to (17)–(18) and thus, $z_\Omega^\star = z_D^\star$, as depicted in the figure. However, what if constraint L_2 is not included in subproblem (19)–(20), then the feasible region contains $A_1 \bigcup A_3$. Depending on the slope of the objective function, the optimal solution may be at vertex $z_\Omega^\star$, as depicted in the figure. In any case, $z_\Omega^\star \neq z_D^\star$ as the vertex $z_D^\star$ does not exist for such an instance, and hence, the simplex method will not generate the optimal solution $z_D^\star$.

Therefore, based on the observations above, in order to implement a cutting plane algorithm one needs to solve the current subproblem and check that the solution is feasible with

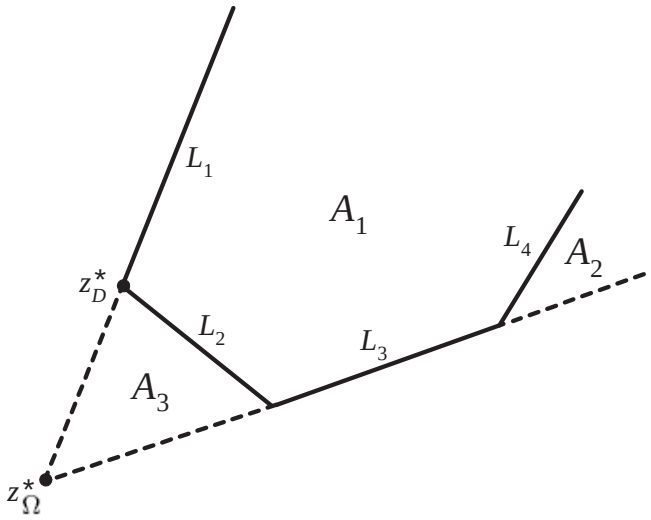

Figure 1: Illustrative example of issues related to *Cutting Planes*.

respect to the original problem. If the solution is not feasible, an efficient method to identify the violated constraint(s) is necessary. This requires finding

$$\min_{\ell} \quad \mathrm{c}_\ell - \mathrm{A}_\ell^\top y_\ell \tag{21}$$

for $\ell \notin \Omega$. If (21) is nonnegative for all elements ℓ then we have a feasible and therefore, optimal solution. Otherwise, we add constraint $\mathrm{A}_\ell^\top y_\ell > \mathrm{c}_\ell$ to subproblem (19)–(20) and continue. Depending on the problem, solving equation (21) may be challenging. There are techniques that can be employed for such instances; the reader may refer to [1, 2, 6] for more information on this topic.

5 Dantzig-Wolfe Decomposition

Given the LP in (1)–(3) of Section 2, what if constraint (2) has the following structure:

$$\begin{bmatrix} A_{01} & A_{02} & \cdots & A_{0\ell} \\ A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{\ell\ell} \end{bmatrix} x = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_\ell \end{bmatrix}, \tag{22}$$

where $A_{0\ell}$ and $A_{\ell\ell}$ represent a set of partitioned parameters in the form of block matrices that contain all of the elements in the initial problem. The constraint structure presented in (22) is quite common for many practical problems. For example, problems in production, finance, and transportation often involve models where some subset of constraints only pertain to particular elements/areas of the problem and others span the whole domain. For instances where various constraints only pertain to one or a small set of variables (i.e. rows $1, ..., \ell$), and only one constraint or set of constraints involves all variables (i.e. row 0), the Dantzig-Wolfe decomposition is an efficient solution approach. For the prescribed problem structure, the algorithm proceeds by decomposing the A matrix as follows:

$$\begin{bmatrix} A_{01} & A_{02} & \cdots & A_{0\ell} \end{bmatrix} x = \begin{bmatrix} b_0 \end{bmatrix} \tag{23}$$

$$\begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{\ell\ell} \end{bmatrix} x = \begin{bmatrix} b_1 \\ \vdots \\ b_\ell \end{bmatrix}. \tag{24}$$

Thus, the problem in (1)–(3) can be represented as:

$$\min \quad \sum_{i=1}^{\ell} c_i x_i \tag{25}$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} D_i x_i = b_0 \tag{26}$$

$$F_i x_i = b_i, \qquad \forall\, i = 1, ..., \ell, \tag{27}$$

$$x_i \geq 0, \qquad \forall\, i = 1, ..., \ell, \tag{28}$$

where $1, ..., \ell$ contains all partitioned elements from the initial problem, and we let $D_i = A_{0i}$ and $F_i = A_{ii}$ for simplicity. Next we define

$$\chi_i = \{x_i \geq 0,\ F_i x_i = b_i\} \quad \forall\, i = 1, ..., \ell, \tag{29}$$

where given $\chi_i \neq \emptyset$, (25)–(28) becomes:

$$\min \quad \sum_{i=1}^{\ell} c_i x_i \tag{30}$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} D_i x_i = b_0 \tag{31}$$

$$x_i \in \chi_i, \qquad \forall\, i = 1, ..., \ell. \tag{32}$$

From *Resolution Theorem*, we have that any element $x_i \in \chi_i$ can be expressed as

$$x_i = \sum_{q \in Q_i} \lambda_i^q v_i^q + \sum_{k \in K_i} \mu_i^k d_i^k, \tag{33}$$

where $v_i^q\ \forall\ q \in Q_i$ consists of the set of extreme points in χ_i and $d_i^k\ \forall\ k \in K_i$ is the set of extreme rays in χ_i. Also, note that $\lambda_i^q \geq 0$, $\mu_i^k \geq 0$, and $\sum_{q \in Q_i} \lambda_i^q = 1\ \forall\ i = 1, ..., \ell$.

Therefore, using the form presented in (33) the problem in (30)–(32) is equivalent to

$$\min \quad \sum_{i=1}^{\ell} \left(\sum_{q \in Q_i} c_i \lambda_i^q v_i^q + \sum_{k \in K_i} c_i \mu_i^k d_i^k \right) \tag{34}$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell} \left(\sum_{q \in Q_i} D_i \lambda_i^q v_i^q + \sum_{k \in K_i} D_i \mu_i^k d_i^k \right) = b_0 \tag{35}$$

$$\sum_{q \in Q_i} \lambda_i^q = 1 \qquad \forall\, i = 1, ..., \ell, \tag{36}$$

$$\lambda_i^q \geq 0 \qquad \forall\, i = 1, ..., \ell,\ q \in Q_i, \tag{37}$$

$$\mu_i^k \geq 0 \qquad \forall\, i = 1, ..., \ell,\ k \in K_i. \tag{38}$$

In (34)–(38) above, the right-hand-side vector is now $[b_0, 1, \cdots, 1]^\top$. The solution method involves satisfying (35) in a similar method to what was used for the reduced cost vector in Sections 3 and 4. One difference is that the algorithm involves extreme points v_i^q; however, Phase I of the Two-Phase method can be used to find an initial solution to (34)–(38), if necessary. For more information on the functionality of the Dantzig-Wolfe algorithm the reader may refer to [1, 2, 5].

6 Route Balancing

In this section, we present an example of a liner programming formulation to balance the routes for road crews used by street sweepers. Several counties across California have begun to switch from diesel-powered street sweepers to CNG (Compressed Natural Gas) street sweepers in order to comply with Federal and State air quality regulations. However, due to a number reasons, which include slow refueling time at CNG fueling stations and some design characteristics of the CNG sweepers, the productivity of the sweeping operations have decreased with these new sweepers.

The goal of the study is to identify new operating policies to improve the productivity of their previous level when the diesel-powered street sweepers were being used. From a recent

analysis of the assigned road miles, it was evident that the workload varied significantly from crew to crew. Hence, a better balance of the assigned miles-to-crews could significantly improve the productivity of the crews. In this section, we present a mathematical model that optimizes the assignment of road miles to crews.

Input Parameters:

Let,

n:	be the total number of road crews;
$\Phi = \{1, 2, ..., n\}$:	be the set of possible road crews;
R_{ij}:	be the maximum possible road miles that crew i could shift to crew j, $\forall\ (i,j) \in \Phi$ and $R_{ii} = 0$;
A_i:	be the original assigned miles of crew $i \in \Phi$;
U:	be the average miles of the region.

Decision Variables:

Let,

x_{ij}:	be the road miles that crew i shifts to crew j, $\forall\ (i,j) \in \Phi$ and $x_{ii} = 0$;
z_i:	be the deviation between the workload of crew $i \in \Phi$ and the average U.

Using the variable and parameter definitions above, the road crew LP is as follows:

$$\min \quad \sum_{i=1}^{n} z_i \tag{39}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ji} + A_i - \sum_{j=1}^{n} x_{ij} - U \leq z_i \qquad \forall\ i \in \Phi, \tag{40}$$

$$\sum_{j=1}^{n} x_{ji} - A_i - \sum_{j=1}^{n} x_{ij} + U \leq z_i \qquad \forall\ i \in \Phi, \tag{41}$$

$$x_{ij} \leq R_{ij} \qquad \forall\ i, j \in \Phi, \tag{42}$$

$$x_{ij} \geq 0 \qquad \forall\ i, j \in \Phi. \tag{43}$$

The objective in (39)–(43) is to minimize the deviation between the assigned miles of each crew and the average miles of the region, which is

$$|\sum_{j=1}^{n} x_{ji} - A_i - \sum_{j=1}^{n} x_{ij} + U| \quad \forall\, i \in \Phi. \tag{44}$$

Here, variable z_i is used to represent the deviation and linear transformations represented by constraints (40) and (41). The objective function is simply the sum of the deviation of all crews. The smaller the objective value, the more balanced the routes are and an objective value of zero means that each route in the region has exactly the same number of miles. The third constraint (42) limits the miles that crew i can shift to another crew. The data establishment of R_{ij} is based on the physical connection between two adjacent crews, since we only allow the reassignment of miles if the two crews are next to each other on the freeway. The last constraint (43) is simply a sign restriction on the non-negativity property of variable x_{ij}. The non-negativity property of variable z_i has already been ensured by the first and second constraints.

Based on the LP in (39)–(43), we constructed an example that consisted of 37 crews ($n = 37$) and had 31.25 miles average for the region ($U = 31.25$ miles). The resulting linear programming model contained 1406 variables and 2812 constraints. Thus, (39)–(43) may be solved using the column generation method or cutting planes method described in Sections 3 and 4. Table 1 lists the assigned miles to each crew for the current and optimized solutions, which was solved using CPLEX 9.1. The optimal solution is 273 miles, which is a 29.39% improvement over the current actual practice of 386.64 miles.

Crew	Current Solution (miles)	Optimized Solution (miles)	Crew	Current Solution (miles)	Optimized Solution (miles)
1	20.90	31.25	20	19.36	21.40
2	30.65	29.01	21	39.48	33.05
3	39.76	31.25	22	30.16	31.25
4	10.30	10.13	23	40.32	31.25
5	28.57	31.25	24	21.77	11.71
6	30.90	47.05	25	28.79	31.25
7	44.29	77.95	26	20.49	31.25
8	79.02	31.25	27	20.23	28.13
9	38.29	31.25	28	25.84	31.25
10	20.84	23.13	29	35.99	31.25
11	41.03	34.03	30	47.22	38.65
12	12.64	19.64	31	27.36	33.55
13	9.2	15.11	32	31.67	31.25
14	22.96	18.13	33	44.11	38.34
15	18.98	31.25	34	54.2	77.04
16	28.15	25.59	35	57.39	31.25
17	20.21	23.64	36	38.04	38.04
18	29.45	31.25	37	27.95	31.25
19	19.63	12.82			

Table 1: Crew road miles.

References

[1] Bazaraa, M.S., Jarvis, J.J. and Serali, H.D. (2005): *Linear Programming and Network Flows*, John wiley and Sons, Hoboken, NJ.

[2] Bertsimas, D. and Tsitsiklis, J.N. (1997): *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA.

[3] Leachman, R.C. (1993): *Modeling Techniques for Automated Production Planning in the Semiconductor Industry, in Optimization in Industry*, T.A. Ciriani and R.C. Leachman (eds.), John Wiley and Sons, I-30, NY.

[4] Vanderbei, R.J. (2001): *Linear Programming: Foundations and Extensions*, Springer, New York, NY.

[5] Winston, W.L. (2004): *Operations Research: Applications and Algorithms*, Thomson Brooks/Cole, Toronto, ON.

[6] Wolsey, L.A. (1998): *Integer Programming*, John Wiley and Sons Inc., Toronto, ON.