SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK

TECHNICAL REPORT NO. 399

December 1978

OPTIMAL SET PARTITIONING, MATCHINGS AND
LAGRANGIAN DUALITY*

by

George L. Nemhauser and Glenn M. Weber**

* This work has been supported by National Science Foundation Grant
ENG75-00568 to Cornell University.

** Christopher Newport College, Newport News, Virginia.

Optimal Set Partitioning, Matchings and Lagrangian Duality

Abstract

We formulate the set partitioning problem as a matching problem with simple side constraints. As a result we obtain a Lagrangian relaxation of the set partitioning problem in which the primal problem is a matching problem. To solve the Lagrangian dual we must solve a sequence of matching problems each with different edge-weights. We use the cyclic coordinate method to iterate the multipliers, which implies that successive matching problems differ in only two edge-weights. This enables us to use sensitivity analysis to modify one optimal matching to obtain the next one. We give theoretical and empirical comparisons of these dual bounds with the conventional linear programming ones.

1. Introduction

We consider the set partitioning problem

$$\begin{aligned} \max \quad & \sum_{j=1}^n d_j y_j \\ \sum_{j=1}^n a_{ij} y_j &= 1, \quad i = 1, \dots, m \\ y_j &\in \{0,1\}, \quad j = 1, \dots, n \end{aligned} \tag{SP}$$

where d_j is an arbitrary real number for all j and $a_{ij} \in \{0,1\}$ for all i and j . Balas and Padberg [1] give a survey of applications and methods for solving the set partitioning problem. Except for algorithms

developed for small size problems, most algorithms for solving the set partitioning problem use linear programming relaxations. However, for large size problems, because of degeneracy, the linear programs obtained by replacing the binary restriction on each y_j in (SP) by $y_j \geq 0$ often are difficult to solve (Marsten [12]). As a result, the typically large size (and sparse) set partitioning problem sometimes cannot be solved.

We consider a different relaxation that uses matchings on graphs and Lagrangian duality. This is accomplished by reformulating the set partitioning problem as a (weighted, perfect) matching problem, a version of (SP) in which $\sum_{i=1}^n a_{ij} = 2$ for $j = 1, \dots, n$, with simple side constraints. The side constraints are incorporated into the objective function in a Lagrangian fashion, resulting in the primal Lagrangian matching relaxation. The matching problem is one of the few tractable combinatorial problems, and thus is an attractive relaxation for large set partitioning problems.

2. Lagrangian Matching Relaxation

In (SP) let $a_j = (a_{1j}, \dots, a_{mj})$ and suppose that for all j , $\sum_{i=1}^m a_{ij} = 2K_j$, for some integer K_j . This places no limitation on the generality of (SP) since if $\sum_{i=1}^m a_{ij} = 2K_j - 1$ then a new constraint $y_j \leq 1$ can be added to the problem. We replace a_j by the set of columns $\{a_j^k\}_{k=1}^{K_j}$, where $\sum_{k=1}^{K_j} a_j^k = a_j$, $\sum_{i=1}^m a_{ij}^k = 2$, $k = 1, \dots, K_j$

and $a_{ij}^k \in \{0,1\}$, for all i and k . One can form these columns in such a way that both nonzero components of a_j^k precede those of a_j^{k+1} , for all k . Column a_j^k is given an objective function coefficient of $c_j = d_j/K_j$ and is associated with the variable $x_{jk} \in \{0,1\}$. The $\{x_{jk}\}$ are required to satisfy $x_{jk} = x_{j,k+1}$, $k=1, \dots, K_j-1$, for all j . We thus obtain a problem equivalent to (SP) given by

$$\begin{aligned} \max \quad & \sum_{j=1}^n \sum_{k=1}^{K_j} c_j x_{jk} \\ & \sum_{j=1}^n \sum_{k=1}^{K_j} a_{ij}^k x_{jk} = 1, \quad i = 1, \dots, m \\ & x_{jk} - x_{j,k+1} = 0, \quad j = 1, \dots, n \quad \text{and} \quad k = 1, \dots, K_j-1 \\ & x_{jk} \in \{0,1\}, \quad \text{all } j \text{ and } k, \end{aligned} \tag{MS}$$

which is a matching problem with side constraints $x_{jk} = x_{j,k+1}$. Using matrix notation, this problem can be written as

$$\begin{aligned} \max \quad & cx \\ & Mx = 1 \\ & Sx = 0 \\ & x \text{ binary} \end{aligned}$$

where M and S are the coefficient matrices of the matching and side constraints, respectively.

A solution to (MS) yields a solution for (SP) given in

Proposition 1: If $\{x_{jk}^*\}$ is an optimal solution of (MS) then $y_j^* = x_{j1}^*$, $j = 1, \dots, n$ is an optimal solution to (SP).

Let $G(\lambda, x) = (c - \lambda S)x$ where the domain of x is $\{x | Mx = 1 \text{ and } x \text{ binary}\}$ and λ is an unrestricted vector of Lagrange multipliers. The Lagrangian relaxation of (MS) relative to $Sx = 0$ is

$$F(\lambda) = \max_x G(\lambda, x). \quad (LR_\lambda)$$

Without matrix notation, the Lagrangian relaxation can be written as

$$\begin{aligned} \max \quad & \sum_{j=1}^n \sum_{k=1}^{K_j} (c_j - (\lambda_{jk} - \lambda_{j,k-1})) x_{jk} \\ & \sum_{j=1}^n \sum_{k=1}^{K_j} a_{ij}^k x_{jk} = 1, \quad i = 1, \dots, m \\ & x_{jk} \in \{0, 1\}, \quad \text{all } j \text{ and } k \end{aligned}$$

where, for $j = 1, \dots, n$, λ_{j0} and λ_{jK_j} are defined to be zero. (LR_λ) is a (weighted, perfect) matching problem.

Relaxations play a very important role in integer programming algorithms. To be worthwhile, the relaxed problem should be easier to solve than the original one and should also yield a tight bound on the original problem solution. Lagrangian relaxations often fulfill both of these criteria. Since one of the criteria of a good relaxation is the tightness of its bound, the best choice for λ in (LR_λ) is the one that optimizes the Lagrangian dual

$$\min_{\lambda} F(\lambda) \quad (LD)$$

where $\lambda = (\lambda_1, \dots, \lambda_n)$ and $\lambda_j = (\lambda_{j1}, \dots, \lambda_{j, K_j-1})$, for $j = 1, \dots, n$.

Let $v(P)$ represent the optimal objective function value of any problem (P), and let (SPLP) represent the linear programming relaxation of (SP). Proposition 2 (see Geoffrion [7]) summarizes the relationships between (SP), (SPLP), (MS), (LR_λ) and (LD).

- Proposition 2:
- (a) $v(SP) = v(MS) \leq v(LPSP)$,
 - (b) for all λ , $v(MS) \leq v(LR_\lambda) (=F(\lambda))$,
 - (c) if for a given λ a vector x is optimal in (LR_λ) and $Sx = 0$, then x is an optimal solution of (MS), and
 - (d) $v(LD) \leq v(SPLP)$.

Note that the Lagrangian relaxation using the λ found in (LD) is at least as tight as the linear programming relaxation; this is a consequence of the fact that matrix M is not totally unimodular. Typically, $v(SP) < v(LD) < v(SPLP)$.

3. Optimizing the Lagrangian Dual

Many methods (surveyed in Fisher, Northup and Shapiro [6] and Bazaraa and Goode [2]) have been proposed for solving Lagrangian duals. By far the most widely used is the subgradient optimization method described in Held and Karp [8] and Held, Wolfe and Crowder [9]. Compared to other methods, very little "overhead" is needed and, most importantly, it has proven to be very effective computationally. In subgradient optimization,

a sequence $\{\lambda^i\}$ of multiplier vectors is generated iteratively, using at each iteration the solution $F(\lambda^i)$. Many components of each λ^i change from iteration to iteration, and in the context of solving (LD), new optimal matchings must be solved "from scratch." Although solving a large matching problem is much easier than solving a large linear programming problem, it still can be time consuming.

The (weighted, perfect) matching problem

$$\begin{aligned} \max \quad & cx \\ \text{Mx} \quad &= 1 \\ x \quad &\text{binary} \end{aligned}$$

where each column of M contains exactly two nonzero entries, both equal to one, can be interpreted graphically by letting M be the node-edge incidence matrix of a graph in which each row of M represents a node and each column represents an edge where edge k meets node i if and only if $m_{ik} = 1$, and c_k is the weight assigned to edge k . The problem then is to choose a set of edges, called a feasible matching, so that each node meets exactly one of the edges selected, in such a way that the sum of the weights on the edges chosen is as large as possible. Edmonds [3,4,5] developed an efficient (polynomially bounded) primal-dual algorithm for solving the matching problem and Weber [14] showed how sensitivity analysis can be performed on optimal matchings to get the new optimal solution from the original optimal solution if the weight on an edge is changed. Except for some very simple special cases, the techniques

involve modifying the graph by attaching additional nodes and edges near the edge whose edge-weight is to be altered. Edmonds' algorithm is re-entered with all the needed properties, including complementary slackness, being maintained. The final primal and dual solutions for the modified problem are then "translated back" to yield the optimal matching for the single altered edge-weight problem, in such a way that again, all the needed properties are maintained (and thus, the process can be repeated if other edge-weights are altered). Reoptimizing using these techniques when a single edge-weight is altered is on the order of cardinality (N) more efficient than using Edmonds' algorithm "from scratch," where N equals the number of nodes in the graph.

Because of the special structure present in the side constraints of (MS) in which each variable appears in at most two equations, we choose to attempt to optimize (LD) by using an improved version of the cyclic coordinate method of nonlinear programming. The structure of the S matrix results in each λ_{jk} appearing in the coefficient of at most two variables in $G(\lambda, x)$. This allows the sensitivity analysis techniques to be used to improve significantly the usual cyclic coordinate method. In this method, $F(\lambda)$ is optimized cyclically in each of the coordinate directions. Thus, after initializing λ , we minimize $F(\lambda)$ with respect to $\lambda_{11}, \dots, \lambda_{1, K_1-1}, \dots, \lambda_{n1}, \dots, \lambda_{n, K_n-1}$ in that order, one at a time. This process, which involves $\sum_{j=1}^n (K_j - 1)$ single variable minimizations, is repeated until the objective function stops decreasing. Typically, each one-variable minimization is accomplished by one of the iterative or grid type of procedures used in unconstrained optimization problems.

However, because of the special structure of the problem, Theorem 1 provides a direct formula for each minimization, thus avoiding the time consuming "line searches." The proof of Theorem 1 appears in the Appendix.

Theorem 1: Suppose x^* is the current optimal matching vector, λ^* is the current optimal Lagrange multiplier vector, $\hat{\lambda}$ and $\ddot{\lambda}$ are identical to λ^* except for the λ_{jk} component, and $\hat{\lambda}_{jk} = 1 + |c - \lambda^* S| \cdot 1$ and $\ddot{\lambda}_{jk} = -1 - |c - \lambda^* S| \cdot 1$, $\hat{x}$ maximizes $G(\hat{\lambda}, x)$ and $\ddot{x}$ maximizes $G(\ddot{\lambda}, x)$. An optimal λ_{jk}^{**} that minimizes $F(\lambda)$ with respect to λ_{jk} with all other components of λ fixed at their values in λ^* depends on x^* , $\hat{x}$ or $\ddot{x}$, and λ^* as follows:

Case 1: If $x_{jk}^* = x_{j,k+1}^* = 1$ then $\lambda_{jk}^{**} = \lambda_{jk}^*$.

Case 2: If $x_{jk}^* = x_{j,k+1}^* = 0$ then $\lambda_{jk}^{**} = \lambda_{jk}^*$.

Case 3: If $x_{jk}^* = 1$ and $x_{j,k+1}^* = 0$ then

(a) if $\hat{x}_{jk} = 1$ and $\hat{x}_{j,k+1} = 0$ then
 $\lambda_{jk}^{**} = \infty$ (LD is unbounded and MS is infeasible),

(b) if $\hat{x}_{jk} = 1$ and $\hat{x}_{j,k+1} = 1$ then
 $\lambda_{jk}^{**} = \lambda_{jk}^* + [F(\lambda^*) - F(\hat{\lambda})],$

(c) if $\hat{x}_{jk} = 0$ and $\hat{x}_{j,k+1} = 0$ then
 $\lambda_{jk}^{**} = \lambda_{jk}^* + [F(\lambda^*) - F(\hat{\lambda})],$

(d) if $\hat{x}_{jk} = 0$ and $\hat{x}_{j,k+1} = 1$ then
 $\lambda_{jk}^{**} = \lambda_{jk}^* + (1/2)[(\hat{\lambda}_{jk} - \lambda_{jk}^*) - (F(\hat{\lambda}) - F(\lambda^*))].$

Case 4: If $x_{jk}^* = 0$ and $x_{j,k+1}^* = 1$ then

(a) if $\ddot{x}_{jk} = 0$ and $\ddot{x}_{j,k+1} = 1$ then

$$\lambda_{jk}^{**} = -\infty \quad (\text{LD is unbounded and MS is infeasible}),$$

(b) if $\ddot{x}_{jk} = 0$ and $\ddot{x}_{j,k+1} = 0$ then

$$\lambda_{jk}^{**} = \lambda_{jk}^* - [F(\lambda^*) - F(\ddot{\lambda})],$$

(c) if $\ddot{x}_{jk} = 1$ and $\ddot{x}_{j,k+1} = 1$ then

$$\lambda_{jk}^{**} = \lambda_{jk}^* - [F(\lambda^*) - F(\ddot{\lambda})],$$

(d) if $\ddot{x}_{jk} = 1$ and $\ddot{x}_{j,k+1} = 0$ then

$$\lambda_{jk}^{**} = \lambda_{jk}^* - (1/2)[(\lambda_{jk}^* - \ddot{\lambda}_{jk}) - (F(\ddot{\lambda}) - F(\lambda^*))].$$

Theorem 1 is fairly easy to implement. The only unknown quantities in the formulas for λ_{jk}^{**} are $F(\hat{\lambda})$ and $F(\ddot{\lambda})$ and depending on x^* , at most one of these must be found. Computationally, the task of finding either one of these quantities is quite simple, since it is not necessary to solve a new matching problem "from scratch," but only to use sensitivity analysis techniques to reoptimize the matching with two edge-weights altered. The techniques are applied on those edges, one at a time. After computing λ_{jk}^{**} , the two edge-weights are again altered using λ_{jk}^{**} and a new optimal matching is determined using the sensitivity analysis techniques.

At each step of the cyclic coordinate method a new vector λ is generated, differing from the previous λ by at most one component. Let λ^i represent the i -th such vector.

Theorem 2: Assuming $v(\text{MS})$ exists, the sequence $\{F(\lambda^i)\}$ converges.

Proof: For all i , $F(\lambda^{i+1}) \leq F(\lambda^i)$ and, by Proposition 2, the sequence has a lower bound of $v(\text{MS})$. A bounded, nonincreasing sequence has a limit. $\square$

Let $\bar{F}$ denote the limit of $\{F(\lambda^i)\}$. Zangwill [15] and Luenberger [11] give mild restrictions, including $F(\lambda)$ having continuous first partial derivatives and a unique minimum point along any coordinate direction, that guarantee global convergence of the cyclic coordinate method. Unfortunately, $F(\lambda)$ violates these restrictions. It is not necessarily true that $\bar{F} = v(\text{LD})$. In fact, if $F(\bar{\lambda}) = \bar{F}$ then $\lambda = \bar{\lambda}$ might not even be a local minimum, since it is only a relative minimum with respect to the coordinate directions.

The following is an example in which the sequence $\{F(\lambda^i)\}$ generated by the cyclic coordinate method does not converge to $v(\text{LD})$.

Let 0_5 be the null matrix of order 5 and

$$Z = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Let the A matrix of the set partitioning version of (SP) be the 20×25 matrix

Thus, $v(\text{SP}) = v(\text{LD}) < \bar{F}(\lambda) < v(\text{SPLP})$. Notice that the bound achieved from solving the Lagrangian dual, for which the subgradient method is successful (established empirically), and the bound from the cyclic coordinate method are both superior to the one obtained by using linear programming. It should be pointed out that in the usual implementation of subgradient optimization, global convergence is not guaranteed.

In addition to avoiding "from scratch" solutions to large matching problems, another important reason for choosing the cyclic coordinate method instead of the subgradient method is that subgradient optimization lacks an important property of the easy to perform cyclic coordinate method. In using subgradient optimization, the sequence $\{F(\lambda^i)\}$ is not monotone; it can take quite a few iterations until any progress is made in minimizing $F(\lambda)$. Since in a branch-and-bound method we are more interested in getting a close approximation for $v(\text{LD})$ in a short period of time than we are in solving it exactly, it seems reasonable to choose a method that begins showing progress in minimizing $F(\lambda)$ immediately. Actual computational comparisons of the two methods are given in the next section.

3. Computational Results

The results of the computational experiments performed only at the initial node of a branch-and-bound tree are summarized in three tables. Fourteen problems of varying sizes were run using the cyclic coordinate

method (Table 1) and the subgradient method (Table 2) for optimizing the Lagrangian dual (LD), and using linear programming (Table 3) for solving the continuous relaxation of (SP). Each problem contains exactly four ones per column, and in the tables, each is labeled type S2, R1 or RR. S2 is the example given in Section 2. The other two types have constraint matrices consisting of a randomly generated portion containing $n-m/4$ columns and a set of $m/4$ "dummy" columns to insure feasibility. The i -th such "dummy" column contains ones in rows $4i-3$, $4i-2$, $4i-1$ and $4i$, and zeros elsewhere. The objective function coefficients are zero for these columns. Types R1 and RR differ in the objective function coefficients for the other columns. Problems of type R1 have all the coefficients equal to one, while problems of type RR have randomly generated integer coefficients with values between one and ten.

In Table 1 a distinction is made between a cycle and an iteration. Each time $F(\lambda)$ is minimized with respect to all of $\lambda_{11}, \dots, \lambda_{1,K_1-1}, \dots, \lambda_{n1}, \dots, \lambda_{n,K_n-1}$ in that order, one at a time while the others are fixed, a cycle is completed. However, if when minimizing $F(\lambda)$ with respect to say λ_{jk} we have that $x_{jk} \neq x_{j,k+1}$ then this is considered an iteration. Thus, there are potentially as many as $m/2$ iterations per cycle. Loosely speaking, a cycle in the cyclic coordinate method corresponds to an iteration in the subgradient method.

Very seldom does an algorithm perform uniformly better than another on all problems, and the three methods tested are no exception to this

rule. Each out-performs a competing method on at least one of the fourteen problems tested. However, certain general observations can be made. The cyclic coordinate method performs much slower than anticipated, although it does do better than the subgradient method on problem 11. Not surprisingly, linear programming was highly successful in all randomly generated problems except problem 14, the largest one, in which it was inferior to the other two methods. For this problem, linear programming failed to reach an optimum in one minute, while the other two methods were each able to provide useful information since several matching problems were able to be solved in one minute.

We are not discouraged by the fact that the linear programming method out-performs the cyclic coordinate and subgradient methods on the majority of the test problems. The results of problem 3 indicate that there could be a class of problems in which, regardless of the size, the cyclic coordinate and subgradient methods are superior to linear programming, and perhaps more importantly, problem 14 indicates that perhaps for large problems the methods developed here could be a viable alternative to those algorithms that use linear programming.

Table 1

Cyclic Coordinate Method*

Problem	m x n	Type of Data	Initial Value	Final Value	Sx=0?	Cycle No. at Termination	Iterations	Time(sec.) [†] on IBM 370/168
1	20x20	R1	4.5	3	No	2	13	10
2	20x20	R1	5	3.98	No	4	30	15
3	20x25	S2	4	3 [†]	No	13	67	4.78
4	20x50	R1	5	5	No	5	40	15
5	20x50	R1	5	5	No	2	14	10
6	20x50	RR	40.5	35.00	No	5	32	30
7	40x40	R1	8.5	3.80	No	2	22	20
8	40x40	R1	9.5	8.25	No	1	16	20
9	40x100	R1	10	10	No	1	15	15
10	40x100	R1	10	10	No	1	17	20
11	40x100	RR	84.5	81.06	No	1	13	30
12	60x60	R1	13	9.70	No	2	29	60
13	60x150	R1	15	15	No.	1	18	60
14	100x250	R1	25	25	No	1	5	60

[†] CPU time. Integer values indicate arbitrarily set CPU time limits.

^{††} Converging to within .00001 of 3.

* The program for the matching algorithm is given in [13].

Table 3[†]
Linear Programming

Problem	mxn	Type of Data	Final Value	Optimal?	Binary?	Iterations	Time (sec.) on IBM 370/168
1	20x20	R1	0	Yes	Yes	19	.29
2	20x20	R1	0	Yes	Yes	24	.34
3	20x25	S2	5	Yes	No	20	.31
4	20x50	R1	5	Yes	No	42	.58
5	20x50	R1	5	Yes	No	78	.80
6	20x50	RR	29.90	Yes	No	45	.73
7	40x40	R1	0	Yes	Yes	54	.65
8	40x40	R1	0	Yes	Yes	57	.74
9	40x100	R1	10	Yes	No	308	5.18
10	40x100	R1	10	Yes	No	350	6.18
11	40x100	RR	71.02	Yes	No	156	2.76
12	60x60	R1	0	Yes	Yes	93	1.23
13	60x150	R1	15	Yes	No	516	13.44
14	100x250	R1	<25.44	No	—	1209	60

[†]FORTTRAN code given in Land and Powell [10].

Appendix

Proof of Theorem 1: The proof of Case 2 parallels Case 1 and Case 4 parallels Case 3; thus the proofs of only Cases 1 and 3 are given.

Throughout the proof we use the fact that a change in λ_{jk}^* to $\tilde{\lambda}_{jk}$ changes the objective function coefficients of x_{jk} and $x_{j,k+1}$ in (LR_λ) by $\lambda_{jk}^* - \tilde{\lambda}_{jk}$ and $\tilde{\lambda}_{jk} - \lambda_{jk}^*$, respectively.

Case 1: By definition $F(\lambda^{**}) = \max_x G(\lambda^{**}, x) \geq G(\lambda^{**}, x^*)$. Since $x_{jk}^* = x_{j,k+1}^* = 1$, for any λ identical to λ^* except for the λ_{jk} component, $G(\lambda, x^*) = G(\lambda^*, x^*) = F(\lambda^*)$. In particular, $G(\lambda^{**}, x^*) = F(\lambda^*)$ so that $F(\lambda^{**}) \geq F(\lambda^*)$. Now $F(\lambda^{**}) = \min_\lambda \{F(\lambda) : \lambda = \lambda^* \text{ except for the } \lambda_{jk} \text{ component}\} \leq F(\lambda^*)$. Hence $F(\lambda^{**}) = F(\lambda^*)$ and $\lambda^{**} = \lambda^*$.

Case 3: Let $\lambda = \lambda^*$ and consider continuously increasing λ_{jk} from λ_{jk}^* and altering the matching x^* only if $G(\lambda, x)$ would increase by doing so. Let $\bar{\lambda}_{jk}$ be the value of λ_{jk} when x_{jk} first becomes 0, if such a λ_{jk} exists, otherwise set $\bar{\lambda}_{jk} = \infty$. Let $\bar{\bar{\lambda}}_{jk}$ be the value of λ_{jk} when $x_{j,k+1}$ first becomes 1, if such a λ_{jk} exists, otherwise set $\bar{\bar{\lambda}}_{jk} = \infty$. Let $a = \min(\bar{\lambda}_{jk}, \bar{\bar{\lambda}}_{jk})$ and $b = \max(\bar{\lambda}_{jk}, \bar{\bar{\lambda}}_{jk})$. Note that $\hat{\lambda}_{jk}$ has been chosen sufficiently large so that if a or b are finite, they are smaller than $\hat{\lambda}_{jk}$. Thus

$$\hat{x}_{jk} = \begin{cases} 0 & \bar{\lambda}_{jk} \text{ finite} \\ 1 & \text{otherwise} \end{cases} \quad \hat{x}_{j,k+1} = \begin{cases} 0 & \text{otherwise} \\ 1 & \bar{\bar{\lambda}}_{jk} \text{ finite.} \end{cases}$$

As long as $x_{jk} = 1$, there is a unit decrease in the objective function per unit increase in λ_{jk} and when $x_{j,k+1} = 1$ there is a unit increase in the objective function per unit increase in λ_{jk} . Thus,

$$\partial F(\lambda)/\partial \lambda_{jk} = \begin{cases} -1, & \text{if } \lambda_{jk}^* \leq \lambda_{jk} < a \\ 0, & \text{if } a < \lambda_{jk} < b \\ 1, & \text{if } b < \lambda_{jk} \leq \hat{\lambda}_{jk}. \end{cases} \quad (1)$$

In Case 3(a), we have $a = \infty$ so that $F(\lambda)$ decreases monotonically with λ_{jk} and the dual is unbounded ($\lambda_{jk}^{**} = \infty$).

In Cases 3(b,c) a is finite and $b = \infty$. We can set $\lambda_{jk}^{**} = a$. From (1), $F(\hat{\lambda}) - F(\lambda^*) = -(a - \lambda_{jk}^*)$ so that

$$\lambda_{jk}^{**} = a = \lambda_{jk}^* + F(\lambda^*) - F(\hat{\lambda}).$$

In Case 3(d) a and b are finite. We can set $\lambda_{jk}^{**} = (a+b)/2$. From (1), $F(\hat{\lambda}) - F(\lambda^*) = -(a - \lambda_{jk}^*) + (\hat{\lambda}_{jk} - b)$ so that

$$\lambda_{jk}^{**} = (a+b)/2 = (\lambda_{jk}^* + \hat{\lambda}_{jk} + F(\lambda^*) - F(\hat{\lambda}))/2. \quad \square$$

Acknowledgment

We would like to thank Jack Edmonds for many helpful suggestions, particularly with regard to sensitivity analysis of the optimal matchings.