Mathematical Programming 40 (1988) 165-181
North-Holland

LAYERING STRATEGIES FOR CREATING EXPLOITABLE STRUCTURE IN LINEAR AND INTEGER PROGRAMS

Fred GLOVER

School of Business, University of Colorado, Boulder, CO 80309-0419, USA

Darwin KLINGMAN

Department of General Business, Graduate School of Business, University of Texas at Austin, Austin, TX 78712, USA

Received 30 August 1985
Revised manuscript received 22 June 1987

The strategy of subdividing optimization problems into layers by splitting variables into multiple copies has proved useful as a method for inducing exploitable structure in a variety of applications, particularly those involving embedded pure and generalized networks. A framework is proposed in this paper which leads to new relaxation and restriction methods for linear and integer programming based on our extension of this strategy. This framework underscores the use of constructions that lead to stronger relaxations and more flexible strategies than previous applications. Our results establish the equivalence of all layered Lagrangeans formed by parameterizing the equal value requirement of copied variables for different choices of the principal layers. It is further shown that these Lagrangeans dominate traditional Lagrangeans based on incorporating non-principal layers into the objective function. In addition a means for exploiting the layered Lagrangeans is provided by generating subgradients based on a simple averaging calculation. Finally, we show how this new layering strategy can be augmented by an integrated relaxation/restriction procedure, and indicate variations that can be employed to particular advantage in a parallel processing environment. Preliminary computational results on fifteen real world zero-one personnel assignment problems, comparing two layering approaches with five procedures previously found best for those problems, are encouraging. One of the layering strategies tested dominated all non-layering procedures in terms of both quality and solution time.

Key words: Relaxation, network, integer programming, linear programming.

1. Introduction

The strategy of subdividing optimization problems into layers of constraints that replicate the original constraints, with a different copy of some subset of the original variables attached to each layer, has proved valuable in a variety of applications as a means of inducing exploitable structure. First proposed as a means of creating pure and generalized network structure for netforms [17, 18, 19, 22], the layering strategy, as elaborated in succeeding sections, is not confined in principle to this setting, but continues to find its chief practical application in the domain of

This research was supported in part by the Office of Naval Research Contract N00014-78-C-0222 with the Center for Business Decision Analysis and by the US Department of Agriculture Contract 51-3142-4-020 with Management Science Software Systems.

network-related formulations [7, 16, 20, 31, 32]. Complementing this strategy, interest has also emerged in identifying pre-existing embedded pure and generalized network structure or its equivalent [2, 3, 4, 5, 12, 33, 34]. Attention has also been given to developing special methods for networks with "equal flow" side constraints [1, 6, 9, 10, 29, 30], motivated in part by the fact that such constraints arise as a consequence of the layering strategy. Further motivation, both for such methods and the formulations to which they are applied, derives from the recognition [1, 7, 9, 17, 22, 30] that all linear and linear integer programs can be modeled as generalized networks by compelling subsets of arcs to have equal flows. It has also been demonstrated that zero-one integer programs are equivalent to zero-one generalized networks without separately imposing such equal flow conditions [22]. Variations that do not break all variables into network variables, but that maintain a small number of non-network side variables, have also attracted interest [9, 14].

A useful feature of layering strategies is precisely the fact that they may be viewed as creating "structured layers" which are linked by exceedingly simple conditions—a feature well noted since they were first introduced, and which has motivated our use of the "layering strategy" term. The purposes of this paper are: (1) to provide a general framework for representing layering strategies; (2) to establish theoretical results concerning alternate Lagrangean solution approaches; (3) to specify new solution approaches; and (4) to identify useful attributes of layering strategies in a parallel processing environment. Computational results are then presented on fifteen real world zero-one problems comparing two layering strategies and five non-layering procedures.

2. Formulation and induced structure

Consider the original linear or integer programming problem in the following form:

$$\begin{array}{lll} \text{P:} & \text{Minimize} & cx + dy \\ & \text{subject to} & Cx + Dy \leq b, \\ & & x \in X, \\ & & y \in Y, \end{array}$$

where $x \in X$ and $y \in Y$ may summarize special constraints such as integer restrictions on some of the problem variables. The variables have been partitioned into the vectors x and y to permit the isolation of pre-existing structure in the matrix C; e.g., C may represent the matrix for a pure or generalized network. (C and x may have null dimension; i.e., may not exist.)

We undertake to induce structure in the rest of the problem by partitioning rows of D exhibiting desired structure into matrices $D^1, D^2, \ldots, D^k, \ldots, D^r$ which are mutually exclusive and collectively exhaustive of the rows of D. (The "mutually exclusive" property may be usefully subverted in certain circumstances, noted subsequently, by supposing D already contains duplicate rows.)

One structure of particular interest is the two-multiplier generalized network, or GN_2 structure, in which each column contains at most two nonzero elements (see, e.g., Dantzig [8]). The more common definition of a generalized network problem [13, 25, 30, 35] stipulates that the nonzero entry of each singleton column consists of -1 or $+1$, while the nonzeros of each doubleton column consist of a -1 and an arbitrary positive entry (or in some developments, a $+1$ and an arbitrary negative entry). The "arbitrary" element has been called the *multiplier*, *arc multiplier*, or *column multiplier*. We call this more common generalized network problem the *single multiplier* generalized network problem, or GN_1 problem.

The GN_2 structure, for which specialized labeling methods were first proposed in [21], can be transformed into a GN_1 structure by scaling columns and by complementing certain variables relative to their upper bounds (replacing x_j by $x'_j = U_j - x_j$). However, such devices have apparent computational shortcomings. When dealing with integer conditions, for example, scaling has the drawback that each scaled variable must take on its own set of values. In the case where tight upper bounds are not known, complementing variables relative to "large" upper bounds may introduce numerical difficulties and algorithmic inefficiency. Moreover, both of these types of disadvantages are compounded in the context of layering strategies, where each layer requires its own scaling and complementing operations.

Our motive in emphasizing the GN_2 structure is threefold: (i) it provides greater practical flexibility than pure and single multiplier generalized networks for identifying sets of rows to compose the submatrices $D^1, D^2, \ldots$, etc.; (ii) a highly efficient computer code for GN_2 problems has existed for a number of years and has proved of value in a variety of applications, including those involving embedded network structures of varying levels of generality [11, 16, 20]; (iii) the current widespread focus on identifying pure network and single multiplier generalized network model structures in linear and integer programs may advantageously be expanded, in our opinion, to include GN_2 model structure. A useful step in this latter direction has recently been undertaken in [5].

A simple manifestation of the GN_2 framework arises by letting D^1 consist of any two rows of D, letting D^2 consist of any two remaining rows, and so on. Even simpler is to let each D^k consist of a distinct row of D. We later discuss the generation of surrogate constraints which can be incorporated into such D^k components.

Taking advantage of partitioning

The partitioning of D induces a corresponding partition C and b, and thus P may be written in the form

$$P^*: \quad \text{Minimize} \quad cx + dy$$

$$\text{subject to} \quad c^k x + D^k y \leq b^k, \quad k = 1, \ldots, r,$$

In order to exploit the special structure of D^k in each of the preceding r layers of constraints, r different copies $y^1, y^2, \dots, y^r$ of the vector y are created. Let one of them, y^h, represent the "original" copy, and create the enlarged equivalent representation:

$$
\begin{array}{lll}
P_h & \text{Minimize} & cx + dy^h \\
& \text{subject to} & C^k x + D^k y^k \leqslant b^k, \quad k = \qquad , r, \\
& & x \in X, \\
& & y^k \in Y, \quad k = 1, \dots, r, \\
& & \qquad\qquad \text{all } k \neq h.
\end{array}
$$

The constraints $C^k x + D^k y^k \leqslant b^k$ not only individually but collectively exhibit the desired exploitable structure, since each D^k is associated with a different set of variables. We call P_h the *layered representation of P.*

Some of the columns of a particular D^k may correspond to the zero vector. Clearly these columns and the associated variables may be discarded from the associated layer. We therefore suppose in practice that the system is represented by removing unnecessary variables and disregarding irrelevant components of the constraint $y^k = y^h$.

3. Exploiting the layered representation by relaxation

By associating Lagrangian multiplier vectors, u^k, with $y^k = y^h$ in P_h, the following Lagrangean problem is created:

$$
\begin{array}{lll}
P_h(u) & \text{Minimize} & cx + dy^h + \sum\limits_{k \neq h} u^k (y^k - y^h) \\
& \text{subject to} & C^k x + D^k y^k \leqslant b^k, \quad k = 1, \dots, r, \\
& & x \in X_0, \\
& & y^k \in Y_0, \quad k = 1, \quad , r,
\end{array}
$$

where X_0 and Y_0 are supersets of X and Y (obtained, for example, by discarding integer restrictions), and $u = (u^1, u^2, \dots, u^r)$. It might at first be suspected that $P_h(u)$ can be strengthened by incorporating additional equalities into its objective function, as motivated by the observation that we know not only $y^k = y^h$ for $k \neq h$, but also $y^p = y^q$ for all p and q, and the bulk of these latter equalities are missing from the objective of $P_h(u)$. Reflection shows, however, that including the larger set of equalities does not strengthen $P_h(u)$, since equivalent systems of linear simultaneous equations have the same set of linear combinations.

For comparative purposes, we also create a Lagrangean from the precursor P^* of P_h, i.e., the representation specified before the y vector was split into multiple copies. Since the Lagrangean multiplier vectors u^k apply only to $k \neq h$ in $P_h(u)$, the comparative Lagrangean takes only the inequalities $C^k x + D^k y \leq b^k$ for $k \neq h$ into the objective function. This Lagrangian is expressed as follows:

$$\begin{aligned} Q_h(v): \quad & \text{Minimize} \quad cx + dy + \sum_{k \neq h} v^k (C^k x + D^k y - b^k) \\ & \text{subject to} \quad C^h x + D^h y \leq b^h, \\ & \qquad\qquad\quad x \in X_0, \\ & \qquad\qquad\quad y \in Y_0, \end{aligned}$$

where $v = (v^1, v^2, \ldots, v^r) \geq 0$.

We call the constraint layer associated with the special index h the principal layer in $P_h(u)$ and $Q_h(v)$. While it is clear the P_h is the same problem for all h (i.e., the choice of a principal layer is irrelevant to this problem), it is evident that this is not the case for $Q_h(v)$, and the relationship between different instances of $P_h(u)$ as h varies is not obvious. Our first result shows that the choice of the principal layer in defining $P_h(u)$ in fact makes no difference, by establishing a useful additional equivalence. For this we define the problem:

$$\begin{aligned} & \text{Minimize} \quad cx + \sum_{k=1}^{r} w^k y^k \\ & \text{subject to} \quad C^k x + D^k y^k \leq b^k, \quad k = 1, \ldots, r, \\ & \qquad\qquad\quad x \in X_0, \\ & \qquad\qquad\quad y^k \in Y_0, \quad k = 1, \end{aligned}$$

where

$$w = (w^1, w^2, \ldots, w^r) \quad \text{with} \quad \sum_{k=1}^{r} w^k = d.$$

Theorem 1. *$P_h(u)$ is equivalent to $R(w)$ for all h.*

Proof. We first show $P_h(u)$ is equivalent to the problem $R_h(z)$: Minimize $cx + dy^h + \sum_k z^k y^k$ subject to the constraints of $P_h(u)$ where $z = (z^1, z^2, \ldots, z^r)$ satisfies $\sum_k z^k = 0$. For this define $z^k = u^k$ for $k \neq h$, and $z^h = -\sum_{k \neq h} z^k$. Then $\sum_k u^k (y^k - y^h) = \sum_k z^k y^k$. Starting from $\sum_k z^k = 0$ and arguing backward establishes the reverse direction and hence the equivalence of $R_h(z)$ and $P_h(u)$. Finally, defining $w^k = d + z^k$ (and $z^k = w^k - d$) and $w^k = z^k$ for $k \neq h$ establishes similarly the equivalence of $R_h(z)$ and $R(w)$.

Since in general the standard Lagrangeans $Q_h(v)$ are different for each h, and do not involve splitting the variables into copies, it is natural to ask whether at least one of them may provide a stronger relaxation than $R(w)$. The answer to this question follows:

Theorem 2. *$Q_h(v)$ is never stronger than $R(w)$ for any h.*

Proof. We show that $Q_h(v)$ cannot be stronger than $P_h(u)$ by choosing $u^k = -v^kD^k$. Then the objective for $P_h(u)$ becomes

$$\text{Minimize} \quad cx + dy^h + \sum_{k\neq h} (v^kD^ky^h - v^kD^ky^k).$$

From $C^kx + D^ky^k \leqslant b^k$ with $v^k \geqslant 0$ we obtain $v^kD^ky^k \leqslant v^kb^k - v^kC^kx$ and hence the optimized value of the preceding objective is at least as large as that of the objective

$$\text{Minimize} \quad cx + dy^h + \sum_{k\neq h} v^k(C^kx + D^ky^h - b^k).$$

Since we may readily denote y^h by y in $P_h(u)$, the stated conclusion holds.

The fundamental observation of the preceding theorem is independently established by M. Guignard-Spielberg in [23a] (see also [23b]) for the case where C is null and $r=2$. The theorem motivates the creation of strong composite surrogate/Lagrangean relaxations by the following device. Suppose we incorporate among the inequalities for generating a surrogate constraint the inequality

$$cx + dy \leqslant e$$

where e is a scalar selected to bound the objective function (or generated by a branch and bound procedure). Further suppose $x \in X$ implies $x \geqslant 0$, and we also allow upper bounds on components of x to be incorporated among the inequalities for generating a surrogate constraint. Then by the "constrained" surrogate constraint ideas of [15], it may be possible to generate strong surrogate constraints of the form

$$c_0x + d_0y \leqslant e_0 \quad \text{where } c_0 \geqslant 0.$$

If such a surrogate constraint does not exist, the original problem has an unbounded optimum. Otherwise, the foregoing inequality implies

$$d_0y \leqslant e_0$$

and this latter surrogate constraint can be added to the system $Cx + Dy \leqslant b$ without modifying the structure of C, since the added rows of C are all 0. (If $x \in X$ does not imply $x \geqslant 0$, we may incorporate any implied lower bounds on components of x into the original inequalities and require $c_0 = 0$.) In case C and X are null, either by necessity or choice, then of course the requirement $c_0 \geqslant 0$ is trivially met. Our emphasis on creating layers with GN_2 structure permits the incorporation of a pair of surrogate constraints into a single layer, if desired, thus producing a particularly strong composite relaxation.

We next seek a convenient and intuitively appealing means to generate subgradients for $R(w)$ to be used in a subgradient search process [24]. In particular, given a vector $\bar{w}$ which defines a current relaxation $R(\bar{w})$, we seek a subgradient vector g to obtain an improved vector w by reference to the equation

$$w = \bar{w} + sg$$

where s is a nonnegative scalar representing the step size for the subgradient search. (Here w is partitioned as $(w^1, w^2, \ldots, w^r)$ and $g = (g^1, g^2, \ldots, g^r)$ is partitioned similarly.)

Theorem 3. *Let $\bar{x}, \bar{y}^k$, $k = 1, \ldots, r$, be an optimal solution to $R(\bar{w})$, and let a denote the average of the $\bar{y}^k$ vectors*:

$$a = \left(\sum_{k=1}^{r} \bar{y}^k\right) \Big/ r.$$

Then the vector g is a subgradient for $R(w)$ at $\bar{w}$, when

$$g^k = 2(\bar{y}^k - a), \quad k = 1, 2, \ldots, r.$$

Proof. First we note that each problem $P_h(u)$ gives rise to a different subgradient $f(h)$, by setting $f(h)^k = \bar{y}^k - \bar{y}^h$ for $k \neq h$. For $R(w)$, $f(h)$ translates into the subgradient $g(h)$ given by $g(h)^k = \bar{y}^k - \bar{y}^h$ for $k \neq h$ and

$$g(h)^h = -\sum_{k \neq h} (\bar{y}^k - \bar{y}^h) = -\sum_{k=1}^{r} (\bar{y}^k - \bar{y}^h).$$

Although valid, the preceding subgradient $g(h)$ is "lopsided" relative to h. Every convex combination of such subgradients also qualifies as a subgradient, and we choose an equal weighting to yield

$$g^k = \frac{1}{r}\left[\sum_{h=1}^{r} g(h)^k\right] = \frac{1}{r}\left[\sum_{h=1}^{r} (\bar{y}^k - \bar{y}^h) - \sum_{k=1}^{r} (\bar{y}^h - \bar{y}^k)\right]$$
$$= 2(\bar{y}^k - a) \quad \text{for each } k.$$

The subgradient g identified in Theorem 3 assures $\sum_k w^k = d$ as required, upon determining w from $w = \bar{w} + sg$, given $\sum_k \bar{w}^k = 0$.

4. Incorporating problem restriction

To compensate for the absence of the equations $y^k = y^h$ in the relaxation $R(w)$, we propose augmenting $R(w)$ to include simple parameterized bounds on the components of each y^k, thereby obtaining the problem

$R(w, L, U)$:

$$\begin{aligned}
\text{Minimize}\quad & cx + \sum_k w^k y^k \\
\text{subject to}\quad & c^k x + D^k y^k \leq b^k, \ k = 1, \ldots, r, \\
& L \leq y^k \leq U, \ k = 1, \ldots, r, \\
& x \in X_0, \\
& y^k \in Y_0, \ k = 1, \ldots, r,
\end{aligned}$$

where $\sum_k w^k = d$. In this combined relaxation/restriction of P, the goal on the restriction side is to gradually modify components of L and U to bring these vectors closer together. We suggest a heuristic approach for doing this, with the aim of generating good candidate solutions for P (as where, for example, we seek an advanced start for linear programming or an improved incumbent for integer programming).

Relaxation/Restriction method

0. Begin with $\bar{w} = 0$ and choose $\bar{L}$ and $\bar{U}$ so that the inequalities $\bar{L} \leq y^k \leq \bar{U}$ are redundant. Also assign large values to two scalars L_0 and U_0.
1. Solve $R(\bar{w}, \bar{L}, \bar{U})$.
2. Employ the subgradient of Theorem 3 within a standard subgradient optimization procedure to determine a new $\bar{w}$.
3. Redefine $\bar{L}_j := a_j - L_0$ and $\bar{U}_j := a_j + U_0$ for all components a_j of a as given in Theorem 3.
4. Gradually reduce L_0 and U_0 to 0 over successive iterations, and repeat Steps 1-3 alternately or in combination until either all $|\bar{y}_j^k - a_j|$ reach an acceptably small tolerance or $R(\bar{w}, \bar{L}, \bar{U})$ has no feasible solution.

The foregoing procedure is stated without a high degree of specificity in its component steps to provide a general format that allows latitude for alternative implementations. Reasonable variants, for example, could include a non-terminating recovery from a problem with no feasible solution and a more general formula in Step 3 such as

$$\bar{L}_j := t\bar{L}_j + (1-t)a_j - L_0,$$

$$\bar{U}_j := t\bar{U}_j + (1-t)a_j + U_0,$$

where t is a selected parameter between 0 and 1. Good decay rates for L_0 and U_0 and appropriate values of parameters such as t may well provide fertile ground for empirical research.

We further suggest that the reduction of L_0 and U_0 in Step 4 should not be initiated until the subgradient optimization process enters an advanced stage, and that these two terms be allowed to change at different rates. For example, if C is null and $D \leq 0$ then only L_0 should be reduced, while if C is null and $D \geq 0$, then only U_0 should be reduced. The method therefore does not terminate when one of L_0 or U_0 reaches 0, and it is appropriate under these circumstances to employ a refined search scheme that extrapolates where the a_j progression is headed, and substitutes this extrapolated value for a_j in the preceding formulas for $\bar{L}_j$ and $\bar{U}_j$ at each step. Accordingly, the L_0 and U_0 terms may in general be allowed to differ for each j. We emphasize again the value of incorporating a surrogate constraint from other layers into the layer currently under consideration.

5. Preliminary computational analysis

In this section, we provide a preliminary comparison of two layering relaxation strategies with five other relaxation strategies on the class of multi-criteria personnel planning problems called the extended goal programming (EGP) manpower planning model in [18b]. To state the EGP model and to understand the strategies tested, it is convenient to begin with a statement of the following quasi-assignment model.

$$\text{Minimize} \quad \sum_{(i,j)\in A'} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{\{j|(i,j)\in A'\}} x_{ij} = 1, \quad i\in M, \tag{2}$$

$$\sum_{j\in N'} x_{m+1,j} = n, \tag{3}$$

$$\sum_{\{i|(i,j)\in A'\}} x_{ij} = 1, \quad j\in N, \tag{4}$$

$$\sum_{i\in M'} x_{i,n+1} = m, \tag{5}$$

$$x_{ij} \text{ nonnegative and integer } (i,j)\in A', \tag{6}$$

where $M=\{1,2,\ldots,m\}$ is the set of personnel, $N=\{1,2,\ldots,n\}$ is the set of jobs, A is the set of admissible assignments (arcs), x_{ij} equals one (zero) if person i is (is not) assigned to job j, and c_{ij} is the cost of assigning person i to job j. Also, $M'=M\cup\{m+1\}$, $N'=N\cup\{n+1\}$ (where $m+1$ represents a dummy repository of personnel and $n+1$ represents a dummy repository of jobs), and $A'=A\cup\{(m+1,j)\,|\,j\in N\}\cup\{(i,n+1)\,|\,i\in M\}\cup\{m+1,n+1\}$ (which allows dummy personnel to fill any job and dummy jobs to be filled by any person). The c_{ij}, $(i,j)\in A'-A$ can be defined to reflect the cost of an unassigned person or unfilled job or to provide for the maximum assignment of personnel to jobs.

The EGP model is an extension of (1)-(6) which accommodates multiple objectives. Suppose there are three objectives c^k, $k=1,2,3$ and let $\bar{c}^k$ be the optimum objective function values obtained where (1)-(6) is solved using $c=c^k$. (Note that $\bar{c}^k$ exists since (2)-(6) is always feasible.) The EGP model is then stated as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in A'} c^1_{ij}x_{ij} \tag{1'}$$

subject to constraints (2)-(6)

and

$$\left(\sum c^k_{ij}x_{ij} - \bar{c}^k\right)/\bar{c}^k \le q_k\left(\sum c^1_{ij}x_{ij} - \bar{c}^1\right)/\bar{c}^1, \quad k=2,3, \tag{7}$$

where the summations are over (i,j) in A' and the q_k are weights defining the relative importance of the objectives. (The origin and nature of the EGP model is more fully described in [18a].) The EGP model is an instance of the class of integer programming problems with a large embedded network structure.

The computational testing reported in this paper was performed on the set of fifteen EGP problems reported in [18a, 18b]. These problems range in size from 20 constraints and 54 variables to 135 constraints and 2683 variables. Table 1 provides problem specifications.

The test problems represent actual Navy personnel rotation assignment decisions [18a] having three objectives: *c*, the dollar cost of making an assignment; *d*, the desirability to the person of an assignment; and *u*, the utility of the Navy of an assignment. Table 1 gives the optimal values of these objectives considered independently (i.e., $c^1 = c$, $c^2 = d$, and $c^3 = u$). In the present study, as in the earlier studies, the dollar cost objective *c* is considered to be the primary one appearing in the objective function (1′) and in the right-hand side of the two extra non-network constraints (7). To facilitate comparisons of different strategies, we present in Table 2 the best known feasible integer solution for these problems. For each problem the values of the objectives *cx*, *dx* and *ux* are given. Subsequent tables give the same information about integer solutions obtained from the various strategies to be described, solution times in seconds on a Dual CYBER 170/750 and, for the feasible integer solutions, the percent deviation from the best solution in Table 2 in terms of the primary objective *cx*.

In [18a], twelve different strategies for obtaining high quality feasible solutions were examined and compared. The methods tested were: vertex ranking, linear search, restricted basis entry, solution testing of quasi-assignment extreme points, surface optimization, generalized Lagrangean relaxation using optimal dual weights, generalized Lagrangean relaxation using *a priori* intuitive weights, generalized Lagrangean relaxation using weights from subgradient search, surrogate relaxation using optimal dual weights, surrogate relaxation using interval weights, surrogate-Lagrangean relaxation using simple partitioning and interval weights, and surrogate-

Table 1
Problem specifications

Problem	Person	Jobs	Arcs	*c*	*d*	
1	15	12	105	1974	5342	2711
2	21	12	175	216	7312	771
3	9	9	41	1049	908	3014
4	27	9	117	142	1254	2028
5	30	18	280	121	8390	4329
6	9	9	51	2806	3246	3584
7	21	14	205	167	457	4499
8	74	41	1207	11231	23009	14387
9	90	22	697	6938	13298	6893
10	86	47	2549	264	19622	8555
11	19	7	66	1215	3275	1556
12	24	17	51	14221	14904	14767
13	44	33	363	22917	28704	21997
14	18	14	105	5470	8619	7254
15	13	5	35	1948	1357	958

Table 2
Best feasible integer solutions

Problem	*cx*	*dx*	*ux*
	2870	5342	2984
	256	8259	863
	1923	1368	3139
	175	1254	2098
	138	9365	4890
	3085	3246	3763
	1100	457	5544
8	12255	23009	15414
9	7024	13338	6919
10	313	22985	10125
11	1658	3363	1730
12	14918	15106	15228
13	23410	28704	22253
14	5607	8726	7419
15	380	2044	1251

Lagrangean relaxation using strong surrogates and interval weights.

Tables 3-7 presents the computational results on the generalized Lagrangean relaxation using optimal dual weights, generalized Lagrangean relaxation using weights from subgradient search, surrogate relaxation using optimal weights, surrogate relaxation using interval weights, and surrogate-Lagrangean relaxation using

Table 3
Generalized Lagrangean with optimal dual weights

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2074	6241*	3354	inf	33
2	276	9157	787	7.81	4
3	1081	908	3466*	inf	20
4	175	1254	2253	0	14
5	138	9365	4890	0	15
6	2896	3246	3763*	inf	14
7	1209	457	6090	9.91	11
	12339	23547	14664	0.65	40
	7002	13448*	6917	inf	51
	313	22985	10125	0	53
11	1694	3363	1600	2.17	3
12	14918	15106	15228	0	2
	23496	28704	22083	0.37	17
	5638	8726	7419	0.55	7
	2846	2223*	958	inf	21
Total					300

inf: No feasible integer solution was found.
*Extra constraint corresponding to this objective was violated.

Table 4
Generalized Lagrangean using weights from subgradient search

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2947	5342	3096	2.68	38
2	276	9157	787	7.81	6
3	1934	908	3537	0.57	26
4	175	1254	2253	0	19
	138	9365	4890	0	18
	2896	3246	3763*	inf	19
	1209	457	6090	9.91	16
	12339	23547	14664	0.65	44
	7002	13448*	6917	inf	57
10	313	22985	10125	0	
11	1694	3363	1600	2.17	
12	14918	15106	15228	0	6
13	23496	28704	22083	0.37	24
14	5638	8726	7419	0.55	11
15	4323	2223	1251	1037.63	26
Total					360

Table 5
Surrogate relaxation using optimal weights

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2892	6105	2823	0.77	6
2	254	7861	1223*	inf	40
3	1923	1368	3139	0	28
4	175	1254	2253	0	22
5	148	10264*	4645	inf	18
6	2922	3803*	3585	inf	23
7	1100	457	5879	0	38
8	12301	23162	15105	0.34	31
9	7021	13263	7167*	inf	58
10	315	23478*	9963	inf	23
11	1694	3363	1600	2.17	10
12	14918	15106	15228	0	29
13	23293	29603*	22083	inf	11
14	5778	9625*	7254	inf	23
15	2846	1357	1691*	inf	20
Total					380

inf: No feasible integer solution was found.
*Extra constraint corresponding to this objective was violated.

Table 6
Surrogate relaxation using interval weights

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2870	5342	3688	0	10
2	256	8259	863	0	23
3	1925	1368	3139	0.10	27
4	175	1254	2253	0	24
5	138	9365	4890	0	18
6	3085	3246	3763	0	14
7	1100	457	6231	0	36
8	12281	23997	14485	0.18	31
9	7024	13338	6919	0	38
10	313	22985	10125	0	51
11	1658	3363	1730	0	14
12	14918	15106	15228	0	13
13	23496	28704	22083	0.37	22
14	5607	8726	7419	0	9
15	2846	2223*	958	inf	21
Total					351

inf: No feasible integer solution was found.
*Extra constraint corresponding to this objective was violated.

Table 7
Surrogate-Lagrangean using simple partitioning

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2870	5342	3539	0	11
2	256	8259	863	0	23
3	1925	1368	3139	0.10	27
4	175	1254	2253	0	22
5	138	9365	4890	0	19
6	3085	3246	3763	0	18
7	1100	457	6231	0	32
8	12259	23009	15414	0	28
9	7024	13338	6945	0	39
10	320	22492	10203	2.24	4
11	1658	3363	1730	0	18
12	14918	15106	15228	0	17
13	23496	28704	22083	0.37	24
14	5623	8726	7419	0.29	14
15	380	2044	1251	0	18
Total					370

Table 8
Layering with subgradient search

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2947	5342	3096	2.68	
2	276	9157	787	7.81	
	1925	1368	3139	0.10	22
	175	1254	2253	0	16
	138	9365	4890	0	14
	2896	2246	3763*	inf	11
	1100	457	6231	0	15
	12339	23547	14664	0.65	29
	7797	13411	6945	11.1	37
	313	22985	10125	0	42
	1694	3363	1600	2.17	4
	14918	15106	15228	0	11
	23496	28704	22083	0.37	15
	5638	8726	7419	0.55	8
	2846	2223*	958	inf	17
					233

Table 9
Layering with surrogate and subgradient search

Problem	*cx*	*dx*	*ux*	% From best	Time (sec)
1	2947	5342	3688	0	6
2	256	8259	863	0	7
3	1923	1368	3139	0	27
4	175	1254	2253	0	21
5	138	9365	4890	0	16
6	3085	3246	3763	0	14
7	1100	457	6231	0	17
8	12281	23997	14485	0.18	33
9	7024	13338	6945	0	42
10	313	22985	10125	0	46
11	1658	3363	1730	0	8
12	14918	15106	15228	0	13
13	23496	28704	22083	0.37	20
14	5607	8726	7419	0	10
15	380	2044	1251	0	19
Total					299

simple partitioning, respectively. These five methods were found to be superior to the other seven methods tested in [18a, 18b].

To compare the layering approach on the EGP model, we tested two different strategies. In the first strategy, we created two layers, one consisting of the embedded quasi-assignment problem and the other, the two constraints of (7). The quasi-assignment problem was solved with the same network optimizer for all strategies tested and the other layering problem, which is a two node GN_2 problem, was solved using a specially designed code for two node GN_2 problems. In our testing we used a very simple instance of the relaxation/restriction method where $\bar{L}=0$, $\bar{U}=1$, $L_0=U_0=$ infinity, $\bar{L}$ and $\bar{U}$ were never modified, and steps 1-3 were performed a maximum of twenty times. After each iteration of 1-3, the solution was examined for integer feasibility and the best recorded. The results tabulated in Table 8 show that feasible solutions were obtained for 13 of the problems. In terms of both quality and solution times, these solutions are generally much better than those obtained by the generalized Lagrangean relaxation with optimal dual weights and the surrogate relaxation with optimal dual weights. (See Tables 3 and 5.) The solution quality is similar to the generalized Lagrangean where candidate solutions were generated for each set of weights produced during subgradient search (Table 4), but much better in terms of total solution time. The solution quality is somewhat worse than the surrogate relaxation using interval weights (Table 6) and surrogate-Lagrangean relaxation using simple partitioning (Table 7), but the total solution times are much better.

The second layering strategy tested used the dual variables associated with an optimal solution to the two node GN_2 problem on a particular iteration to form a surrogate constraint of the constraints in (7). The surrogate constraint was added to the quasi-assignment problem before solving it on the current iteration. (Again the same software was used to solve this network with one side constraint as employed by the procedures tested in [18a, 18b].) After each iteration of steps 1-3, we transformed a noninteger solution into an integer solution (for the quasi-assignment problem with a surrogate constraint) by executing a single pivot to bring the slack variable for the surrogate constraint into the basis. This solution was examined for feasibility, keeping track of the best integer feasible solution obtained at any iteration. As shown in Table 9 this approach was remarkably robust. For all fifteen problems this approach dominated all non-layering methods in terms of both solution time and solution quality.

6. Conclusions and implications

Our development of the layering framework, the results for exploiting it, and the preliminary computational results, show that layering strategies have a number of appealing attributes. Our orientation strongly motivates a shift in the current dominant focus on pure and single multiplier generalized network structures to a broader focus that encompasses the GN_2 structure. The issue of identifying "good layers," i.e., D^k submatrices with large numbers of rows and small numbers of nonzero columns (to effectively reduce the number of components of the y^k vectors), poses intriguing questions for future investigation.

We suggest the possibility, however, that it may prove worthwhile to employ layering strategies that do not at once seek to generate D^k submatrices with numerous rows but rather initially generate "thin" layers and choose C and x null. By such an approach the problems $R(w)$ and $R(w, L, U)$ completely decompose into small disjoint problems for each layer, inviting the use of parallel processing to solve the problems associated with all layers simultaneously.

In the extreme where each layer consists of just two or three rows of the original problem, a GN_2 procedure can be tailored (as was done in our computational testing) to solve the resulting subproblems with greatly increased efficiency. Still more broadly, a GN_2 problem with a single side constraint (e.g., a surrogate constraint from other layers) can similarly be treated with a highly tailored approach. Very fast initial progress may therefore be sought by reference to such an extreme layering, followed by recourse to alternative layerings when progress slows. In general, we find attractive the notion of employing a strategy that integrates the solution of different layerings. When C and x are null, such a strategy corresponds simply to allowing constraints to be duplicated, thus permitting the layers to be composed in different fashions. In a parallel processing environment, one subset of layers can be solved while the solution of another subset is being transferred to subgradient optimization, where the latter may optionally incorporate attenuated subgradients from solutions to previous subsets.

References

[1] I. Ali, Jeffrey Kennington and Bala Shetty, "The equal flow problem," Technical Report 85-OR-1, Southern Methodist University (Dallas, TX, 1985).

[2] R.E. Bixby, "Recent algorithms for two versions of graph realization and remarks on applications to linear programming," in: W.R. Pulleyblank, ed., *Progress in Combinatorial Optimization* (1984) 39-67.

[3] R.E. Bixby and W.H. Cunningham, "Converting linear programs to network problems," *Mathematics of Operations Research* 5, (1980) 321-357.

[4] R.E. Bixby and Donald K. Wagner, "An almost linear-time algorithm for graph realization," Technical Report 85-2, Dept. of Mathematical Sciences, Rice University (Houston, TX, 1985).

[5] G.G. Brown, R.D. McBride and R.K. Wood, "Extracting embedded generalized networks from linear programming problems," *Mathematical Programming* 32 (1985) 11-31.

[6] C.-H. Chen and M. Enquist, "A primal simplex approach to pure processing networks," Research Report CCS 496, Center for Cybernetic Studies, University of Texas (Austin, TX, 1985).

[7] R. Crum, D. Klingman and L. Tavis, "An operational approach to integrated working capital planning," *Journal of Economics and Business* 35 (1983) 343-378.

[8] George Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ, 1973).

[9] M. Enquist and C.-H. Chen, "Efficient tree handling procedures for allocation/processing networks," Research Report CCS 437, Center for Cybernetic Studies, The University of Texas (Austin, TX, 1982).

[10] M. Enquist and C.-H. Chen, "Computational comparison of two solution procedures for allocation/processing networks," *Mathematical Programming Study* 26 (1986) 218-220.

[11] R. Farina and F. Glover, "Optimal development and allocation of Colorado's energy resources over the coming decade," *Energy Issues in Colorado's Future*, Colorado Energy Research Institute (1980) 161-199.

[12] S. Fujishige, "An efficient *PQ*-graph algorithm for solving the graph realization problem," *Journal of Computer and System Science* 21 (1980) 63-86.

[13] Saul Gass, *Linear Programming: Methods and Applications* (McGraw-Hill, 1975).

[14] F. Glover, "Creating network structure in lp's," in: Greenberg and Maybee, ed., *Computer-Assisted Analysis and Model Simplification* (Academic Press, 1981) 361-368.

[15] F. Glover, "Surrogate constraints," *Operations Research* 16 (1968) 741-749.

[16] F. Glover, R. Glover and F. Martinson, "A netform system for resource planning in the U.S. Bureau of Land Management," *Journal of the Operational Research Society* 35 (1984) 605-616.

[17] F. Glover, J. Hultz and D. Klingman, "Improved computer-based planning techniques, part II," *Interfaces* 9 (1979) 12-20.

[18] F. Glover, J. Hultz, D. Klingman and J. Stutz, "Generalized networks: A fundamental computer-based planning tool," *Management Science* 24 (1978) 1209-1220.

[18a] F. Glover, D. Karney and D. Klingman, "A policy evaluation model and prototype computer assisted policy evaluation system for naval personnel management," Final Technical Report, NPRDC Contract N00123-74-C-2272 (1975).

[18b] F. Glover, D. Karney and D. Klingman, "A study of alternative relaxation approaches for a manpower planning problem," in: Ijiri and Whinston, ed., *Quantitative Planning and Control* (Academic Press Inc., New York, NY, 1979).

[19] F. Glover and D. Klingman, "Network applications in industry and government," *AIIE Transactions* 9 (1977) 363-376.

[20] F. Glover and F. Martinson, "Linear programming/netform model for vegetation allocation," in: W. Lauenroth, G. Skogerboe and M. Flug, ed., *Analysis of Ecological Systems: State of the Art in Ecological Modeling* (Elsevier Scientific Publishing Co., Denmark, 1982).

[21] F. Glover, D. Klingman and J. Stutz, "Extensions of the augmented predecessor method to generalized network problems," *Transportation Science* 7 (1973) 377-384.

[22] F. Glover and J. Mulvey, "Equivalence of the 0-1 integer programming problem to discrete generalized and pure networks," *Operations Research* 28 (1980) 829-835.

[23a] M. Guignard-Spielberg, "Lagrangean decomposition: An improvement over lagrangean and surrogate duals," Technical report #62, Wharton School, University of Pennsylvania (Philadelphia, PA, 1984).

[23b] M. Guignard and S. Kim, "Lagrangean decomposition: A model yielding stronger lagrangean bounds," to appear in *Mathematical Programming*.

[24] M. Held, P. Wolfe and H.D. Crowder, "Validation of subgradient optimization," *Mathematical Programming* 6 (1974) 62-88.

[25] Paul Jensen and J. Wesley Barnes, *Network Flow Programming* (John Wiley and Sons, 1980).

[26] Ellis Johnson, "Flows in networks," in: Elmaghraby and Moder, ed., *Handbook of Operations Research* (Van Nostrand Rheinhold, 1979) 183-206.

[27] J. Koene, "Minimal cost flow in processing networks, a primal approach," Ph.D. thesis, Eindhoven University of Technology (Eindhoven, The Netherlands, 1982).

[28] A. Manne, R. Richels and J. Weynant, "Energy policy modeling: A survey," *Operations Research* 27 (1979) 1-36.

[29] R. McBride, "Solving generalized processing network problems," Working paper, School of Business, University of California (Los Angeles, CA, 1982).

[30] D. Phillips and A. Garcia-Diaz, *Fundamentals of Network Analysis*, Prentice-Hall (New York, NY, 1981).

[31] F. Shepardson and R. Marsten, "A lagrangean relaxation algorithm for the two duty period scheduling problem," *Management Science* 26 (1980) 274-281.

[32] E. Steinberg and H. Napier, "Optimal multi-level lot sizing for requirements planning systems," *Management Science* 26 (1980) 1258-1271.

[33] K. Truemper, "How to detect hidden networks and totally-unimodular subsections of linear programs," TIMS/ORSA Joint National Meeting (April 1983).

[34] D.K. Wagner, "An almost linear-time graph realization algorithm," Ph.D. dissertation, Northwestern University (Evanston, Il, 1983).

[35] Harvey Wagner, *Principles of Operations Research* (Prentice-Hall, New York, NY, 1969).