A Lagrangian Approach to the Pooling Problem

Nilanjan Adhya and Mohit Tawarmalani

Department of Mechanical and Industrial Engineering, University of Illinois at Urbana−Champaign, 1206 West Green Street, Urbana, Illinois 61801

Nikolaos V. Sahinidis*

Department of Chemical Engineering, University of Illinois at Urbana−Champaign, 600 South Mathews Avenue, Urbana, Illinois 61801

Pooling and blending problems occur frequently in the petrochemical industry where crude oils, procured from various sources, are mixed together to manufacture several end-products. Finding optimal solutions to pooling problems requires the solution of nonlinear optimization problems with multiple local minima. We introduce a new Lagrangian relaxation approach for developing lower bounds for the pooling problem. We prove that, for the multiple-quality case, the Lagrangian approach provides tighter lower bounds than the standard linear-programming relaxations used in global optimization algorithms. We present computational results on a set of 13 problems which includes four particularly difficult problems we constructed.

1. Introduction

The pooling problem is a planning problem that arises in blending materials to produce products, an example being the blending of crude or refined petroleum. Pooling occurs whenever streams are mixed together, often in a storage tank, and the resulting mixture is dispatched to several locations. Pooling and blending of raw materials and stored products is an important step in the synthesis of end-products having diverse component quality specifications.

Optimization of gasoline blending is a critical refinery operation. Consider, for example, the case of a large company such as Texaco. According to DeWitt et al.,[8] the use of a nonlinear optimizer to predict output blend qualities given input stock qualities and volumes led to an improvement in the blending procedure over what was being followed earlier. The nonlinear optimizer, which was used to solve pooling and blending problems, led to a saving of about 2.5 cents/gal of gasoline which translated into a saving of more than 30 million dollars annually for Texaco in the 1980s. Texaco's blending system evolved to a decision support system used in all Texaco refineries in the 1990s.[32]

Pooling also occurs in distillation and other separation processes. The mathematics of the pooling problem apply to such processes and their applications, which are numerous. A recent application in New Zealand refineries is reported by Amos et al.[2]

The process of pooling introduces nonlinearities and nonconvexities into optimization models leading to the existence of several locally optimal solutions. Naturally, it takes more effort to solve a problem to global optimality than it takes to find a locally optimal solution and one must often weigh the benefits against the costs. It is apparent though that, given the volumes of sales of petroleum products, the global optimization of the pooling and blending process could lead to substantial savings in cost, resulting in higher margins of profit.

* Address all correspondence to this author. E-mail: nikos@uiuc.edu. Phone: 217-244-1304. Fax: 217-333-5052.

Perhaps the most popular global optimization technique is branch-and-bound.[22] Branch-and-bound is a deterministic global optimization technique which uses controlled enumeration and relaxation to divide the original feasible set into a number of subsets and then derive lower and upper bounds of the objective function over each of these subsets. On the basis of these bounds, some of the subsets are subjected to further refinement, while others are excluded from further consideration based on optimality or feasibility arguments. The tightness of the bounding procedure used is an important factor which determines the efficacy of the branch-and-bound procedure. Consequently, there is a need to develop bounding procedures which would provide tight bounds within the branch-and-bound framework.

Lagrangian relaxation was popularized in the early 1970s by Held and Karp[20,21] in their work on the traveling salesman problem. Lagrangian relaxation is based upon the observation that many problems can be considered to be relatively easy problems made difficult because of the presence of complicating constraints. The Lagrangian approach creates a subproblem, termed the Lagrangian subproblem, wherein the complicating constraints are replaced with a penalty term in the objective function involving the amount of violation of the constraints and their dual variables—also known as Lagrange multipliers. The Lagrangian subproblem is typically easy to solve compared to the original problem and provides a lower bound on the global optimum of the original problem for the case of a minimization problem. This allows the use of a Lagrangian relaxation for generating lower bounds for use in algorithms to solve combinatorial optimization problems.

Lagrangian relaxation has been applied to different classes of problems, both discrete and continuous. Applications of Lagrangian duals for nonconvex optimization have been suggested by Falk,[10] Dür and Horst,[9] and Ben-Tal et al.[7] Properties of Lagrangian relaxations and strategies for generating and updating Lagrange multipliers have been discussed by Bazaraa and Goode,[5] Fisher,[12] and Minoux.[29]

10.1021/ie980666q CCC: $18.00
Published on Web 04/02/1999

Whereas it is well-known that Lagrangian relaxation provides bounds that are not weaker than standard linear-programming (LP) relaxations, it is not known how to construct Lagrangian relaxations that provide bounds that are strictly stronger than LP-based bounds. Depending on what we consider complicating constraints, several alternative lagrangian relaxations can be developed. In general, we would like to choose a relaxation which makes the Lagrangian subproblem easy to solve. However, it is often observed that an easy subproblem leads to lower bounds that are not very strong and consequently one has to solve a more difficult and therefore more expensive subproblem to get tighter bounds on the original problem. The primary purpose of this paper is to introduce a Lagrangian relaxation approach that provides tighter bounds than conventional Lagrangian and linear-programming-based approaches for the case of the pooling problem.

In the case of the pooling problem, the complicating terms are constraints having bilinear terms. Dualizing all constraints leads to a Lagrangian subproblem which consists of optimizing a bilinear objective function over a hypercube. The Lagrangian subproblem obtained from this relaxation is thus a special case of bilinear-programming problems—an important area of research in itself. We provide a method of solving this Lagrangian subproblem by reformulating it as a mixed integer linear program.

Several alternative Lagrangian relaxations could be constructed from the constraint set of the pooling problem. One approach would be the introduction of separability for the bilinear terms which makes the corresponding Lagrangian subproblem separable in each variable and therefore easy to optimize. We prove that this easier approach does not provide bounds that are as tight as our Lagrangian approach.

The paper is organized in the following way. Section 2 describes the pooling problem. Section 3 presents a literature survey on pooling and blending problems and describes some of the techniques commonly used to solve such problems. Section 4 introduces a Lagrangian lower bounding procedure for the pooling problem and provides a method for solving the Lagrangian subproblem to optimality. Section 5 describes several properties of the Lagrangian subproblem and the proposed Lagrangian relaxation. Section 6 compares and contrasts the Lagrangian relaxation procedure with a linear relaxation procedure obtained by using the convex and concave envelopes of the bilinear terms using McCormick estimators. The comparison is made for both the single-quality and the multiple-quality cases of the pooling problem. We prove that the lower bounds obtained by the Lagrangian relaxation procedure are at least as strong as those obtained by the McCormick-estimator-based linear relaxation and they might be strictly stronger. Section 7 provides computational results with 13 test problems, including four hard problems designed in the course of this study. For the latter four examples, the Lagrangian lower bounds are strictly stronger than the linear-programming based bounds. Finally, in section 8 we show through an illustrative example how the Lagrangian procedure can be used within a branch-and-bound framework.

2. Problem Description

Figure 1 shows a small pooling problem taken from Haverly.[18] There is a single pool which receives supplies

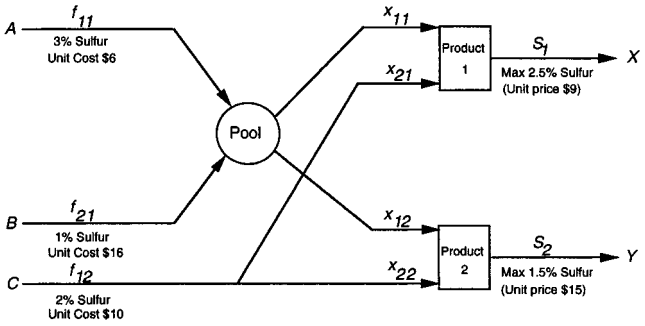

Figure 1. Haverly's pooling problem.

from two different sources of crude oil A and B. Since A and B are different sources, they have different sulfur qualities. A third supply C is not fed into the pool but is directly mixed with the two outflows from the pool. The quality parameters for the streams going into the pool are 3% for A, 1% for B, and 2% for C. The blending of flows from the pool and from the supply stream C produces products X and Y, which have to adhere to sulfur quality specifications of 2.5% and 1.5%, respectively. These restrictions on end-products X and Y are dictated by the consumer. In the context of the petroleum industry, for instance, these restrictions might be the maximum percentage of sulfur in gasoline. The maximum demands for products X and Y are $S_1 = 100$ and $S_2 = 200$, respectively, which also restrict the quantities of end-products produced.

The variables in the above example are the quantities of supplies of A, B, and C, denoted by f_{11}, f_{21}, and f_{12}, respectively, the final quality of sulfur in the pool as a result of mixing of A and B, denoted by q, the magnitudes of flows from the pool products, denoted by x_{11} and x_{12}, respectively, and the amounts of supply C which go to products X and Y, denoted by x_{21} and x_{22}, respectively. The quantities of each end-product and their final qualities can be easily recovered from the values of the above variables. On the basis of the assumption of a linear mixing model, the problem in Figure 1 can then be formulated as

(H)

$$\min\ 6f_{11} + 16f_{21} + 10f_{12} - 9(x_{11} + x_{21}) - 15(x_{12} + x_{22}) \quad (1)$$

$$\text{s.t. } f_{11} + f_{21} - x_{11} - x_{12} = 0 \quad (2)$$

$$f_{12} - x_{21} - x_{22} = 0 \quad (3)$$

$$q(x_{11} + x_{12}) - 3f_{11} - f_{21} = 0 \quad (4)$$

$$qx_{11} + 2x_{21} - 2.5(x_{11} + x_{21}) \le 0 \quad (5)$$

$$qx_{12} + 2x_{22} - 1.5(x_{12} + x_{22}) \le 0 \quad (6)$$

$$x_{11} + x_{21} \le S_1 \quad (7)$$

$$x_{12} + x_{22} \le S_2 \quad (8)$$

(1) represents the difference between the cost of the input streams and the profits from selling the products. Equations 2 and 3 represent mass balances. Equation 4 expresses the pool quality, q, in terms of the input streams and their qualities. Equations 5 and 6 represent the quality restrictions on the products. Finally, (7) and (8) ensure that the flows do not exceed demands.

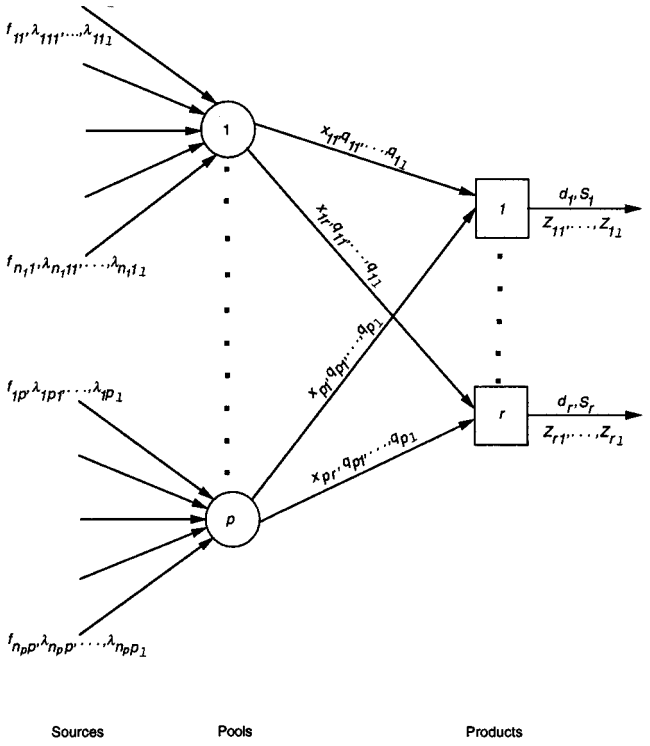

Figure 2. Pooling problem.

Table 1. Pooling Problem Nomenclature

indices	i	streams, $i = 1, ..., n_j$
	j	pools, $j = 1, ..., p$
	k	products, $k = 1, ..., r$
	w	qualities, $w = 1, ..., l$
parameters	c_{ij}	unit cost of the ith stream into pool j
	d_k	unit price of product k
	l	total number of component qualities
	n_j	number of streams entering pool j
	p	total number of pools
	r	total number of end-products
	S_k	demand for product k
	Z_{kw}	wth quality requirement for product k
	λ_{ijw}	wth quality specification of the ith stream into pool j
variables	f_{ij}	flow of ith input stream into pool j
	q_{jw}	wth quality of pool j from pooling of streams
	x_{jk}	total flow from pool j to product k
bounds	f_{ij}^{L}	lower bound on flow for f_{ij}
	f_{ij}^{U}	upper bound on flow for f_{ij}
	q_{jk}^{L}	lower bound on flow for q_{jk}
	q_{jk}^{U}	upper bound on flow for q_{jk}
	x_{jk}^{L}	lower bound on flow for x_{jk}
	x_{jk}^{U}	upper bound on flow for x_{jk}

The bilinear terms in constraints (4)−(6) introduce nonconvexities in the problem leading to the existence of multiple local optima. Problem (1)−(8), for example, has infinite local solutions with an objective function of 0, a local minimum with an objective function of −100, and a single global optimum with a value of −400. Consequently, standard nonlinear-programming techniques may provide grossly suboptimal solutions and we need to look at global optimization techniques for such problems.

In practice, pooling problems become even more complicated because of the presence of a large number of pools and end-products. Often, each stream into a pool can have more than one quality component (e.g., sulfur and phosphorus). The pooling problem then becomes a problem with multiple-component qualities and every end-product has to adhere to quality specifications on each of its several qualities. The existence of multiple pools, products, and qualities leads to the existence of hundreds of bilinear terms even for moderately sized problems and therefore a large number of local optima, thereby increasing the need for a global optimization approach. Figure 2 depicts a general pooling problem with p pools, r products, and l quality parameters. Table 1 describes the nomenclature used throughout this article.

Problem P below provides a mathematical formulation of the general pooling problem described in Figure 2.

(P)

$$\min \sum_{j=1}^{p}\sum_{i=1}^{n_j} c_{ij} f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk}$$

$$\text{s.t.} \sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk} = 0 \qquad j = 1, ..., p \tag{9}$$

$$q_{jw}\sum_{k=1}^{r} x_{jk} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij} = 0 \qquad j = 1, ..., p; \quad w = 1, ..., l \tag{10}$$

$$\sum_{j=1}^{p} x_{jk} - S_k \le 0 \qquad k = 1, ..., r \tag{11}$$

$$\sum_{j=1}^{p} q_{jw} x_{jk} - Z_{kw}\sum_{j=1}^{p} x_{jk} \le 0 \qquad k = 1, ..., r; \quad w = 1, ..., l \tag{12}$$

$$f_{ij}^{L} \le f_{ij} \le f_{ij}^{U} \qquad i = 1, ..., n_j; \quad j = 1, ..., p \tag{13}$$

$$q_{jw}^{L} \le q_{jw} \le q_{jw}^{U} \qquad j = 1, ..., p; \quad w = 1, ..., l \tag{14}$$

$$x_{jk}^{L} \le x_{jk} \le x_{jk}^{U} \qquad j = 1, ..., p; \quad k = 1, ..., r \tag{15}$$

In (P), the objective function represents the difference between the cost of using the input streams and the returns from selling the end-products. Equation 9 represents the mass balances for each pool. Equation 10 expresses the pool quality, q_{jw}, in terms of the qualities, λ_{ijw}, of the input streams. Equation 11 ensures that the flows do not exceed demand. Equation 12 enforces the quality requirements for each of the end-products.

3. Literature Review

Variants of formulation P above have been used by Haverly,[18,19] Lasdon et al.,[23] Floudas and Aggarwal,[13] and Foulds et al.[14] for the case of pooling problems with a single-component quality. A similar formulation for the pooling problem is the p-formulation of Ben-Tal et al.[7] The more general formulation based on total flows and component compositions for general process networks suggested by Quesada and Grossmann[31] is similar to (P), if applied in the context of the pooling problem. The formulations used by Fieldhouse,[11] Main,[26] and Amos et al.[2] are similar to (P). Thus, (P) appears

Table 2. Summary of Work on the Pooling Problem

authors	qualities	optimum	comments
		Recursive Methods	
Haverly[18,19]	single	local	recursion
Baker and Lasdon[4]	single	local	successive linear programming (SLP)
Lasdon et al.[23]	single	local	generalized reduced gradient and SLP
Fieldhouse[11]	single	local	distributed recursion
Main[26]	single	local	stability analysis
		Sensitivity and Feasibility Analysis	
Lodwick[25]	multiple	N/A	interval analysis
Greenberg[17]	multiple	N/A	computational geometry
		Decomposition Methods	
Floudas and Aggarwal[13]	single	local	benders decomposition based
Floudas and co-workers[3,40,41]	multiple	global	GOP algorithm
		Branch-and-Bound Methods	
Foulds et al.[14]	single	global	convex approximations of bilinear terms
Ben-Tal et al.[7]	multiple	global	Lagrangian duality
Quesada and Grossman[31]	single	global	reformulation−linearization

to be the formulation most commonly used for modeling the pooling problem in chemical process industries.

Alternative formulations of the pooling problem include the *q*-formulation suggested by Ben-Tal et al.,[7] which does not explicitly use the pool qualities as variables, but instead uses variables based on the fraction that each input stream contributes to the total input to each pool. This leads to a concise formulation of the dual suggested by the authors. Quesada and Grossmann[31] model the pooling problem as a special case of more general process networks. They propose a formulation based on flows of individual quality components, which produces fewer nonconvexities, but gives rise to a large number of linear equations.

Various solution procedures for the pooling problem have also been reported in the literature. These solution procedures can be classified based on their convergence to either a local or a global optimum. The first algorithm for the pooling problem in the form of a "poor man's NLP" was suggested by Haverly.[18,19] Haverly's approach was based on the idea of using recursion to solve the pooling problem. A recursive approach "guesses" the value of the pool qualities. These guessed values of pool qualities convert (*P*) into a linear program in the flow variables, f_{ij} and x_{jk}. The actual values of the pool qualities can then be calculated from the values of the flow variables that are obtained by solving the linear program. The process continues until the actual values of the qualities are within a range of tolerance from the guessed values. The main drawback in using any form of recursive method for the pooling problem lies in the fact that often the process does not converge to a solution, and when it does, it converges only to a local optimum. Moreover, Main[26] shows that as the number of pools and end-products increases, recursive methods tend to become more unstable, resulting in computational difficulties.

Successive linear-programming (SLP) approaches which solve nonlinear problems as a sequence of linear programs have also been popular. Lasdon et al.[23] describe an algorithm based on SLP procedures. Such an approach has been used at Exxon and is described by Baker and Lasdon.[4] These approaches converge to locally optimal solutions.

Decomposition methods are based on the observation that a difficult problem could be converted to an easy problem by fixing values of certain variables. In the case of the pooling problem, for example, fixing the pool quality variables, q_{jw}, converts (*P*) into a linear program. On the basis of this observation, Floudas and Aggarwal[13] suggest an approach based on decomposing the original pooling problem into a primal problem and a relaxed master problem and iterating between these problems based on a generalized Benders decomposition procedure until appropriate stopping conditions are met. Though their decomposition strategy is successful for the problems suggested by Haverly, in general, it offers no guarantee for global optimality. This variant of the Benders decomposition algorithm may converge to a local minimum, a local maximum, or even a non-KKT (Karush−Kuhn−Tucker) point as shown by Sahinidis and Grossmann.[37]

Visweswaran and Floudas[41] propose a deterministic global optimization (GOP) algorithm for solving certain classes of nonconvex problems including the pooling problem. The algorithm was proven to finitely terminate with an ϵ-global optimum, even though this might require a modification of the original algorithm as noted by Gourdin et al.[16] Using this algorithm, they were able to solve three cases of the Haverly problem. The authors also report solving a single pool, five-product problem, with each stream having two quality components, to global optimality using their algorithm. Subsequently, the authors suggested improvements to enhance the performance of the GOP algorithm,[40] and were able to solve problems with three pools and five products with each stream having two quality components. Large-scale pooling problems, generated randomly, having up to five pools, five products, and 30 qualities, were solved by Androulakis et al.[3] using a distributed implementation of the GOP algorithm.[3]

Branch-and-bound methods for pooling and blending problems have been suggested by many authors. These methods differ in the relaxations used to provide valid lower bounds to the global optimum. The procedure used by Foulds et al.[14] involves replacing the bilinear terms in the pooling problem by their concave and convex envelopes. These envelopes are defined over the hypercube derived from the bounds on the variables in the bilinear terms. The concave and convex envelopes are provided by a set of linear constraints proposed by McCormick.[27,28] The nonlinear pooling problem can then be relaxed to a linear-programming problem, the solution to which provides a lower bound on the global optimal solution. The branch-and-bound procedure proceeds by partitioning the problem and relaxing each partition in the way described in the Introduction. On the basis of the fact that the error introduced by

replacing each bilinear term by its concave or convex envelope tends to zero as the partitions get finer, the algorithm converges to the global optimal solution as has been proved by Al-Khayyal and Falk.[1] Using this approach, Foulds et al.[14] were able to solve single-quality problems, with the largest problem having eight pools and 16 products.

The linear constraints which provide the convex and concave envelopes of the problem at some node of the branch-and-bound tree are not in general valid for other nodes of the tree. Thus, the convex and concave envelopes have to be freshly generated at every node of the branch-and-bound tree. Further, this approach requires four linear constraints to provide the envelopes for each bilinear term in the problem. Therefore, as the number of pools, products, or component qualities increase, the size of the linear program to be solved at each node of the branch-and-bound tree can become quite large.

Ben-Tal et al.[7] describe a lower bounding procedure based on the Lagrangian dual for the *q*-formulation of the pooling problem. They provide a branch-and-bound algorithm which partitions the feasible set of the pooling problem and show that such partitioning of the feasible set can reduce the duality gap between a nonconvex problem and its Lagrangian dual. Dür and Horst[9] show that when the branch-and-bound procedure is applied to partly convex problems like the pooling problem, under certain regularity conditions, branching on the space of nonconvex variables can reduce the duality gap between the primal and the Lagrangian dual to zero.

The nonlinear formulation of general process networks can be based on individual flows of each component or the total flows for each stream as proposed by Quesada and Grossmann.[31] The authors establish a relationship between the total flow formulation and the individual component formulation based on the reformulation and linearization technique of Sherali and Alameddine[39] and devise a reformulated model for general process networks which avoids nonlinear terms in the constraint set. The reformulated model is used to obtain lower bounds on the global optimum within a spatial branch-and-bound algorithm.

Table 9 provides a summary of existing approaches to the pooling problem.

4. Lagrangian Lower Bounding

We introduce a Lagrangian relaxation procedure for the pooling problem. Consider the following Lagrangian relaxation of problem (*P*), denoted by (*LRP*), obtained by dualizing (9)−(12).

(*LRP*)

$$\max \theta(\mathbf{v},\mathbf{v}',\mathbf{u},\mathbf{u}')$$

$$\text{s.t. } u_k \geq 0 \qquad k = 1, ..., r$$

$$u'_{kw} \geq 0 \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$v_j \text{ unrestricted} \qquad j = 1, ..., p$$

$$v'_{jw} \text{ unrestricted} \qquad j = 1, ..., p;\ w = 1, ..., l$$

where the Lagrangian subproblem *LSP* is defined as

(*LSP*)

$$\theta(\mathbf{v},\mathbf{v}',\mathbf{u},\mathbf{u}') = \min \sum_{j=1}^{p}\sum_{i=1}^{n_j} c_{ij}f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk} + \sum_{j=1}^{p} v_j\left(\sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk}\right) + \sum_{j=1}^{p}\sum_{w=1}^{l} v'_{jw}\left(q_{jw}\sum_{k=1}^{r} x_{jk} - \sum_{i=1}^{n_j} \lambda_{ijw} f_{ij}\right) + \sum_{k=1}^{r} u_k\left(\sum_{j=1}^{p} x_{jk} - S_k\right) + \sum_{k=1}^{r}\sum_{w=1}^{l} u'_{kw}\left(\sum_{j=1}^{p} q_{jw}x_{jk} - Z_{kw}\sum_{j=1}^{p} x_{jk}\right)$$

$$\text{s.t. (13)−(15)}$$

(*LRP*) involves a maximization problem over the set of Lagrange multipliers v_j, v'_{jw}, u_k, and u'_{kw} and a minimization problem, denoted by (*LSP*), over the set of variables of the pooling problem f_{ij}, q_{jw}, and x_{jk}. (*LSP*), when solved to optimality for any given set of feasible values of the Lagrange multipliers, provides a lower bound to (*P*). Consequently, we are interested in a procedure to solve (*LSP*) to optimality.

Consider (*LSP*1) below, which is obtained from (*LSP*) by a rearrangement of terms.

(*LSP*1)

$$\min \sum_{j=1}^{p}\sum_{i=1}^{n_j} f_{ij}\left(c_{ij} + v_j - \sum_{w=1}^{l} v'_{jw}\lambda_{ijw}\right) + \sum_{j=1}^{p}\sum_{k=1}^{r} x_{jk}\left(-d_k - v_j - \sum_{w=1}^{l} Z_{kw}u'_{kw} + u_k + \sum_{w=1}^{l}(v'_{jw} + u'_{kw})q_{jw}\right) - \sum_{k=1}^{r} u_k S_k \tag{16}$$

$$\text{s.t. (13)−(15)}$$

Since the Lagrange multipliers v_j, v'_{jw}, u_k, and u'_{kw} in problem (*LSP*1) are known, we can reduce (*LSP*1) to the following by an appropriate definition of constants γ_{ij}, β_{jk}, α_{jkw}, and δ_k.

(*MLSP*)

$$\min \sum_{j}\sum_{i} \gamma_{ij} f_{ij} + \sum_{j}\sum_{k} x_{jk}\left(\beta_{jk} + \sum_{w=1}^{l} \alpha_{jkw} q_{jw}\right) - \sum_{k} \delta_k$$

$$\text{s.t. (13)−(15)}$$

where

$$\gamma_{ij} = c_{ij} + v_j - \sum_{w=1}^{l} v'_{jw}\lambda_{ijw}$$

$$\beta_{jk} = -d_k - v_j - \sum_{w=1}^{l} Z_{kw}u'_{kw} + u_k$$

$$\alpha_{jkw} = v'_{jw} + u'_{kw}$$

$$\delta_k = u_k S_k$$

Problem (*MLSP*) can be decomposed into $p + 1$ subproblems—a single problem, (*F*), for optimizing over

the variables f_{ij}, and p subproblems, ($XQ(j)$), for $j = 1, ..., p$, as follows:

(F)

$$\min \sum_j \sum_i \gamma_{ij} f_{ij}$$

$$\text{s.t. } f_{ij}^{\mathrm{L}} \le f_{ij} \le f_{ij}^{\mathrm{U}} \qquad i = 1, ..., n_j; \quad j = 1, ..., p$$

($XQ(j)$)

$$\min \sum_{k=1}^{r} x_{jk}(\beta_{jk} + \sum_{w=1}^{l} \alpha_{jkw} q_{jw})$$

$$\text{s.t. } x_{jk}^{\mathrm{L}} \le x_{jk} \le x_{jk}^{\mathrm{U}} \qquad k = 1, ..., r$$

$$q_{jw}^{\mathrm{L}} \le q_{jw} \le q_{jw}^{\mathrm{U}} \qquad w = 1, ..., l$$

Proposition 4.1. LSP1 *can be decomposed into* p + *1 separate subproblems, each of which can be solved independently of the others.*

Thus, instead of solving the bigger problem ($LSP1$), we can solve a single subproblem (F), to get the optimal f_{ij}'s, and p subproblems of the type ($XQ(j)$), to get the optimal x_{jk}'s and q_{jw}'s. These results can be combined to obtain the optimal solution for ($LSP1$). The decomposition into p subproblems of the type ($XQ(j)$) for $j = 1, ..., p$ is possible because of the absence of coupling of variables of a certain pool with variables of another pool in any of the constraints. The absence of coupling of variables of one pool with another allows us to solve for variables associated with each pool j as an optimization problem ($XQ(j)$).

The following propositions characterize an optimal solution to problem ($LSP1$).

Proposition 4.2. *There exists an optimal solution to (*XQ(j)*) that has every variable* x_{jk} *and* q_{jw} *at one of its bounds. Similarly, every* f_{ij} *in (*F*) will also be at one of its bounds.*

Proof. Consider any given solution vector (x_{ij}, q_{jw}) to the problem ($XQ(j)$). We can then construct a solution ($\bar{x}_{jk}$, q_{jw}), which is at least as good as (x_{jk}, q_{jw}), where

$$\bar{x}_{jk} = \begin{cases} x_{jk}^{\mathrm{L}} & \text{if } \beta_{jk} + \sum_w \alpha_{jkw} q_{jw} \ge 0 \\ x_{jk}^{\mathrm{U}} & \text{if } \beta_{jk} + \sum_w \alpha_{jkw} q_{jw} < 0 \end{cases} \qquad (17)$$

Thus, there exists an optimal solution where each x_{jk} will be at one of its bounds.

Replacing (17) in problem $XQ(j)$ leads to the following problem:

$$\min \sum_{k=1}^{r} \bar{x}_{jk}(\beta_{jk} + \sum_w \alpha_{jkw} q_{jw})$$

$$\text{s.t. } q_{jw}^{\mathrm{L}} \le q_{jw} \le q_{jw}^{\mathrm{U}} \qquad w = 1, ..., l$$

The above problem now has only l variables of the type q_{jw} and is a linear program. As all the extreme points of this linear program have the q_{jw}'s at one of its bounds, q_{jw}^{L} or q_{jw}^{U}, there exists an optimal solution to the above problem having every q_{jw} at one of its bounds.

Thus, given any solution vector (x_{jk}, q_{jw}), we can construct by the method above a solution ($\bar{x}_{jk}$, $\bar{q}_{jw}$), which is at least as good as (x_{jk}, q_{jw}) and where each $\bar{x}_{jk}$ and each $\bar{q}_{jw}$ is at some bound. There exists therefore an optimal solution of ($XQ(j)$) having this property.

Problem F is also a linear program in f_{ij} with all extreme points being the bounds of f_{ij} and will have an optimal solution at one of the bounds of f_{ij} by a similar argument. □

Corollary 4.3. *There exists an optimal solution to* (LSP1)*, which has every variable at one of its bounds.*

The separability of ($LSP1$) into $p + 1$ subproblems and the characterization of an optimal solution for problems $XQ(j)$ and F imply that problem ($LSP1$) can be solved to optimality if we can solve each ($XQ(j)$) and (F) to optimality. (F) is a linear program defined over a rectangular region and is therefore easy to solve. Each ($XQ(j)$) is of the same form; hence, we denote a generic problem of the type ($XQ(j)$) as (XQ). We are interested in solving problems of the type (XQ):

(XQ)

$$\min \sum_{k=1}^{r} x_k(\beta_k + \sum_{w=1}^{l} \alpha_{kw} q_w)$$

$$\text{s.t. } x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$q_w^{\mathrm{L}} \le q_w \le q_w^{\mathrm{U}} \qquad w = 1, ..., l$$

We now reformulate (XQ) into a mixed integer program, ($XQYR$), by defining binary variables y_w and variables ζ_{kw} as follows:

$$q_w = q_w^{\mathrm{L}} + (q_w^{\mathrm{U}} - q_w^{\mathrm{L}}) y_w \qquad w = 1, ..., l \qquad (18)$$

$$\zeta_{kw} = x_k y_w \qquad w = 1, ..., l \qquad (19)$$

$$y_w \in \{0, 1\} \qquad w = 1, ..., l$$

Variables y_w replace q_w in (XQ), while those of ζ_{kw} replace the resulting bilinear terms. Equations 20−23 in problem ($XQYR$) below are obtained by replacing the bilinear term in (19) by the epigraph and hypograph of their convex and concave envelopes, respectively, provided by the McCormick over- and underestimators:[27]

($XQYR$)

$$\min \sum_k (\beta_k + \sum_w \alpha_{kw} q_w^{\mathrm{L}}) x_k + \sum_w ((q_w^{\mathrm{U}} - q_w^{\mathrm{L}}) \sum_k \alpha_{kw} \zeta_{kw})$$

$$\text{s.t. } \zeta_{kw} - y_w^{\mathrm{L}} x_k - x_k^{\mathrm{L}} y_w \ge -y_w^{\mathrm{L}} x_k^{\mathrm{L}} \qquad k = 1, ..., r; \quad w = 1, ..., l \qquad (20)$$

$$\zeta_{kw} - y_w^{\mathrm{U}} x_k - x_k^{\mathrm{U}} y_w \ge -y_w^{\mathrm{U}} x_k^{\mathrm{U}} \qquad k = 1, ..., r; \; w = 1, ..., l \qquad (21)$$

$$\zeta_{kw} - y_w^{\mathrm{U}} x_k - x_k^{\mathrm{L}} y_w \le -y_w^{\mathrm{U}} x_k^{\mathrm{L}} \qquad k = 1, ..., r; \; w = 1, ..., l \qquad (22)$$

$$\zeta_{kw} - y_w^{\mathrm{L}} x_k - x_k^{\mathrm{U}} y_w \le -y_w^{\mathrm{L}} x_k^{\mathrm{U}} \qquad k = 1, ..., r; \; w = 1, ..., l \qquad (23)$$

$$x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$y_w \in \{0, 1\} \qquad w = 1, ..., l$$

Theorem 4.4. (XQYR) *is equivalent to* (XQ).

Proof. The proof has two parts. In the first part (⇐), we show that an optimal solution to (XQ) is feasible to

(*XQYR*) with the same objective function value. In the second part (⇒), we show that an optimal solution to (*XQYR*) is feasible to (*XQ*) with the same objective function value.

⇐ From proposition 4.2, it follows that there exists an optimal solution to (*XQ*) which has x_k, $k = 1, ..., r$, and q_w, $w = 1, ..., l$, at one of their bounds. Let us denote such an optimal solution of (*XQ*) by ($\bar{x}_k$, $\bar{q}_w$).

We proceed to construct a feasible solution (x_k, y_w, ζ_{kw}) to (*XQYR*) from ($\bar{x}_k$, $\bar{q}_w$). We assign to y_w the following binary values based on $\bar{q}_w$ as follows:

$$y_w = \bar{y}_w = \begin{cases} 0 & \text{if } \bar{q}_w = q_w^{\mathrm{L}} \\ 1 & \text{if } \bar{q}_w = q_w^{\mathrm{U}} \end{cases} \tag{24}$$

Equation 24 follows directly from (18).

Setting $x_k = \bar{x}_k$ for (*XQYR*) makes x_k feasible to (*XQYR*). We let $\zeta_{kw} = \bar{x}_k\bar{y}_w$. ($\zeta_{kw}$, $\bar{x}_k$, $\bar{y}_w$) is feasible to (20)−(23).

The solution (x_k, y_w, ζ_{kw}) is thus feasible to (*XQYR*). We now prove that the solution (x_k, y_w, ζ_{kw}) also has the same objective function value as the optimal solution for (*XQ*).

Let $\vartheta(\cdot)$ and $\vartheta_{\mathrm{opt}}(\cdot)$ denote the objective function value and optimal objective function value of a given problem (·). Then,

$$\begin{aligned}
\vartheta(XQYR) &= \sum_k(\beta_k + \sum_w \alpha_{kw} q_w^{\mathrm{L}})x_k + \\
&\qquad \sum_w (q_w^{\mathrm{U}} - q_w^{\mathrm{L}}) \sum_k \alpha_{kw}\zeta_{kw} \\
&= \sum_k(\beta_k + \sum_w \alpha_{kw} q_w^{\mathrm{L}})\bar{x}_k + \\
&\qquad \sum_w (q_w^{\mathrm{U}} - q_w^{\mathrm{L}}) \sum_k \alpha_{kw}\bar{x}_k\bar{y}_w \\
&= \sum_k \bar{x}_k(\beta_k + \sum_w \alpha_{kw}(q_w^{\mathrm{L}} + (q_w^{\mathrm{U}} - q_w^{\mathrm{L}})\bar{y}_w)) \\
&= \sum_k \bar{x}_k(\beta_k + \sum_w \alpha_{kw}\bar{q}_w) \\
&= \vartheta_{\mathrm{opt}}(XQ)
\end{aligned} \tag{25}$$

It follows then from (25) that

$$\vartheta_{\mathrm{opt}}(XQYR) \le \vartheta(XQYR) \le \vartheta_{\mathrm{opt}}(XQ) \tag{26}$$

⇒ We now prove that, given an optimal solution (x_k, y_w, ζ_{kw}) to (*XQYR*), we can construct a solution ($\bar{x}_k$, $\bar{q}_w$) feasible to (*XQ*).

Consider an optimal solution of (*XQYR*). Note $y_w \in \{0, 1\}$ implies $y_w^{\mathrm{L}} = 0$ and $y_w^{\mathrm{U}} = 1$. Substituting these values of y_w in eqs 20−23 leads to the following:

$$\zeta_{kw} = \begin{cases} 0 & \text{if } y_w = y_w^{\mathrm{L}} = 0 \\ x_k & \text{if } y_w = y_w^{\mathrm{U}} = 1 \end{cases} \tag{27}$$

Thus, given any optimal solution to (*XQYR*), one can determine ζ_{kw} from (27). Replacing ζ_{kw} in (*XQYR*) by the corresponding quantities in (27) and given that y_w are known reduces (*XQYR*) to a linear program in variables x_k. As seen from (27) the effect of constraints (20)−(23) is to fix the value of ζ_{kw} in any optimal solution. Thus, once ζ_{kw} is fixed to one of the two values, depending on the value of y_w, constraints 20−23 are not explicitly required, and (*XQYR*) becomes a linear program over variables x_k. This linear program has all its extreme points at the bounds of x_k and, consequently, has an optimal solution where x_k is at its bounds. We denote such an optimal solution to (*XQYR*) as ($\bar{x}_k$, $\bar{y}_w$, ζ_{kw}). In such an optimal solution, $\bar{x}_k$ is at some bound, $\bar{y}_w$ has binary values, and ζ_{kw} is defined as in (27).

We can now construct $\bar{q}_w$ from the optimal solution of (*XQYR*) by a procedure similar to that given in (18). Since y_w is binary, it follows that $\bar{q}_w$ will be at one of its bounds.

The solution $\bar{x}_k$ and q_w constructed above is feasible to (*XQ*) since ($\bar{x}_k$, $\bar{q}_w$) has x_k and q_w at some bound.

Thus, any optimal solution of (*XQYR*) can be transformed into a feasible solution of (*XQ*), by the transformation defined above. We now prove that this transformation leaves the objective function value of (*XQYR*) and (*XQ*) the same. For this, we only need to replace ζ_{kw} in (*XQYR*) by (27). We can use the transformation $\zeta_{kw} = x_k y_w$ which is identical to (27) when y_w is binary. The optimal solution ($\bar{x}_k$, $\bar{y}_w$, ζ_{kw}) to (*XQYR*) and a corresponding feasible solution, ($\bar{x}_k$, $\bar{q}_w$), to (*XQ*) have the same objective function value. The proof of this is very similar to that provided in (25).

It follows then that

$$\vartheta_{\mathrm{opt}}(XQYR) \ge \vartheta(XQ) \ge \vartheta_{\mathrm{opt}}(XQ) \tag{28}$$

Combining (26) and (28), it follows that

$$\vartheta_{\mathrm{opt}}(XQYR) = \vartheta_{\mathrm{opt}}(XQ)$$

□

Theorem 4.4 suggests a method for solving the Lagrangian subproblem (*LSP*1). For each pool *j*, we can solve a mixed integer program (*XQYR*) instead of (*XQ*(*j*)) as (*XQYR*) has the same optimal solution as (*XQ*(*j*)). Problem *F* can be solved as a linear program. Thus, the solution of each Lagrangian subproblem is obtained by solving *p* mixed integer programs and a linear program.

5. Properties of the Lagrangian Subproblem

We now consider some properties of the Lagrangian subproblem (LSP1), which relate to the use of the Lagrangian procedure for the pooling problem.

We first state the following result which is well-known:[6]

Lemma 5.1. *Let* X *be a given set, not necessarily convex, and let* $\boldsymbol{c}$ *be a vector of coefficients. Then,*

$$min\{\boldsymbol{c}\mathrm{x}:\ \mathrm{x} \in \mathrm{X}\} = min\{\boldsymbol{c}\mathrm{x}:\ \mathrm{x} \in conv(\mathrm{X})\}$$

where

conv(X) *refers to the convex hull of* X

The following theorem states an important property of the Lagrangian relaxation procedure.

Theorem 5.2. *Solving the Lagrangian subproblem* (XQ) *to optimality is equivalent to convexifying* $\sum_{k=1}^{r}\sum_{w=1}^{l}\alpha_{kw}x_k q_w$ *over the hypercube* $\{x_k^L \le x_k \le x_k^U;\ k = 1, ..., r,\ q_w^L \le q_w \le q_w^U;\ w = 1, ..., l\}$.

Proof. (*XQ*) can be solved to optimality by solving a problem of the type (*XQYR*), as shown in Section 4. By a redefinition of terms, it can be easily seen that (*XQ*) is exactly the same as the following problem (*XQL*) and

therefore has the same optimal solution and optimal objective function value.

(*XQL*)

$$\min \sum_{k=1}^{r} \beta_k x_k + \eta(x_1,...,x_r,q_1,...,q_l)$$

$$\text{s.t. } \eta(x_1,...,x_r,q_1,...,q_l) = \sum_{k=1}^{r}\sum_{w=1}^{l} \alpha_{kw} x_k q_w$$

$$x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$q_w^{\mathrm{L}} \le q_w \le q_w^{\mathrm{U}} \qquad w = 1, ..., l$$

Problem *XQL* has a linear objective function and therefore the results of lemma 5.1 hold for this case. Consequently, (*XQL*) can be transformed into problem *XQL*2, given below, having the same optimal objective function value as (*XQ*).

(*XQL*2)

$$\min \sum_{k=1}^{r} x_k \beta_k + \eta(x_1,...,x_r,q_1,...,q_l)$$

$$\text{s.t. } (x, \eta) \in \text{conv}$$

$$\left\{ \begin{array}{ll} \eta(x_1,...,x_r,q_1,...,q_l) = \sum_{k=1}^{r} \sum_{w=1}^{l} \alpha_{kw} x_k q_w & \\ x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}} & k = 1, ..., r \\ q_w^{\mathrm{L}} \le q_w \le q_w^{\mathrm{U}} & w = 1, ..., l \end{array} \right\}$$

If $\vartheta(\cdot)$ denotes the optimal objective function value of problem $(\cdot)$, it follows from the above that

$$\vartheta(XQL2) = \vartheta(XQL) = \vartheta(XQ)$$

Consequently, solving (*XQ*) to optimality is equivalent to convexifying η over a hypercube. □

The above theorem and proposition 4.1 have the following implication. Consider the Lagrangian subproblem (*LSP*1). Theorem 5.2 implies that the Lagrangian procedure convexifies the bilinear term $\sum_j \sum_k \sum_w (v'_{jw} + u'_{kw}) x_{jk} q_{jw}$ in (*LSP*1) over the hypercube defined by the bounds on the variables x_{jk} and q_{jw}.

For the special case when $l = 1$, there exists just a single variable q_1 and therefore $\eta(x_1,...,x_r,q_1) = \sum_{k=1}^{r} \alpha_k x_k q_1$. Solving (*XQ*) for this special case is then, according to theorem 5.2, equivalent to convexifying η over a hypercube defined by variable bounds. We state without proof the following lemma that appeared in Rikun:[33]

Lemma 5.3. *Let* P *be a Cartesian product of polytopes,* $\mathrm{P} = \mathrm{P}_0 \times \mathrm{P}_1 \times ... \mathrm{P}_r$, $\mathrm{P}_k \in R^{n_k}$, *and let* $g_k(x_0, x_k)$ *be a continuous function defined on* $\mathrm{P}_0 \times \mathrm{P}_k$, $k = 1, ..., r$. *If each* $g_k(x_0, x_k)$ *is a concave function of* x_0 *when* x_k *is fixed and* P_0 *is a simplex, then*

$$conv_{\mathrm{P}}\left(\sum_{m} g_k(x_0,x_k)\right) = \sum_{m} conv\ g_k(x_0, x_k) \qquad (29)$$

where $conv_{\mathrm{P}}$ *refers to the convex envelope over the region* P.

Theorem 5.4. *For the case* $l = 1$, *the McCormick over- and underestimators for the terms* $q_1 x_k$, $k = 1, ..., r$, *convexify the term* $\eta(x_1,...,xr,q_1) = \sum_{k=1}^{r} \alpha_k x_k q_1$, *over the hypercube*

$$HP = \left\{ \begin{array}{l} x_k^{\mathrm{L}} \le x_k \le x_k^{\mathrm{U}} \quad k = 1, ..., r \\ q_1^{\mathrm{L}} \le q_1 \le q_1^{\mathrm{U}} \end{array} \right\}$$

Proof. We define

$$g_k(x_0,x_k) = g_k(q_1,x_k) = \alpha_k x_k q_1 \qquad (30)$$

The preceding definitions satisfy the conditions of lemma 5.3. Note that the polytope P_0 stated in lemma 5.3 is given by the bounds on q_1, while each polytope P_k, $k = 1, ..., r$, is defined by the bounds on variable x_k. The following result follows directly from lemma 5.3:

$$\text{conv}_P \sum_k \alpha_k x_k q_1 = \sum_k \text{conv}_P\ \alpha_k x_k q_1 \qquad (31)$$

By the definition of η,

$$\text{conv}_P\ \eta(x_1,...,x_r,q_1) = \sum_k \text{conv}_P\ \alpha_k x_k q_1 \qquad (32)$$

It is wellknown[28] that the epigraph and hypograph of the convex and concave envelopes of bilinear terms of the form $\alpha_k q_1 x_k$ over the hypercube *HP* are provided by the McCormick over- and underestimators of $q_1 x_k$. We define a new variable ξ_k, where

$$\xi_k = q_1 x_k \qquad (33)$$

The epigraph of the convex envelope of ξ_k over *HP* is provided by the following polyhedral set:

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{L}} q_1 \ge -q_1^{\mathrm{L}}\, x_k^{\mathrm{L}}$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{U}} q_1 \ge -q_1^{\mathrm{U}}\, x_k^{\mathrm{U}}$$

Similarly, the hypograph of the concave envelope of ξ_k over *HP* is provided by the following:

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{L}} q_1 \le -q_1^{\mathrm{U}}\, x_k^{\mathrm{L}}$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{U}} q_1 \le -q_1^{\mathrm{L}}\, x_k^{\mathrm{U}}$$

Depending on whether α_k is greater/lesser than zero, the convex/concave envelope of ξ_k is used to construct the envelopes of $\alpha_k q_1 x_k$ over the *HP*. The epigraph of the convex envelope of $z_k = \alpha_k q_1 x_k$ over *HP* is

(*CE*1)

$$z_k = \alpha_k \xi_x$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{L}} q_1 \ge -q_1^{\mathrm{L}}\, x_k^{\mathrm{L}} \qquad \alpha_k > 0$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{U}} q_1 \ge -q_1^{\mathrm{U}}\, x_k^{\mathrm{U}} \qquad \alpha_k > 0$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{L}} q_1 \le -q_1^{\mathrm{U}}\, x_k^{\mathrm{L}} \qquad \alpha_k < 0$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{U}} q_1 \le -q_1^{\mathrm{L}}\, x_k^{\mathrm{U}} \qquad \alpha_k < 0$$

For the rest of this article, we refer to the set of three equations in *CE*1 as the convex envelope of the term z_k.

From lemma 5.3 and eq 32 it follows that the epigraph of the convex envelope of η for the case $l = 1$ is a superset of *CE* below.

(*CE*)

$$\eta = \sum_{k=1}^{r} \alpha_k \xi_k$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{L}} q_1 \geq -q_1^{\mathrm{L}} x_k^{\mathrm{L}} \qquad k = 1, ..., r$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{U}} q_1 \geq -q_1^{\mathrm{U}} x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{L}} q_1 \leq -q_1^{\mathrm{U}} x_k^{\mathrm{L}} \qquad k = 1, ..., r$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{U}} q_1 \leq -q_1^{\mathrm{L}} x_k^{\mathrm{U}} \qquad k = 1, ..., r \quad \square$$

Theorem 5.5. *If* $l = 1$, *(*XQ*) and, therefore, (*XQYR*) can be solved as a linear program.*

For the special case of $l = 1$, problem *XQ* reduces to the following problem:

(*XQ*1)

$$\min \sum_k x_k(\beta_k + \alpha_k q_1)$$

$$\text{s.t. } x_k^{\mathrm{L}} \leq x_k \leq x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$q_1^{\mathrm{L}} \leq q_1 \leq q_1^{\mathrm{U}}$$

Defining $\eta(x_1,...,x_r,q_1) = \sum_{k=1}^{r} \alpha_k x_k q_1$, it follows from theorem 5.2 that the solution of (*XQ*1) would be the same as that obtained by convexifying η over the hypercube *HP*. As seen from theorem 5.4, the epigraph of the convex envelope of η is given by *CE*, which has only linear constraints. Thus, for the case of a single quality, problem *XQ*1 can be solved by solving the following linear program:

(*LPXQ*1)

$$\min \sum_k \beta_k x_k + \alpha_k \xi_k$$

$$\text{s.t. } \xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{L}} q_1 \geq -q_1^{\mathrm{L}} x_k^{\mathrm{L}} \qquad k = 1, ..., r \qquad (34)$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{U}} q_1 \geq -q_1^{\mathrm{U}} x_k^{\mathrm{U}} \qquad k = 1, ..., r \qquad (35)$$

$$\xi_k - q_1^{\mathrm{U}} x_k - x_k^{\mathrm{L}} q_1 \leq -q_1^{\mathrm{U}} x_k^{\mathrm{L}} \qquad k = 1, ..., r \qquad (36)$$

$$\xi_k - q_1^{\mathrm{L}} x_k - x_k^{\mathrm{U}} q_1 \leq -q_1^{\mathrm{L}} x_k^{\mathrm{U}} \qquad k = 1, ..., r \qquad (37)$$

$$x_k^{\mathrm{L}} \leq x_k \leq x_k^{\mathrm{U}} \qquad k = 1, ..., r$$

$$q_1^{\mathrm{L}} \leq q_1 \leq q_1^{\mathrm{U}}$$

(*XQ*1), which is a special case of (*XQ*) for $l = 1$, can therefore be solved as a linear program (*LPXQ*1). From theorem 4.4, it follows that (*XQYR*) can also be solved as a linear program. □

6. Comparisons of Bounds

We now apply the properties and results obtained in section 5 to the case of the pooling problem. We consider separately the single-component quality case and the multiple-quality case. In each case, we compare the lower bound obtained by the Lagrangian relaxation, (*LRP*), with the McCormick underestimating linear program, (*MCP*), defined below. We first let

$$\xi_{jkw} = q_{jw} x_{jk} \qquad (38)$$

Then

(MCP)

$$\min \sum_{j=1}^{p} \sum_{i=1}^{n_j} c_{ij} f_{ij} - \sum_{k=1}^{r} d_k \sum_{j=1}^{p} x_{jk}$$

$$\text{s.t. } \sum_{i=1}^{n_j} f_{ij} - \sum_{k=1}^{r} x_{jk} = 0 \qquad j = 1, ..., p$$

$$\sum_{k=1}^{r} \xi_{jkw} - \sum_{i=1}^{n_j} \lambda_{ij} f_{ij} = 0 \qquad j = 1, ..., p; \quad w = 1, ..., l$$

$$\sum_{j=1}^{p} x_{jk} - S_k \leq 0 \qquad k = 1, ..., r$$

$$\sum_{j=1}^{p} \xi_{jkw} - Z_{kw} \sum_{j=1}^{p} x_{jk} \leq 0 \qquad k = 1, ..., r; \quad w = 1, ..., l$$

$$\xi_{jkw} - q_{jw}^{l} x_{jk} - x_{jk}^{l} q_{jw} \geq -q_{jw}^{l} x_{jk}^{l} \qquad j = 1, ..., p; \quad k = 1, ..., r; \quad w = 1, ..., l \qquad (39)$$

$$\xi_{jkw} - q_{jw}^{u} x_{jk} - x_{jk}^{u} q_{jw} \geq -q_{jw}^{u} x_{jk}^{u} \qquad j = 1, ..., p; \quad k = 1, ..., r; \quad w = 1, ..., l \qquad (40)$$

$$\xi_{jkw} - q_{jw}^{u} x_{jk} - x_{jk}^{l} q_{jw} \leq -q_{w}^{u} x_{jk}^{l} \qquad j = 1, ..., p; \quad k = 1, ..., r; \quad w = 1, ..., l \qquad (41)$$

$$\xi_{jkw} - q_{jw}^{l} x_{jk} - x_{jk}^{u} q_{jw} \leq -q_{jw}^{l} x_{jk}^{u} \qquad j = 1, ..., p; \quad k = 1, ..., r; \quad w = 1, ..., l \qquad (42)$$

$$f_{ij}^{l} \leq f_{ij} \leq f_{ij}^{u} \qquad i = 1, ..., n_j; \quad j = 1, ..., p$$

$$q_{jw}^{l} \leq q_{jw} \leq q_{jw}^{u} \qquad j = 1, ..., p; \quad w = 1, ..., l$$

$$x_{jk}^{l} \leq x_{jk} \leq x_{jk}^{u} \qquad j = 1, ..., p; \quad k = 1, ..., r$$

6.1. Single-Quality Pooling Problems. Theorem 6.1. *For cases of pooling problems with a single-component quality, the lower bound provided by (*MCP*), is exactly the same as the lower bound obtained by Lagrangian relaxation (*LRP*) of problem* P.

Proof. The proof follows directly from theorem 5.5 and proposition 4.1. □

Corollary 6.2. *For the case of the pooling problem with a single-component quality, the Lagrangian subproblem (*LSP*1) can be solved by a sequence of linear programs.*

Proof. From theorem 5.5 it follows that, for this case, (*XQ*) can be solved as a linear program. From proposition 4.1 it follows that the optimal solution to (*LSP*1) can be obtained by solving p problems of the type (*XQ*) and a single problem (*F*). □

From theorem 6.1 it is clear that, for the single-component quality case, a Lagrangian approach such as (*LRP*) does not provide stronger lower bounds than the McCormick underestimating linear program.

6.2. Multiple-Quality Case. Theorem 6.3. *For the multiple-quality pooling problem, the Lagrangian re-*

*laxation (*LRP*) of problem* P *provides at least as strong a lower bound as the McCormick underestimating linear program, (*MCP*).*

Proof. We have seen from section 4 that the Lagrangian subproblem, (*LSP*1), can be solved through a series of mixed integer programs, (*XQYR*). This was possible as a consequence of proposition 4.1. The McCormick underestimating linear program, (*MCP*), can in theory also be solved by a Lagrangian relaxation procedure by dualizing every constraint other than the McCormick bounds and the variable bounds. As (*MCP*) is a linear program, there is no duality gap between the optimal solution of (*MCP*) and a Lagrangian relaxation of it. It can be shown, by arguments similar to those in proposition 4.1, that the Lagrangian relaxation of (*MCP*) is also separable by pools and can be solved as a single problem (*F*) and *p* problems of the kind (*XQ*(*j*)) with the bilinear terms $x_{jk}q_{jw}$ in (*XQ*(*j*)) being replaced by ξ_{jkw} and the corresponding McCormick over- and underestimators. We denote these problems as (*MCXQ*(*j*)):

(*MCXQ*(*j*))

$$\min \sum_{k=1}^{r}\beta_{jk}x_{jk} + \sum_{w=1}^{l}\alpha_{jkw}\xi_{jkw}$$

$$\text{s.t. } \xi_{jkw} - q^{l}_{jw}x_{jk} - x^{l}_{jk}q_{jw} \geq -q^{l}_{jw}\,x^{l}_{jk} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\xi_{jkw} - q^{u}_{jw}x_{jk} - x^{u}_{jk}q_{jw} \geq -q^{u}_{jw}\,x^{u}_{jk} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\xi_{jkw} - q^{u}_{jw}x_{jk} - x^{l}_{jk}q_{jw} \leq -q^{u}_{jw}\,x^{l}_{jk} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\xi_{jkw} - q^{l}_{jw}x_{jk} - x^{u}_{jk}q_{jw} \leq -q^{l}_{jw}\,x^{u}_{jk} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$x^{L}_{jk} \leq x_{jk} \leq x^{U}_{jk} \qquad k = 1, ..., r$$

$$q^{L}_{jw} \leq q_{jw} \leq q^{U}_{jw} \qquad w = 1, ..., l$$

By following a procedure similar to the one followed in section 4 to transform (*XQ*(*j*)) to (*XQYR*), we transform each problem (*MCXQ*(*j*)) to an equivalent problem (*MCXQYR*):

(*MCXQYR*)

$$\min \sum_{k}(\beta_k + \sum_{w}\alpha_{kw}q^{L}_{w})x_k + \sum_{w}(q^{U}_{w} - q^{L}_{w})\sum_{k}\alpha_{kw}\zeta_{kw}$$

$$\text{s.t. } \zeta_{kw} - y^{L}_{w}x_k - x^{L}_{k}y_w \geq -y^{L}_{w}\,x^{L}_{k} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\zeta_{kw} - y^{U}_{w}x_k - x^{U}_{k}y_w \geq -y^{U}_{w}\,x^{U}_{k} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\zeta_{kw} - y^{U}_{w}x_k - x^{L}_{k}y_w \leq -y^{U}_{w}\,x^{L}_{k} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$\zeta_{kw} - y^{L}_{w}x_k - x^{U}_{k}y_w \leq -y^{L}_{w}\,x^{U}_{k} \qquad k = 1, ..., r;\ w = 1, ..., l$$

$$x^{L}_{k} \leq x_k \leq x^{U}_{k} \qquad k = 1, ..., r$$

$$y_w \in [0, 1] \qquad w = 1, ..., l$$

Table 3. Characteristics of Test Problems

	number of			
problem	raw materials	pools	qualities	final products
Haverly 1	3	1	1	2
Haverly 2	3	1	1	2
Haverly 3	3	1	1	2
Foulds 2	6	2	1	4
Foulds 4	11	8	1	16
Foulds 5	11	4	1	16
Ben-Tal 4	4	1	1	2
Ben-Tal 5	5	3	2	5
example 1	5	2	4	4
example 2	5	2	5	4
example 3	8	3	5	4
example 4	8	2	4	5

We observe that (*MCXQYR*) is just a relaxed version of (*XQYR*), with y_w now being continuous variables. Thus, the McCormick underestimating linear program, (*MCP*), can be solved by a Lagrangian relaxation procedure similar to (*LRP*) by solving the Lagrangian subproblem using (*MCXQYR*). As (*MCXQYR*) is just a relaxation of (*XQYR*), the lower bounds obtained by solving (*XQYR*) are at least as good as those obtained from (*MCXQYR*). Consequently, the Lagrangian relaxation, (*LRP*), provides at least as good a lower bound as the McCormick underestimating linear program, (*MCP*). □

Figure 9 describes a pooling problem (Example 1) having two pools, four end-products and four quality components. For each stream entering a pool, we provide, in order, the unit purchase cost of that stream and the four quality parameters corresponding to that stream. Similarly, for each end-product, we provide its unit selling price, the maximum demand possible, and the requirements for each of its four quality components. We apply the Lagrangian relaxation procedure to obtain a lower bound on the global optimum of this problem. A lower bound can also be obtained by replacing the bilinear terms in the problem by their McCormick over- and underestimators, which provides a linear-programming relaxation to the problem.

Example 1 above provides a case where the Lagrangian relaxation provides a tighter lower bound, −939.29, than the conventional linear-programming bound, −999.31, obtained by the McCormick relaxation of bilinear terms. Hence, theorem 6.4.

Theorem 6.4. *For the multiple-quality pooling problem, the Lagrangian relaxation (*LRP*) of problem* P *may provide a strictly stronger lower bound than the McCormick underestimating linear program, (*MCP*).*

7. Computational Results

Computations were carried out for a total of 13 pooling problems, some with single and some with multiple qualities. The number of pools and streams involved in each of these problems is shown in Table 3. The three Haverly problems, the four Foulds problems, and Ben-Tal four are all single-quality problems. All other problems involve multiple qualities.

Examples 1−4 were constructed in the course of this study. All other problems were collected from the literature. In particular, problems Haverly 1, 2, and 3 are from Haverly,[18,19] problems Foulds 2, 3, 4, and 5 are from Foulds et al.,[14] and problems Ben-Tal 4 and 5 are from Ben-Tal et al.[7] The flowcharts and data for all the test problems are provided in Figures 3−12 and Tables 4−6.

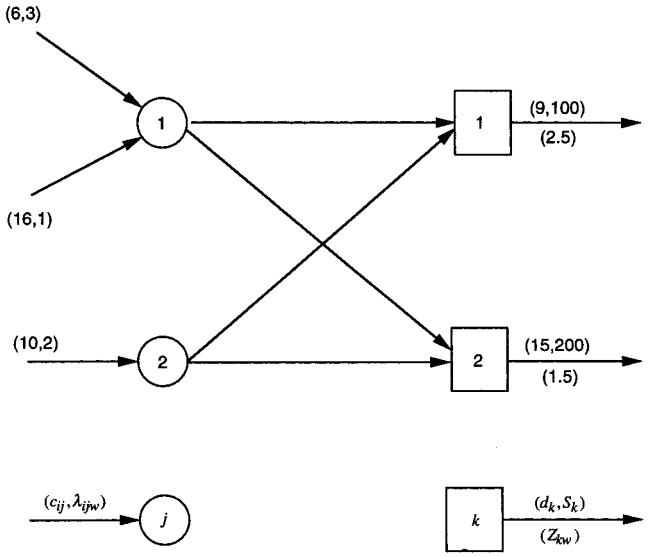

Figure 3. Flowchart for Haverly 1.

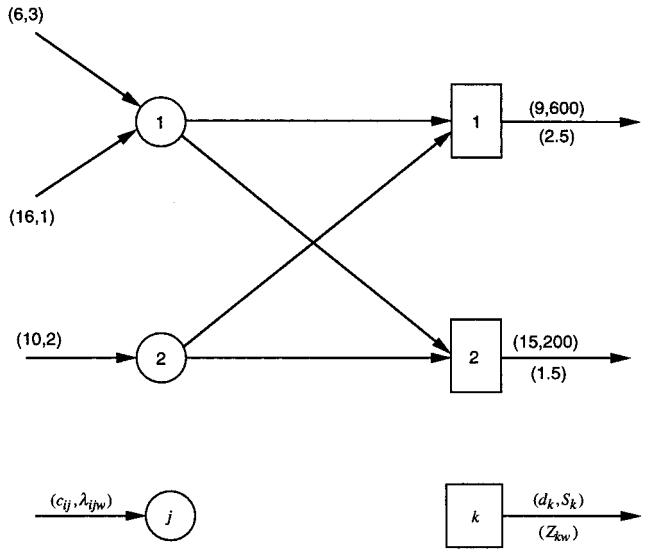

Figure 4. Flowchart for Haverly 2.

We first performed 100 iterations of a local search on each problem. The popular nonlinear-programming code MINOS[30] was used for this purpose. Table 7 shows the CPU requirements for these local searches on an IBM RS/6000 model 43P. The times shown are the total CPU times for all 100 local searches. Also shown is the number of different solutions (presumably local minima) identified for each problem during the process. The local searches were performed with and without the aid of range contraction techniques that are implemented in the global optimization system BARON.[15,36] These range contraction techniques involve a combination of feasibility-based and optimality-based arguments that are used to restrict ranges of variables. Feasibility-based techniques involve the approximate solution of linear programs to minimize and maximize individual problem variables over the linear set of constraints of the problem.[34,35,38] The optimality-based techniques make use of feasible solutions encountered during the search in conjunction with nonlinear-programming duality arguments in order to restrict ranges of variables.[34,35] In some instances, range contraction increases the CPU requirements of the local search. Apparently, as a result of range contraction, the search space becomes smaller

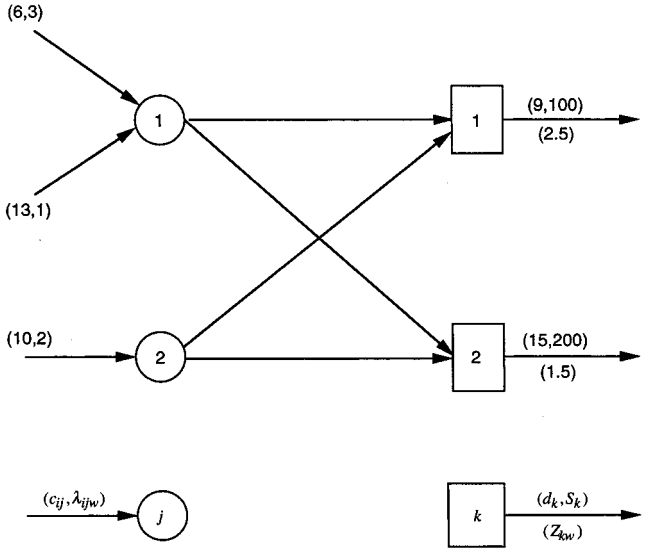

Figure 5. Flowchart for Haverly 3.

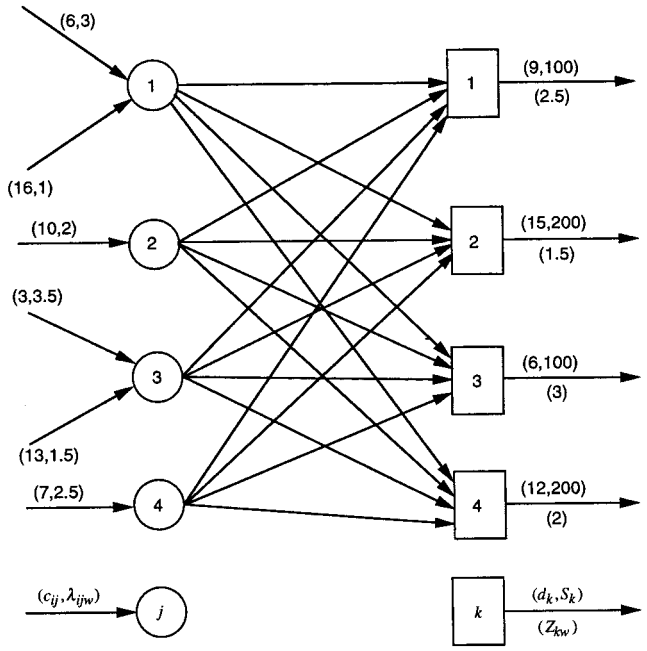

Figure 6. Flowchart for Foulds 2.

and makes it more difficult for MINOS to identify feasible solutions.

Tables 8−11 provide the values of various local solutions encountered and the percentage of times these local solution values were identified by the nonlinear-programming solver. The four problems we constructed (examples 1−4) appear to be the hardest of all test problems as MINOS failed to identify any good local solutions in the majority of the runs. The effect of range contraction is also shown in these tables. It is clear that range contraction is beneficial as it increases the likelihood that a local search will identify better solutions.

Tables 12 and 13 provide a comparison between the lower bounds obtained by solving the Lagrangian approach (*LRP*) and those obtained by solving the standard linear-programming relaxation, (*MCP*). The comparisons for the single-quality case and for the multiple-quality case are made separately.

Tables 12 and 13 indicate that the Lagrangian relaxation procedure produces at least as strong a lower bound as the linear-programming relaxation using

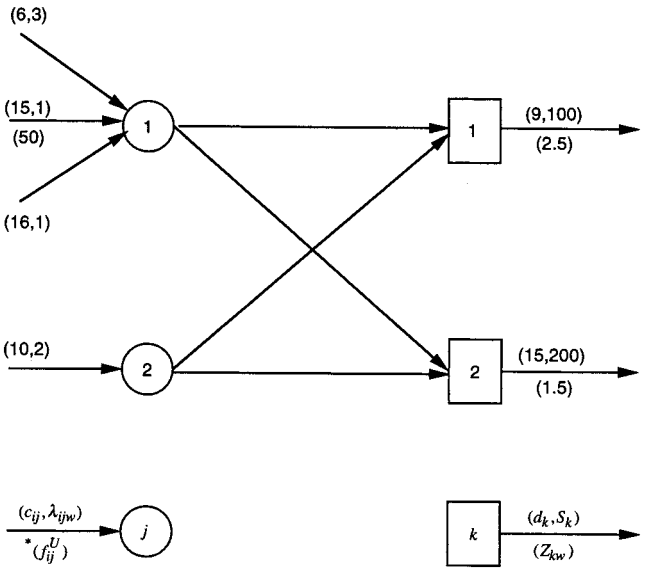

Figure 7. Flowchart for Ben-Tal 4.

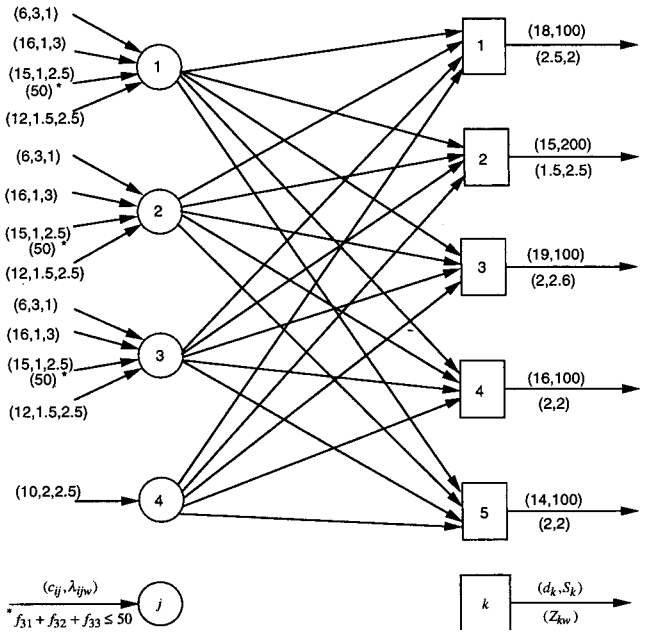

Figure 8. Flowchart for Ben-Tal 5.

McCormick over- and underestimators. For the case of pooling problems with a single-component quality, we find that the lower bounds are the same, as was proved theoretically in section 6.1. For the case of multiple qualities, Table 13 provides four cases where the Lagrangian relaxation procedure produces stronger bounds than the standard underestimating linear program linear program (LP).

The globally optimal solutions presented in Table 12 were obtained by BARON.[15,36] The software implements a branch-and-bound global optimization algorithm where lower and upper bounds are obtained through the McCormick underestimating LP and local search using MINOS, respectively. In addition to the range contraction techniques described above, the implementation involves a number of branching techniques that expedite convergence.[24,38] Table 14 presents comparative results with this global optimization approach and prior approaches to the same set of pooling problems. Clearly, our branch-and-bound algorithm presents a superior performance. Our approach requires fewer branch-and-bound nodes than the branch-and-bound approach used

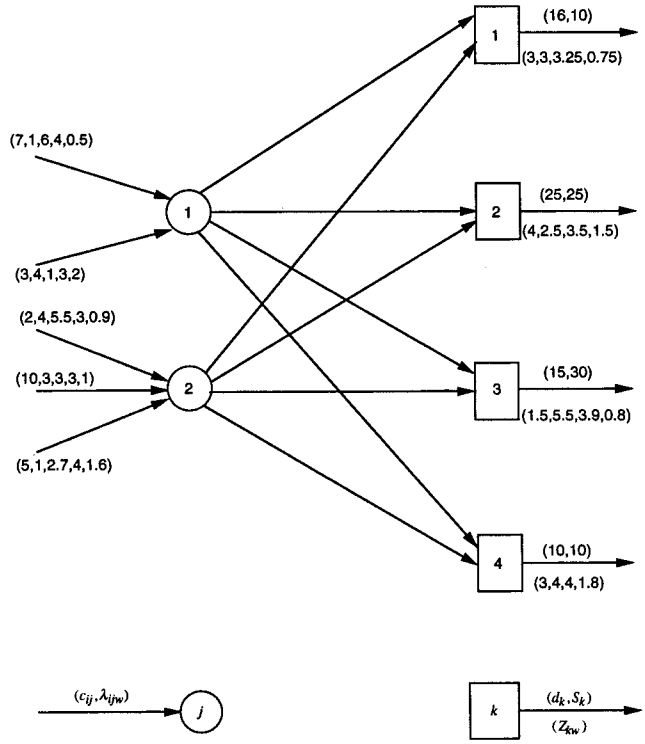

Figure 9. Flowchart for example 1.

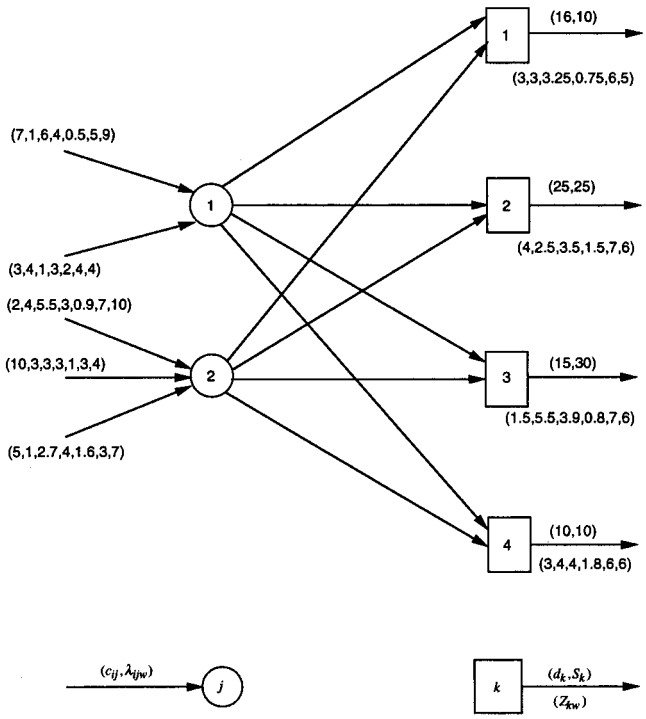

Figure 10. Flowchart for example 2.

by Foulds et al.[14] The same observation holds for Ben-Tal's test problems when our approach is compared to that of Ben-Tal et al.[7] Our algorithm also requires a much smaller number of iterations than the earlier approaches of Floudas and co-workers.[40,41] This results in significantly smaller CPU times despite the fact that our computations were carried out on a much slower computer with a much stricter termination condition. The main conclusion by looking at Table 14 is that all pooling problems currently in the open literature are easy problems as far as our branch-and-reduce algorithm is concerned. The problems we generated (ex-

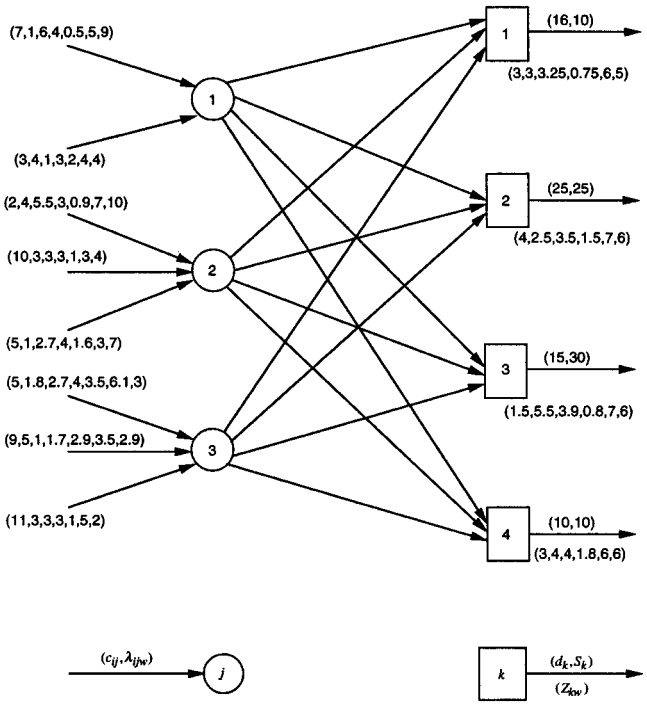

Figure 11. Flowchart for example 3.

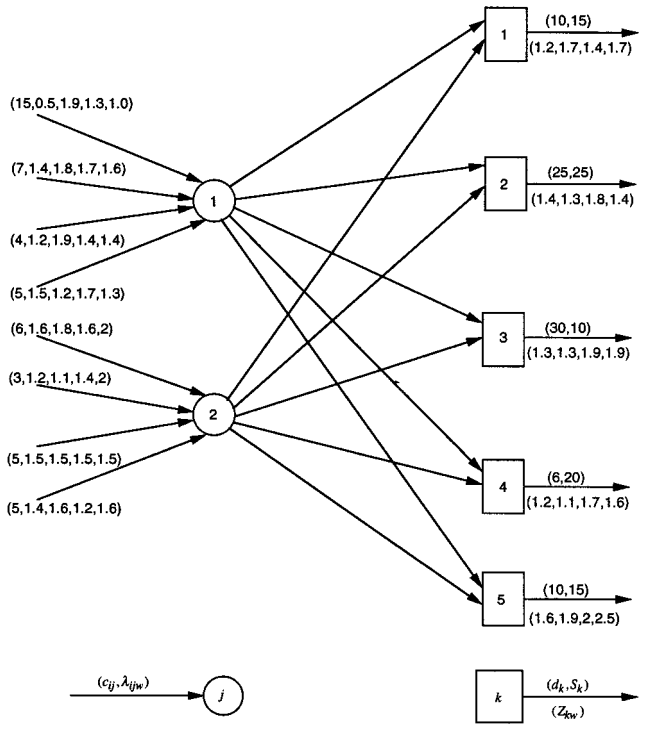

Figure 12. Flowchart for example 4.

amples 1−4), on the other hand, are much more challenging. Referring to Table 3, we note that these problems possess many more qualities than all other problems. This provides evidence that multiple-quality requirements make the pooling problem harder. The Lagrangian relaxation introduced in this paper provides stronger bounds precisely for this class of pooling problems.

8. Lagrangian Relaxation and Branch-and-Bound

We shall use an example to demonstrate a branch-and-bound procedure using the Lagrangian relaxation procedure for lower bounding. At each node of the branch-and-bound tree, a lower bound on the pooling problem is obtained by solving the Lagrangian dual problem (*LRP*) using a cutting plane approach, as described by Bazaraa et al.[6] In this approach, the Lagrangian dual is solved by an iterative procedure

Table 4. Data for Foulds 3

input parameters		end-product parameters	
$c_{11} = 20$	$\lambda_{111} = 1.0$	$Z_{11} = 1.05$	$S_1 = 1$
$c_{21} = 19$	$\lambda_{211} = 1.1$	$Z_{21} = 1.10$	$S_2 = 1$
$c_{31} = 18$	$\lambda_{311} = 1.2$	$Z_{31} = 1.15$	$S_3 = 1$
$c_{41} = 17$	$\lambda_{411} = 1.3$	$Z_{41} = 1.20$	$S_4 = 1$
$c_{12} = 19$	$\lambda_{121} = 1.1$	$Z_{51} = 1.25$	$S_5 = 1$
$c_{22} = 18$	$\lambda_{221} = 1.2$	$Z_{61} = 1.30$	$S_6 = 1$
$c_{32} = 17$	$\lambda_{321} = 1.3$	$Z_{71} = 1.35$	$S_7 = 1$
$c_{42} = 16$	$\lambda_{421} = 1.4$	$Z_{81} = 1.40$	$S_8 = 1$
$c_{13} = 18$	$\lambda_{131} = 1.2$	$Z_{91} = 1.45$	$S_9 = 1$
$c_{23} = 17$	$\lambda_{231} = 1.3$	$Z_{101} = 1.50$	$S_{10} = 1$
$c_{33} = 16$	$\lambda_{331} = 1.4$	$Z_{111} = 1.55$	$S_{11} = 1$
$c_{43} = 15$	$\lambda_{431} = 1.5$	$Z_{121} = 1.60$	$S_{12} = 1$
$c_{14} = 17$	$\lambda_{141} = 1.3$	$Z_{131} = 1.65$	$S_{13} = 1$
$c_{24} = 16$	$\lambda_{241} = 1.4$	$Z_{141} = 1.70$	$S_{14} = 1$
$c_{34} = 15$	$\lambda_{341} = 1.5$	$Z_{151} = 1.75$	$S_{15} = 1$
$c_{44} = 14$	$\lambda_{441} = 1.6$	$Z_{161} = 1.80$	$S_{16} = 1$
$c_{15} = 16$	$\lambda_{151} = 1.4$	$d_1 = 20$	
$c_{25} = 15$	$\lambda_{251} = 1.5$	$d_2 = 19.5$	
$c_{35} = 14$	$\lambda_{351} = 1.6$	$d_3 = 19$	
$c_{45} = 13$	$\lambda_{451} = 1.7$	$d_4 = 18.5$	
$c_{16} = 15$	$\lambda_{161} = 1.5$	$d_5 = 18$	
$c_{26} = 14$	$\lambda_{261} = 1.6$	$d_6 = 17.5$	
$c_{36} = 13$	$\lambda_{361} = 1.7$	$d_7 = 17$	
$c_{46} = 12$	$\lambda_{461} = 1.8$	$d_8 = 16.5$	
$c_{17} = 14$	$\lambda_{171} = 1.6$	$d_9 = 16$	
$c_{27} = 13$	$\lambda_{271} = 1.7$	$d_{10} = 15.5$	
$c_{37} = 12$	$\lambda_{371} = 1.8$	$d_{11} = 15$	
$c_{47} = 11$	$\lambda_{471} = 1.9$	$d_{12} = 14.5$	
$c_{18} = 13$	$\lambda_{181} = 1.7$	$d_{13} = 14$	
$c_{28} = 12$	$\lambda_{281} = 1.8$	$d_{14} = 13.5$	
$c_{38} = 11$	$\lambda_{381} = 1.9$	$d_{15} = 13$	
$c_{48} = 10$	$\lambda_{481} = 2.0$	$d_{16} = 12.5$	

Table 5. Data for Foulds 4

input parameters		end-product parameters	
$c_{11} = 20$	$\lambda_{111} = 1.0$	$Z_{11} = 1.05$	$S_1 = 1$
$c_{21} = 17$	$\lambda_{211} = 1.3$	$Z_{21} = 1.10$	$S_2 = 1$
$c_{31} = 14$	$\lambda_{311} = 1.6$	$Z_{31} = 1.15$	$S_3 = 1$
$c_{41} = 11$	$\lambda_{411} = 1.9$	$Z_{41} = 1.20$	$S_4 = 1$
$c_{12} = 19$	$\lambda_{121} = 1.1$	$Z_{51} = 1.25$	$S_5 = 1$
$c_{22} = 16$	$\lambda_{221} = 1.4$	$Z_{61} = 1.30$	$S_6 = 1$
$c_{32} = 13$	$\lambda_{321} = 1.7$	$Z_{71} = 1.35$	$S_7 = 1$
$c_{42} = 10$	$\lambda_{421} = 2.0$	$Z_{81} = 1.40$	$S_8 = 1$
$c_{13} = 18$	$\lambda_{131} = 1.2$	$Z_{91} = 1.45$	$S_9 = 1$
$c_{23} = 19$	$\lambda_{231} = 1.1$	$Z_{101} = 1.50$	$S_{10} = 1$
$c_{33} = 16$	$\lambda_{331} = 1.4$	$Z_{111} = 1.55$	$S_{11} = 1$
$c_{43} = 15$	$\lambda_{431} = 1.5$	$Z_{121} = 1.60$	$S_{12} = 1$
$c_{14} = 17$	$\lambda_{141} = 1.3$	$Z_{131} = 1.65$	$S_{13} = 1$
$c_{24} = 18$	$\lambda_{241} = 1.2$	$Z_{141} = 1.70$	$S_{14} = 1$
$c_{34} = 15$	$\lambda_{341} = 1.5$	$Z_{151} = 1.75$	$S_{15} = 1$
$c_{44} = 14$	$\lambda_{441} = 1.6$	$Z_{161} = 1.80$	$S_{16} = 1$
$c_{15} = 16$	$\lambda_{151} = 1.4$	$d_1 = 20$	
$c_{25} = 15$	$\lambda_{251} = 1.5$	$d_2 = 19.5$	
$c_{35} = 18$	$\lambda_{351} = 1.2$	$d_3 = 19$	
$c_{45} = 13$	$\lambda_{451} = 1.7$	$d_4 = 18.5$	
$c_{16} = 15$	$\lambda_{161} = 1.5$	$d_5 = 18$	
$c_{26} = 14$	$\lambda_{261} = 1.6$	$d_6 = 17.5$	
$c_{36} = 17$	$\lambda_{361} = 1.3$	$d_7 = 17$	
$c_{46} = 12$	$\lambda_{461} = 1.8$	$d_8 = 16.5$	
$c_{17} = 14$	$\lambda_{171} = 1.6$	$d_9 = 16$	
$c_{27} = 13$	$\lambda_{271} = 1.7$	$d_{10} = 15.5$	
$c_{37} = 12$	$\lambda_{371} = 1.8$	$d_{11} = 15$	
$c_{47} = 17$	$\lambda_{471} = 1.3$	$d_{12} = 14.5$	
$c_{18} = 13$	$\lambda_{181} = 1.7$	$d_{13} = 14$	
$c_{28} = 12$	$\lambda_{281} = 1.8$	$d_{14} = 13.5$	
$c_{38} = 11$	$\lambda_{381} = 1.9$	$d_{15} = 13$	
$c_{48} = 16$	$\lambda_{481} = 1.4$	$d_{16} = 12.5$	

Table 6. Data for Foulds 5

input parameters		end-product parameters	
$c_{11} = 20$	$\lambda_{111} = 1.0$	$Z_{11} = 1.05$	$S_1 = 1$
$c_{21} = 19$	$\lambda_{211} = 1.1$	$Z_{21} = 1.10$	$S_2 = 1$
$c_{31} = 18$	$\lambda_{311} = 1.2$	$Z_{31} = 1.15$	$S_3 = 1$
$c_{41} = 17$	$\lambda_{411} = 1.3$	$Z_{41} = 1.20$	$S_4 = 1$
$c_{51} = 13$	$\lambda_{511} = 1.7$	$Z_{51} = 1.25$	$S_5 = 1$
$c_{61} = 12$	$\lambda_{611} = 1.8$	$Z_{61} = 1.30$	$S_6 = 1$
$c_{71} = 11$	$\lambda_{711} = 1.9$	$Z_{71} = 1.35$	$S_7 = 1$
$c_{81} = 10$	$\lambda_{811} = 2.0$	$Z_{81} = 1.40$	$S_8 = 1$
$c_{12} = 19$	$\lambda_{121} = 1.1$	$Z_{91} = 1.45$	$S_9 = 1$
$c_{22} = 18$	$\lambda_{221} = 1.2$	$Z_{101} = 1.50$	$S_{10} = 1$
$c_{32} = 17$	$\lambda_{321} = 1.3$	$Z_{111} = 1.55$	$S_{11} = 1$
$c_{42} = 16$	$\lambda_{421} = 1.4$	$Z_{121} = 1.60$	$S_{12} = 1$
$c_{52} = 14$	$\lambda_{521} = 1.6$	$Z_{131} = 1.65$	$S_{13} = 1$
$c_{62} = 13$	$\lambda_{621} = 1.7$	$Z_{141} = 1.70$	$S_{14} = 1$
$c_{72} = 12$	$\lambda_{721} = 1.8$	$Z_{151} = 1.75$	$S_{15} = 1$
$c_{82} = 11$	$\lambda_{821} = 1.9$	$Z_{161} = 1.80$	$S_{16} = 1$
$c_{13} = 17$	$\lambda_{131} = 1.3$	$d_1 = 20$	
$c_{23} = 16$	$\lambda_{231} = 1.4$	$d_2 = 19.5$	
$c_{33} = 15$	$\lambda_{331} = 1.5$	$d_3 = 19$	
$c_{43} = 14$	$\lambda_{431} = 1.6$	$d_4 = 18.5$	
$c_{53} = 13$	$\lambda_{531} = 1.7$	$d_5 = 18$	
$c_{63} = 12$	$\lambda_{631} = 1.8$	$d_6 = 17.5$	
$c_{73} = 11$	$\lambda_{731} = 1.9$	$d_7 = 17$	
$c_{83} = 10$	$\lambda_{831} = 2.0$	$d_8 = 16.5$	
$c_{14} = 20$	$\lambda_{141} = 1.0$	$d_9 = 16$	
$c_{24} = 19$	$\lambda_{241} = 1.1$	$d_{10} = 15.5$	
$c_{34} = 18$	$\lambda_{341} = 1.2$	$d_{11} = 15$	
$c_{44} = 17$	$\lambda_{441} = 1.3$	$d_{12} = 14.5$	
$c_{54} = 16$	$\lambda_{541} = 1.4$	$d_{13} = 14$	
$c_{64} = 15$	$\lambda_{641} = 1.5$	$d_{14} = 13.5$	
$c_{74} = 14$	$\lambda_{741} = 1.9$	$d_{15} = 13$	
$c_{84} = 13$	$\lambda_{841} = 1.7$	$d_{16} = 12.5$	

Table 7. Local Search Results

	without reduction		with reduction	
problem	time (s)	solutions	time (s)	solutions
Haverly 1	2	3	2	3
Haverly 2	2	3	2	3
Haverly 3	3	3	3	3
Foulds 2	15	7	21	6
Foulds 3	923	3	997	3
Foulds 4	2099	4	1891	5
Foulds 5	692	7	928	6
Ben-Tal 4	2	3	3	3
Ben-Tal 5	246	6	219	5
example 1	18	7	18	6
example 2	12	3	14	3
example 3	59	8	83	10
example 4	59	12	115	13

Table 8. Local Search Results for Haverly's Problems

values of local solutions obtained (% occurrence)		
Haverly 1	Haverly 2	Haverly 3
Without Reduction		
0 (24)	0 (37)	0 (8)
−100 (20)	−400 (45)	−125 (29)
−400 (56)	−600 (18)	−750 (63)
With Reduction		
0 (21)	0 (37)	0 (5)
−100 (16)	−400 (44)	−125 (23)
−400 (63)	−600 (19)	−750 (72)

consisting of a sequence of master problems and Lagrangian subproblems—the master problems providing the Lagrange multipliers, and the Lagrangian subproblem, (LSP), generating cuts which are added to the master problem. The optimal solutions to the master problem form a nonincreasing sequence and provide an upper bound to the optimal value of the Lagrangian dual. The optimal solution of the Lagrangian subproblem provides a lower bound on the optimal dual solution. Consequently, the iterative procedure for generating the lower bound at any node can be terminated when the solution to the master problem is within a specified tolerance of the highest Lagrangian subproblem solution obtained.

Consider Figure 13 which provides an example of a small pooling problem with a single pool, four end-products, and four quality components.

Table 9. Local Search Results for Fould's Problems

values of local optima obtained (% occurrence)			
Foulds 2	Foulds 3	Foulds 4	Foulds 5
Without Reduction			
∞ (2)	−7 (2)	−4 (2)	∞ (2)
−600 (19)	−7.5 (10)	−6.5 (3)	−1 (1)
−674 (1)	−8 (88)	−7.5 (34)	−3 (2)
−699 (1)		−8 (61)	−3.5 (1)
−700 (13)			−7 (1)
−1000 (44)			−7.5 (5)
−1100 (20)			−8 (88)
With Reduction			
∞ (3)	−7 (2)	∞ (2)	∞ (1)
−600 (13)	−7.5 (7)	−4 (2)	−1 (1)
−700 (8)	−8 (91)	−6.5 (1)	−3 (2)
−840 (1)		−7.5 (33)	−3.5 (1)
−1000 (56)		−8 (62)	−7.5 (4)
−1100 (19)			−8 (92)

Table 10. Local Search Results for Ben-Tal's Problems

values of local optima obtained (% occurrence)	
Ben-Tal 4	Ben-Tal 5
Without Reduction	
0 (9)	∞ (3)
−100 (29)	−900 (2)
−450 (62)	−1900 (6)
	−2700 (1)
	−2700 (8)
	−3500 (80)
With Reduction	
0 (8)	∞ (3)
−100 (21)	−900 (2)
−450 (71)	−2700 (1)
	−2900 (10)
	−3500 (84)

Table 11. Local Search Results for Example Problems

values of local optima obtained (% occurrence)			
example 1	example 2	example 3	example 4
Without Reduction			
0 (84)	0 (98)	0 (85)	∞ (3)
−46 (1)	−33 (1)	−33 (2)	0 (47)
−57 (4)	−549.8 (1)	−57 (3)	−86 (1)
−64 (5)		−364 (1)	−88 (1)
−69 (1)		−412 (4)	−90 (5)
−341 (2)		−510 (2)	−93 (1)
−549.8 (3)		−550 (2)	−105 (14)
		−559.6 (1)	−281 (1)
			−365 (11)
			−373 (2)
			−471 (11)
			−877.6 (4)
With Reduction			
0 (84)	0 (97)	∞ (1)	∞ (19)
−56.7 (6)	−33.3 (1)	0 (86)	0 (9)
−63.9 (4)	−549.8 (2)	−33 (1)	−105 (3)
−340.9 (2)		−57 (2)	−313 (1)
−509.7 (2)		−364 (1)	−365 (2)
−549.8 (2)		−397 (1)	−471 (6)
		−412 (2)	−506 (1)
		−500 (2)	−518 (1)
		−503 (1)	−525 (1)
		−549.8 (3)	−533 (1)
			−545 (10)
			−851 (1)
			−877.6 (45)

Table 12. Comparison of Lower Bounds for Single Quality

problem	LP	Lagrangian	global optimum
Haverly 1	−500	−500	−400
Haverly 2	−1000	−1000	−600
Haverly 3	−800	−800	−750
Foulds 2	−1100	−1100	−1100
Foulds 3	−8.00	−8.00	−8.00
Foulds 4	−8.00	−8.00	−8.00
Foulds 5	−8.00	−8.00	−8.00
Ben-Tal 4	−550	−550	−450

Table 13. Comparison of Lower Bounds for Multiple Qualities

problem	LP	Lagrangian	global optimum
Ben-Tal 5	−3500	−3500	−3500
Example 1	−999.31	−939.29	−549.80
Example 2	−854.10	−825.59	−549.80
Example 3	−882.84	−864.81	−561.05
Example 4	−1012.50	−988.50	−877.65

Figure 14 depicts the lower bounding and branching part of the branch-and-bound solution to the example in Figure 13. For each node, the lower bound obtained is displayed either above or below the node. Branching was performed only on the pool quality variables q_{jw}. The branching variable is noted over each arc of the graph whereas the corresponding range of the branching variable is shown below the arc. For example, at the root node, we branch on variable q_{12} and generate two descendant nodes, one with $q_{12} \in [1, 3.5]$ and another with $q_{12} \in [3.5, 6]$.

The global optimum for the problem in Figure 13 has a value of −260. After evaluating 23 nodes in the branch-and-bound tree, we obtained a lower bound on the pooling problem corresponding to the global optimum.

Table 15 shows the number of cuts required in the master problem to get the lower bound for each node in the branch-and-bound tree, if we do not use cuts from any previous node but generate all the cuts at a given node. Note that the problem has 16 bilinear terms and a linear-programming relaxation based on the McCormick under- and overestimators would require 64 additional constraints at every node to provide the under- and overestimators for the bilinear terms.

Table 15 indicates that the number of cuts required in the master problem for each node of the branch-and-bound tree does not differ significantly from the number of cuts required at the root node, when the cuts from

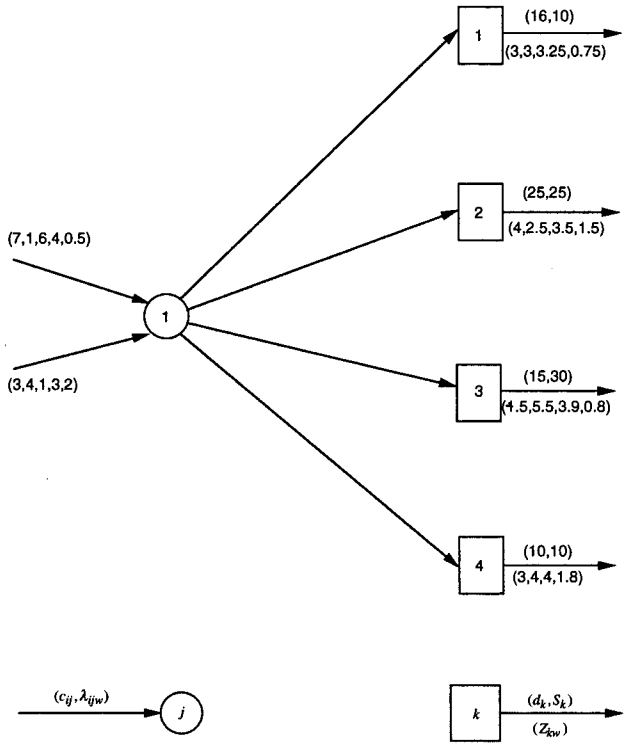

Figure 13. Example to demonstrate branch-and-bound for Lagrangian approach.

previous nodes are not used. The only exception is terminal nodes in the tree; these nodes typically require only a few cuts to be eliminated. The results of Table 15 suggest that if we could use the cuts generated at the parent node for all subsequent children nodes, then the additional cuts required at each child node would be very small, if any. The cuts generated at a given node are, in general, not valid for its children nodes as variable bounds change as a result of branching. We provide below a procedure to make these cuts valid.

The cuts that are generated at any node of the branch-and-bound procedure can be reused for all subsequent nodes by making a minor modification. From proposition 4.3, we know that there exists an optimal solution to the Lagrangian subproblem, (*LSP*), which has every variable at some bound. Considering the previous result, and the fact that every cut in the master problem is generated from an optimal solution to the Lagrangian

Table 14. Comparative Computational Results for Test Problems

algorithm	Foulds'92[14]		Ben-Tal'94[7]	Floudas'93[41]		Floudas'96[40]		BARON'99	
computer[a]	CDC 4340			HP9000/730		HP9000/730[b]		RS6000/43P	
tolerance[a]								10^{-6}	
problem	iter	CPU (s)	iter	iter	CPU (s)	iter	CPU (s)	iter	CPU (s)
Haverly 1	5	0.7	3	7	0.95	12	0.22	3	0.09
Haverly 2			3	19	3.19	12	0.21	9	0.09
Haverly 3			3			14	0.26	5	0.13
Foulds 2	9	3						1	0.10
Foulds 3	1	10.5						1	2.33
Foulds 4	25	125						1	2.59
Foulds 5	125	163.6						1	0.86
Ben-Tal 4			25	47	44.54	7	0.95	3	0.11
Ben-Tal 5			283	42	40.31	41	5.80	1	1.12
example 1								6174	425
example 2								10743	1115
example 3								79944	19314
example 4								1980	182

[a] Blank entries in this table indicate that data were not provided or problems were not solved by prior approaches.[7,14,40,41] [b] Tolerances used in ref 41 were 0.05% for Haverly 1, 2, and 3, 0.5% for Ben-Tal 4, and 1% for Ben-Tal 5.

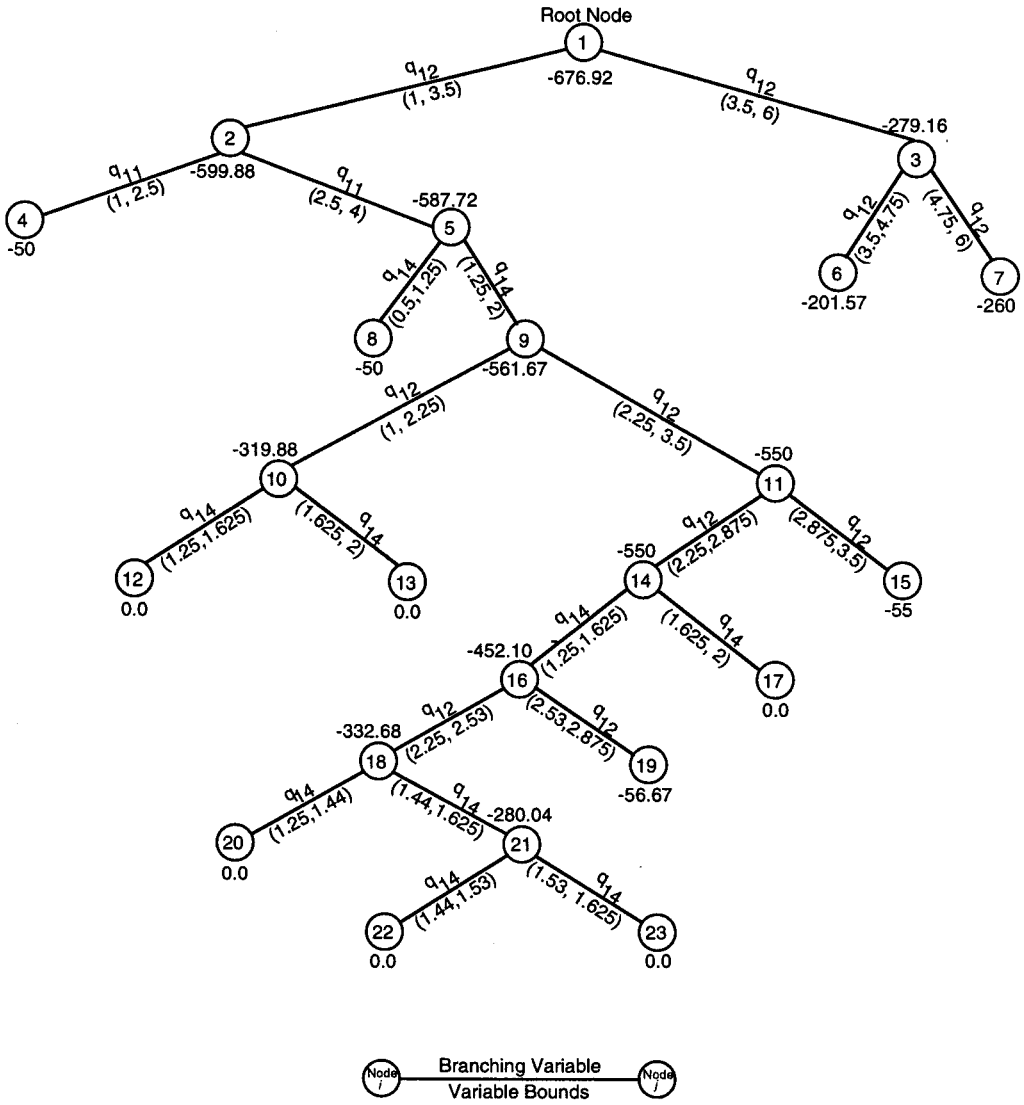

Figure 14. Branch-and-bound tree for illustrative example.

Table 15. Number of Cuts Required

node number	cuts required	node number	cuts required
1	28	14	38
2	26	15	1
3	27	16	41
4	1	17	1
5	26	18	39
6	43	19	1
7	14	20	1
8	1	21	32
9	31	22	1
10	37	23	1
11	30		
12	1		
13	1		

subproblem, it follows that any cut in the master problem can be indexed by the variable bounds which generates that cut. Indexing cuts based on variable bounds allows us to use the new bounds on the branching variable to transform cuts belonging to the parent node into valid cuts that belong to the child node. Thus, theoretically we could retain every cut from a parent node for all subsequent nodes if we follow the procedure for indexing cuts described above.

9. Conclusions

This paper introduced a Lagrangian relaxation approach for pooling and blending problems. The Lagrangian subproblem has a bilinear objective function defined over a hypercube. We provided a procedure for solving the Lagrangian subproblem to optimality and were able to derive lower bounds on the pooling problem. We discussed some theoretical properties of the Lagrangian relaxation and compared the lower bounds obtained by this approach to those obtained from the more traditional linear-programming approach based on McCormick estimators. Computational results indicate that the Lagrangian relaxation procedure provides stronger bounds than the linear-programming approach, for problems with multiple-quality parameters.

The proposed Lagrangian relaxation can be solved by a cutting plane method. We provided a procedure by which the cutting planes at any node of the branch-and-bound tree can be re-used at successor nodes. While the emphasis of the current paper is on theoretical aspects, we conjecture, on the basis of an example in the paper, that the strategy to re-use cutting planes would result in very few, if any, additional cutting planes being required at successor nodes. Further computational testing is required to validate our conjecture. Such testing will require one to resolve several implementation issues as the lower bounding procedure described here is drastically different than those used in currently existing branch-and-bound implementations.

Finally, the computational results clearly demonstrate the importance of using global optimization methods for solving pooling and blending problems. Given the scale of operations in a petrochemical refinery, even minor improvements in solutions could lead to enormous savings in costs. As examples 1−4 show, the best local optimum obtained by a popular nonlinear-programming solver may still be very far from the global optimum. Thus, it is essential that the focus of any optimization procedure for the pooling problem should be on global optimization.

Acknowledgment

We are grateful for partial financial support from the Mobil Technology Company, the DuPont Educational Aid Program, and the National Science Foundation under Grant DMII 94-14615 and CAREER Award DMII 95-02722 to N.V.S.

Literature Cited

(1) Al-Khayyal, F. A.; Falk, J. E. Jointly Constrained Biconvex Programming. *Math. Operat. Res.* **1983**, *8* (2), 124−131.

(2) Amos, F.; Rönnqvist, M.; Gill, G. Modelling the Pooling Problem at the New Zealand Refining Company. *J. Operat. Res. Soc.* **1997**, *48*, 767−778.

(3) Androulakis, I. P.; Visweswaran, V.; Floudas, C. Distributed Decomposition-Based Approaches. In *State of the Art in Global Optimization: Computational Methods and Applications*; Floudas, C. M., Pardalos, P. M., Eds.; Kluwer Academic Publishers: Dordrecht, 1996.

(4) Baker, T. E.; Lasdon, L. S. Successive Linear Programming at Exxon. *Manage. Sci.* **1985**, *31*, 264−274.

(5) Bazaraa, M. S.; Goode, J. J. A Survey of Various Tactics for Generating Lagrangian Multipliers in the Context of Lagrangian Duality. *Eur. J. Operat. Res.* **1979**, *3*, 322−338.

(6) Bazaraa, M. S.; Sherali, H. D.; Shetty, C. M. *Nonlinear Programming: Theory and Algorithms*, 2nd ed.; John Wiley & Sons, Inc.: New York, 1993.

(7) Ben-Tal, A.; Eiger, G.; Gershovitz, V. Global Minimization by Reducing the Duality Gap. *Math. Program.* ***1994***, *63*, 193−212.

(8) DeWitt, C. W.; Lasdon, L. S.; Waren, A. D.; Brenner, D. A.; Melhem, S. A. OMEGA: An Improved Gasoline Blending System for Texaco. *Interfaces* **1989**, *19* (1), 85−101.

(9) Dür, M.; Horst, R. Lagrange Duality and Partitioning Techniques in Nonconvex Global Optimization. *J. Optimiz. Theory Appl.* **1997**, *95* (2), 347−369.

(10) Falk, J. E. Lagrange Multipliers and Nonconvex Programs. *SIAM J. Control* **1969**, *7* (4), Nov, 534−545.

(11) Fieldhouse, M. The Pooling Problem. In *Optimization in Industry*; Ciriani, T. A., Leachman, R. C., Eds.; John Wiley & Sons Ltd: New York, 1993.

(12) Fisher, M. L. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces* **1985**, *15* (2), 10−21.

(13) Floudas, C. A.; Aggarwal, A. A Decomposition Strategy for Optimum Search in the Pooling Problem. *ORSA J. Comput.* **1990**, *2* (3), 225−235.

(14) Foulds, L. R.; Haugland, D.; Jornsten, K. A Bilinear Approach to the Pooling Problem. *Optimization* **1992**, *24*, 165−180.

(15) Ghildyal, V.; Sahinidis, N. V. Solving Global Optimization Problems with BARON. In *From Local to Global Optimization: A Workshop on the Occasion of the 70th Birthday of Professor Hoang Tuy*; Migdalas, A., Pardalos, P., Varbrand, P., Holmqvist, K., Eds.; Kluwer Academic Publishers: Boston, MA, 1998.

(16) Gourdin, E.; Hansen, P.; Jaumard, B. A Convergent Generalised Benders Decomposition Algorithm for Solving Bilinear and Quadratic Programming Problems. Presented at the Mathematical Programming Symposium, Lausanne, Switzerland, Aug 1997; paper WE3-C-CO122.

(17) Greenberg, H. J. Analysing the Pooling Problem. *ORSA J. Comput.* **1995**, *7* (2), 205−217.

(18) Haverly, C. A. Studies of the Behaviour of Recursion for the Pooling Problem. *ACM SIGMAP Bull.* **1978**, *25*, 29−32.

(19) Haverly, C. A. Behaviour of Recursion Model−More Studies. *ACM SIGMAP Bull.* **1979**, *26*, 22−28.

(20) Held, M.; Karp, R. M. The Traveling-Salesman Problem and Minimum Spanning Trees. *Operat. Res.* **1970**, *18*, 1138−1162.

(21) Held, M.; Karp, R. M. The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. *Math. Program.* **1971**, *1*, 6−25.

(22) Horst, R.; Tuy, H. *Global Optimization*, 3rd ed.; Springer: Berlin, 1996.

(23) Lasdon, L. S.; Waren, A. D.; Sarkar, S.; Palacios-Gomez, F. Solving the Pooling Problem using Generalized Reduced Gradient and Successive Linear Programming Algorithms. *ACM SIGMAP Bull.* **1979**, *27*, 9−15.

(24) Liu, M. L.; Sahinidis, N. V.; Shectman, J. P. Planning of Chemical Process Networks via Global Concave Minimization. In *Global Optimization in Engineering Design*; Grossman, I. E., Ed.; Kluwer Academic Publishers: Boston, MA, 1996; pp 195−230.

(25) Lodwick, W. A. Preprocessing Nonlinear Functional Constraints with Applications to the Pooling Problem. *ORSA J. Comput.* **1992**, *4* (2), 119−131.

(26) Main, R. A. Large Recursion Models: Practical Aspects of Recursion Techniques. In *Optimization in Industry*; Ciriani, T. A., Leachman, R. C., Eds.; John Wiley & Sons Ltd.: New York, 1993.

(27) McCormick, G. P. Computability of global solutions to factorable nonconvex programs. Part I−convex underestimating problems. *Math. Program.* **1976**, *10*, 147−175.

(28) McCormick, G. P. *Nonlinear Programming. Theory, Algorithms and Applications*; Wiley Interscience: New York, 1983.

(29) Minoux, M. *Math. Program. Theory Algorithms*; John Wiley and Sons: New York, 1986.

(30) Murtagh, B. A.; Saunders, M. A. *MINOS 5.4 User's Guide*; Technical Report SOL 83-20R; Systems Optimization Laboratory, Department of Operations Research: Stanford University, CA, 1995.

(31) Quesada, I.; Grossmann, I. E. Global Optimization of Bilinear Process Networks and Multicomponent Flows. *Comput. Chem. Eng.* **1995**, *19* (12), 1219−1242.

(32) Rigby, B.; Lasdon, L. S.; Waren, A. D. The Evolution of Texaco's Blending Systems: From OMEGA to StarBlend. *Interfaces* **1995**, *25*, 64−83.

(33) Rikun, A. D. A Convex Envelope Formula for Multilinear Functions. *J. Global Optimiz.* **1997**, *10*, 425−437.

(34) Ryoo, H. S.; Sahinidis, N. V. Global Optimization of Nonconvex NLPs and MINLPs with Applications in Process Design. *Comput. Chem. Eng.* **1995**, *19*, 551−566.

(35) Ryoo, H. S.; Sahinidis, N. V. A Branch-and-Reduce Approach to Global Optimization. *J. Global Optimiz.* **1996**, *8*, 107−139.

(36) Sahinidis, N. V. BARON: A General Purpose Global Optimization Software Package. *J. Global Optimiz.* **1996**, *8*, 201−205.

(37) Sahinidis, N. V.; Grossmann, I. E. Convergence Properties of Generalized Benders Decomposition. *Comput. Chem. Eng.* **1991**, *15* (7), 481−491.

(38) Shectman, J. P.; Sahinidis, N. V. A Finite Algorithm for Global Minimization of Separable Concave Programs. *J. Global Optimiz.* **1998**, *12*, 1−36.

(39) Sherali, H. D.; Alameddine, A. A New Reformulation-Linearization Technique For Bilinear Programming Problems. *J. Global Optimiz.* **1992**, *2*, 379−410.

(40) Visweswaran, V.; Floudas, C. A. New Properties and Computational Improvement of the GOP Algorithm for Problems with Quadratic Objective Functions and Constraints. *J. Global Optimiz.* **1993**, *3*, 439−462.

(41) Visweswaran, V.; Floudas, C. A. New Formulations and Branching Strategies for the GOP Algorithm. In *Global Optimization in Engineering Design*; Grossmann, I. E., Ed.; Kluwer Academic Publishers: Dordrecht, 1996.

Received for review October 19, 1998
Revised manuscript received February 18, 1999
Accepted February 18, 1999

IE980666Q