2

Optimization of Pooling Problems for Two Objectives Using the ε-Constraint Method

Haibo Zhang and Gade Pandu Rangaiah
Department of Chemical and Biomolecular Engineering,
National University of Singapore, Singapore

2.1 Introduction

Pooling and blending problems are important optimization problems in petroleum refineries with huge cost savings potential (Baker and Lasdon, 1985; Rigby *et al.*, 1995). In petroleum refineries, final products are often made by mixing flows from pools or feed streams with different sulfur content, octane number and/or density. Pooling occurs when source/feed streams are mixed in storage tanks (pools) before producing final products. The source streams can be intermediate products from different distillation units, reformers and catalytic crackers, and/or additives like ethanol, and therefore have different compositions and properties (DeWitt *et al.*, 1989). The resulting streams from pools are then dispatched to different final products to meet the specified requirements. The pools enhance the operational flexibility of the process by providing intermediate storage. On the other hand, blending is direct mixing of source/feed streams into final products, and hence blending problems do not involve pools (i.e., storage tanks). Pooling problems are also encountered in waste-water treatment (Bagajewicz, 2000), supply-chain operations and communications (Misener and Floudas, 2009).

Pooling problems have been studied for a single objective, namely, cost minimization (i.e., profit maximization) by optimal allocation of source/feed streams to pools and then

Multi-Objective Optimization in Chemical Engineering: Developments and Applications, First Edition.
Edited by Gade Pandu Rangaiah and Adrián Bonilla-Petriciolet.

blending streams from pools to final products, subject to constraints arising from mass balances, quality balances, feed availability and product requirements. Because of the nonlinearities and nonconvexities in the optimization problem, a global optimization algorithm is required for solving pooling problems.

Many studies have focused on the solution of pooling problems, starting from Haverly (1978), who used recursive linear programming. Next, Lasdon *et al.* (1979) applied successive linear programming for solving pooling problem. Later, Baker and Lasdon (1985) applied successive linear programming to improve blending schemes at Exxon Company. The solution of pooling problems led to annual savings of more than 30 million dollars in the 1980s (DeWitt *et al.*, 1989). Reviews of global optimization methods and their applications to pooling problems can be found in Floudas and Gounaris (2009) and Misener and Floudas (2009). All these attempts to solve pooling problems have used deterministic algorithms for single objectives only.

In general, many real-world problems involve several conflicting objectives. It is often difficult to formulate these problems into a single-objective optimization (SOO) problem. The conflicting objectives in multi-objective optimization (MOO) problems lead to a set of optimal solutions called Pareto-optimal solutions, which are equally good for the specified objectives. These solutions provide better understanding of the tradeoff among objectives and many choices to the decision-maker for choosing one of them for implementation. Over the last two decades, MOO field has grown significantly and many chemical engineering applications of it have been reported (Cheah and Rangaiah, 2009; Jaimes and Coello Coello, 2009; Masuduzzaman and Rangaiah, 2009; Chapter 3 in this book). One approach to solving a MOO problem is to transform it into a single-objective problem that can be solved using a suitable SOO method. For example, in the ε-constraint method, the MOO problem is converted into a SOO problem for one primary objective and the other objectives are treated as constraints.

Khosla *et al.* (2007) have studied fuel oil blending for two and three objectives such as profit, quality give-away, production rate, use of light products and calorific value, using the elitist nondominant sorting genetic algorithm (NSGA-II) with and without jumping-gene adaptations. However, until now, pooling problems have not been optimized for multiple objectives. Hence, in this chapter, pooling problems are optimized for two objectives using the ε-constraint method. Another purpose of this chapter is to illustrate the ε-constraint method. The resulting SOO problem is solved using a recent stochastic global optimization algorithm, namely, integrated differential evolution (IDE) (Zhang and Rangaiah, 2012). Many benchmark pooling problems from the literature are optimized for two objectives: cost (or profit) and product quality, which conflict. The results demonstrate the usefulness of the ε-constraint method and IDE for the MOO of pooling problems.

Comparison of several deterministic and stochastic global optimization algorithms has been discussed in the literature (Nocedal and Wright, 2006; Srinivas and Rangaiah, 2006; Weise 2008; Mashinchi *et al.*, 2011; Exler *et al.*, 2008). In general, stochastic methods are more robust, require little or no assumption on the characteristics of the optimization problem, and yet provide a high probabilistic convergence to the global optimum. Further, they are usually simple in principle and easy to implement and use. Hence, in this chapter, IDE is used for finding the global optimum of the SOO problem.

In pooling problems, objective functions and/or constraints are nonlinear with respect to decision variables; they are also non-convex and have multiple optima. These characteristics

and the presence of many equality and inequality constraints add to the difficulty of finding the global optimum. Mathematical formulation of pooling problems can mitigate these difficulties by changing nonlinear terms, size of the search space and/or number of equality/inequality constraints. In this chapter, we describe and use an improved formulation named, *r*-formulation, for pooling problems.

The remainder of this chapter is organized as follows. Description and formulation of pooling problems is presented in section 2.2 by considering typical problems. Section 2.3 describes the ε-constraint method including an outline of the IDE algorithm. The application of this method to pooling problems is described in section 2.4. The results are presented and discussed in section 2.5. Finally, section 2.6 concludes this chapter with a summary of the findings.

2.2 Pooling Problem Description and Formulations

2.2.1 p-Formulation

The bilinear programming formulation, known as the *p*-formulation, was formulated by Haverly (1978), who also studied the solution of pooling problems using recursive linear programming. Nomenclature for the *p*-formulation of a pooling problem network is presented in Figure 2.1(a); for illustration, one of the benchmark pooling problems, Ben-Tal 4 (Ben-Tal *et al.*, 1994) is shown in Figure 2.1(b).

The source/feed streams with flow rate ($X_{i,j}$ from i^{th} source stream to j^{th} pool), quality ($\lambda_{i,w}$ for w quality of i^{th} source stream) and price (C_i of i^{th} source stream), from process units are mixed in one or more pools (Figure 2.1). There can be an upper limit on the availability of i^{th} source stream (A_i^U in Figure 2.1a) and/or more than one quality variable. Streams from the pools with flow rate ($Y_{j,k}$ from j^{th} pool to k^{th} product) and quality ($\mu_{j,w}$) are mixed again in the product tanks. The final products from the pooling network should satisfy the demand (D_k^U) and product quality ($\eta_{k,w}$), and their selling price is denoted S_k. The objective function is the difference between the cost of source/feed streams and the revenue from selling the final products. The constraints are mainly from mass balances about the pools, quality balances about the pools, raw material availability, product demands and quality requirements. Details of the *p*-formulation of pooling problems can be found in Misener and Floudas (2009).

For example, BT-4 problem shown in Figure 2.1(b) involves four source streams, two pools and two products. Three of the four source streams are mixed in one pool. The second pool has only one source stream, which is referred as bypass stream—it may go through a pool or mixed directly to the products. All streams going from this pool to the products will have the same qualities as its single source stream, and so the presence or absence of a pool does not affect the problem formulation and results except for introducing an additional *X* variable. Considering a pool for each bypass stream is helpful for the general problem formulation. Hence, a pool is considered for each bypass stream in this chapter (see Figures 2.1b and 2.3). Data shown in Figure 2.1(b) follow the symbols in Figure 2.1(a). For example, (6, 3) on top of source stream 1 are the price and quality of the source stream 1, (9, 2.5) on top of product stream 1 are the price and quality requirement of the product stream, and (100) below the product stream 1 is the upper bound on the demand for the product stream 1. Note that (50) below the source stream 2 is the availability of this stream.

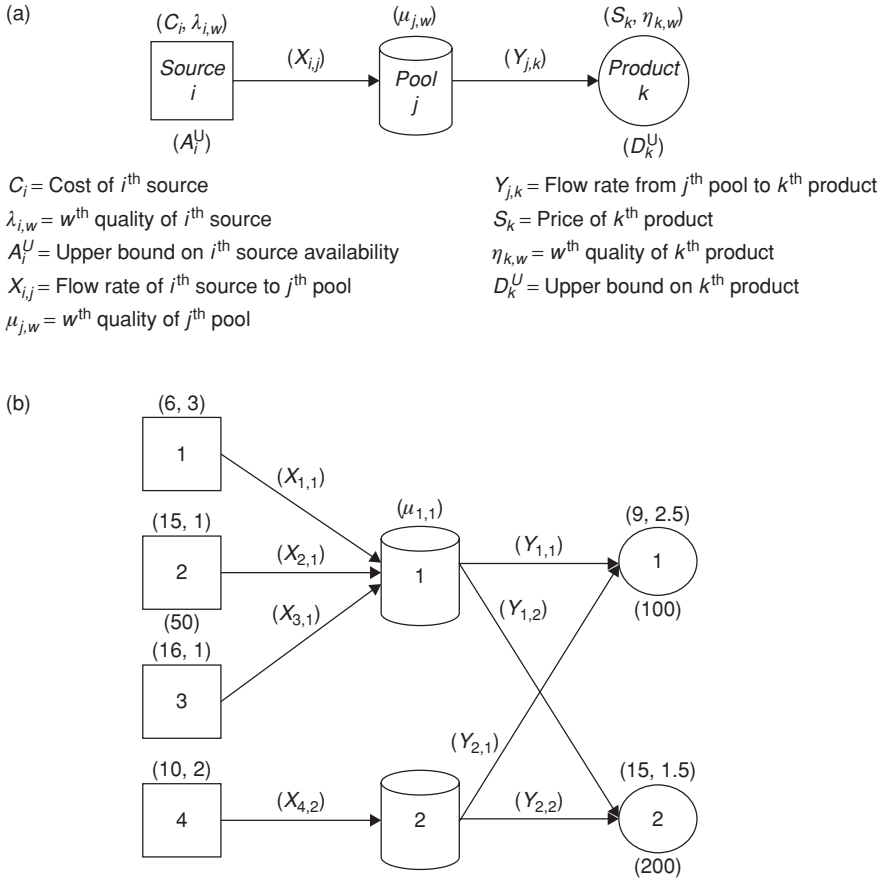

Figure 2.1 *(a) Schematic showing the general notation for the p-formulation. (b) Pooling network with variables for the p-formulation: Ben-Tal4 problem.*

For the Ben-Tal 4 problem, the SOO problem using *p*-formulation is as follows:

$$\min \underbrace{\{6X_{11} + 15X_{21} + 16X_{31} + 10X_{42}\}}_{\text{cost}} - \underbrace{\{9(Y_{11} + Y_{21}) + 15(Y_{12} + Y_{22})\}}_{\text{revenue}} \tag{2.1a}$$

$$\left.\begin{array}{ll} s.t. & X_{11} + X_{21} + X_{31} = Y_{11} + Y_{12} \\ & X_{42} = Y_{21} + Y_{22} \end{array}\right\} \textit{Mass balance around pools} \tag{2.1b}$$

$$\left.\begin{array}{l} 3X_{11} + 1X_{21} + 1X_{31} = \mu_{11}(Y_{11} + Y_{12}) \\ 2X_{42} = 2(Y_{21} + Y_{22}) \end{array}\right\} \textit{Quality balance around pools} \tag{2.1c}$$

$$\left.\begin{array}{l} \mu_{11}Y_{11} + 2Y_{21} \le 2.5(Y_{11} + Y_{21}) \\ \mu_{11}Y_{12} + 2Y_{22} \le 1.5(Y_{12} + Y_{22}) \end{array}\right\} \textit{Quality requirments of products} \tag{2.1d}$$

$$\left.\begin{array}{l} Y_{11} + Y_{21} \le 100 \\ Y_{12} + Y_{22} \le 200 \end{array}\right\} \textit{Product demands} \tag{2.1e}$$

$$X_{21} \le 50 \quad \textit{Availability of the source stream } 2 \tag{2.1f}$$

Decision variables and their values at the global optimum for the Ben-Tal 4 problem are: $X_{11} = 0$, $X_{21} = 50$, $X_{31} = 50$, $X_{42} = 100$, $Y_{11} = 0$, $Y_{21} = 0$, $Y_{12} = 100$, $Y_{22} = 100$ and $\mu_{11} = 1$ with objective function $= \{6 \times 0 + 15 \times 50 + 16 \times 50 + 10 \times 100\} - \{9 \times (0 + 0) + 15 \times (100 + 100)\} = -450$.

2.2.2 r-Formulation

The new formulation, named, the *r*-formulation, reduces the number of decision variables, constraints and search space in the optimization problem. In this formulation, new decision variables, $r_{i,j} \in (0, 1)$ and $R_{j,k} \in (0, 1)$ are introduced instead of flow rates, $X_{i,j}$ and $Y_{j,k}$. The new decision variables, $R_{j,k}$ can be used to calculate flow rates, $Y_{j,k}$ as follows.

$$Y_{1,k} = R_{1,k} D_k^U \quad k = 1, 2, \ldots, K \tag{2.2}$$

$$Y_{j,k} = R_{j,k} \left(D_k^U - \sum_{m=1}^{j-1} Y_{m,k} \right) \quad j = 2, 3, \ldots, P; \quad k = 1, 2, \ldots, K \tag{2.3}$$

In the above, K and P are, respectively, the number of products and the number of pools in the pooling network. Thus, $R_{j,k}$ represents the fraction of the maximum possible value for $Y_{j,k}$. In other words, the upper bound of the product demand and all previous $Y_{j,k}$ values are taken into account (see Equations 2.2 and 2.3). Thus, use of $R_{j,k}$ in the *r*-formulation reduces the search space, the number of decision variables and constraints.

Similarly, new decision variables, $r_{i,j} \in (0, 1)$ are used to calculate source stream flow rates, $X_{i,j}$ via $q_{i,j}$, each of which represents the fraction of total flow through j^{th} pool that comes from i^{th} source stream. For this, $q_{i,j}$ are calculated using $r_{i,j}$ as follows:

$$q_{1,j} = r_{1,j} \quad j = 1, 2, \ldots, P \tag{2.4}$$

$$q_{i,j} = r_{i,j} \left(1 - \sum_{m=1}^{i-1} q_{m,j} \right) \quad i = 2, 3, \ldots, N-1; \quad j = 1, 2, \ldots, P \tag{2.5}$$

Here, N is the total number of source/feed streams. Equation 2.5 takes into account the previous $q_{i,j}$ values calculated from $r_{i,j}$. The last $q_{N,j}$ is calculated from:

$$\sum_{m=1}^{N} q_{m,j} = 1 \quad j = 1, 2, \ldots, P \tag{2.6a}$$

$$\text{or} \qquad q_{N,j} = 1 - \sum_{m=1}^{N-1} q_{m,j} \quad j = 1, 2, \ldots, P \tag{2.6b}$$

This equation eliminates the equality constraints arising from the mass balances for pools. The source/feed flow rates are then given by

$$X_{i,j} = q_{i,j} \sum_{k=1}^{K} Y_{j,k} \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, P \tag{2.7}$$

Similar to $R_{j,k}$, the use of $r_{i,j}$ in the *r*-formulation reduces the search space, number of decision variables and constraints.

The optimization problem using *r*-formulation is simplified to as follows. Objective function:

$$\min \sum_{i=1}^{N}\sum_{j=1}^{P}\sum_{k=1}^{K}(C_i q_{i,j} Y_{j,k}) - \sum_{k=1}^{K} S_k \left(\sum_{j=1}^{P} Y_{j,k}\right) \tag{2.8}$$

Subject to

$$\left.\sum_{i=1}^{N}\sum_{j=1}^{P}\lambda_{i,w} q_{i,j} Y_{j,k} - \eta_{k,w}\sum_{j=1}^{P} Y_{j,k} \le 0, k = 1, 2, \ldots, K; \; w = 1, 2, \ldots, W\right\} \text{ Product Qualities} \tag{2.9}$$

$$\left.\sum_{j=1}^{P}\sum_{k=1}^{K} q_{i,j} Y_{j,k} - A_i^U \le 0, \quad i = 1, 2, \ldots, N\right\} \text{ Feed Availability} \tag{2.10}$$

Note that $q_{i,j}$ and $Y_{j,k}$ in the above equations are calculated using equations 2.4–2.6 and 2.2–2.3 respectively. Decision variables and their bounds in the optimization problem are:

$$0 \le r_{i,j} \le 1 \quad i = 1, 2, \ldots, N-1 \quad j = 1, 2, \ldots, P \tag{2.11}$$

$$0 \le R_{j,k} \le 1 \quad j = 1, 2, \ldots, P \quad k = 1, 2, \ldots, K \tag{2.12}$$

Thus, the optimization problem using *r*-formulation has only two sets of inequality constraints (Equations 2.9 and 2.10), and the decision variables are normalized between 0 and 1 (Equations 2.11 and 2.12). Depending on the interconnections in the pooling network, suitable values (either 0 or 1) can be assigned to some $r_{i,j}$ and $R_{j,k}$ in advance. This will further reduce the number of decision variables.

Figure 2.2 shows a schematic of Ben-Tal 4 problem using $r_{i,j}$ and $R_{j,k}$ for *r*-formulation. The SOO problem for this pooling problem using *r*-formulation is as follows. Decision

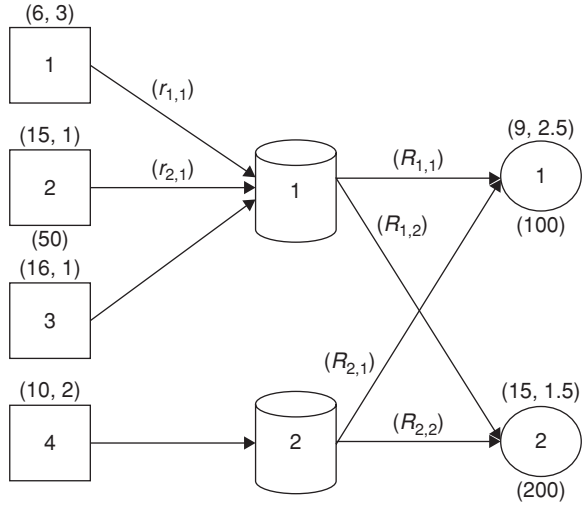

Figure 2.2 *Pooling network with variables for the r-formulation: Ben-Tal4 problem.*

variables in this problem are r's and R's; and q, X and Y are given by the explicit equations (2.2)–(2.7). So, the latter are dependent variables calculated using the values of decision variables, r's and R's. Explicit equations for calculating the dependent variables using decision variables: r_{11}, r_{21}, R_{11}, R_{12}, R_{21} and R_{22}, are:

$$\begin{aligned} &Y_{11} = R_{11} \times D_1^U = 100R_{11}, Y_{21} = R_{21} \times (D_1^U - Y_{11}) = R_{21} \times (100 - Y_{11}) \\ &Y_{12} = R_{12} \times D_2^U = 200R_{12}, Y_{22} = R_{22} \times (D_2^U - Y_{12}) = R_{22} \times (200 - Y_{12}) \end{aligned} \tag{2.13a}$$

$$q_{11} = r_{11}, q_{21} = r_{21} \times (1 - q_{11}), q_{31} = 1 - q_{11} - q_{21} \tag{2.13b}$$

$$\begin{aligned} &X_{11} = (Y_{11} + Y_{12}) \times q_{11}, X_{21} = (Y_{11} + Y_{12}) \times q_{21} \\ &X_{31} = (Y_{11} + Y_{12}) \times q_{31}, X_{42} = (Y_{21} + Y_{22}) \end{aligned} \tag{2.13c}$$

The objective function and constraints for Ben-Tal 4 problem using r-formulation are:

$$\min \quad \underbrace{\{6X_{11} + 15X_{21} + 16X_{31} + 10X_{42}\}}_{\text{cost}} - \underbrace{\{9(Y_{11} + Y_{21}) + 15(Y_{12} + Y_{22})\}}_{\text{revenue}} \tag{2.13d}$$

$$\left.\begin{aligned} &(3q_{11} + 1q_{21} + 1q_{31})Y_{11} + 2Y_{21} \le 2.5(Y_{11} + Y_{21}) \\ &(3q_{11} + 1q_{21} + 1q_{31})Y_{12} + 2Y_{22} \le 1.5(Y_{12} + Y_{22}) \end{aligned}\right\} \begin{array}{l} \textit{Quality requirements} \\ \textit{of products} \end{array} \tag{2.13e}$$

$$X_{21} \le 50 \quad \textit{Availability of the source stream } 2 \tag{2.13f}$$

Values of decision variables at the global optimum of the Ben-Tal 4 problem are: $r_{11} = 0$, $r_{21} = 0.5$, $R_{11} = 0$, $R_{12} = 0.5$, $R_{21} = 0$ and $R_{22} = 1$. Flow rates in the pooling network (i.e., dependent variables) can be calculated easily using Equations 2.13a–2.13c.

$$\begin{aligned} &Y_{11} = R_{11} \times D_1^U = 0 \times 100 = 0; Y_{21} = R_{21} \times (D_1^U - Y_{11}) = 0 \times (100 - 0) = 0; \\ &Y_{12} = R_{12} \times D_1^U = 0.5 \times 200 = 100; Y_{22} = R_{22} \times (D_1^U - Y_{12}) = 1 \times (200 - 100) = 100; \\ &q_{11} = r_{11} = 0; q_{21} = r_{21}(1 - q_{11}) = 0.5; q_{31} = 1 - q_{11} - q_{21} = 0.5 \\ &X_{11} = (Y_{11} + Y_{12}) \times q_{11} = (0 + 100) \times 0 = 0; X_{21} = (Y_{11} + Y_{12}) \times q_{21} = (0 + 100) \\ &\qquad \times 0.5 = 50; X_{31} = (Y_{11} + Y_{12}) \times q_{31} = 100 \times 0.5 = 50; X_{42} = (Y_{21} + Y_{22}) = 100; \end{aligned}$$

The objective function (cost) is

$$\{6 \times 0 + 15 \times 50 + 16 \times 50 + 10 \times 100\} - \{9 \times (0 + 0) + 15 \times (100 + 100)\} = -450$$

The above values are the same as those given at the end of section 2.2.1.

Another more complicated pooling problem, namely Ben-Tal 5 (Ben-Tal *et al.*, 1994) shown in Figure 2.3, is considered for illustrating r-formulation for pooling problems. This problem involves five source streams, four pools, five product streams and two qualities; one source stream is a bypass stream shown with pool 4. The data (6, 3, 1) on top of source stream 1 are the price, quality 1 and quality 2 of the source stream 1, respectively, (18, 2.5, 2) above the product stream 1 are price, quality 1 and quality 2 of product stream 1, respectively, and (100) below the product stream 1 is the upper limit on the demand for product stream 1. For clarity, only the decision variables at pools 1 and 4 are shown in Figure 2.3.

As before, decision variables are r's and R's in r-formulation (see Table 2.1). Explicit equations for calculating the dependent variables: q, X and Y are given by generic

Table 2.1 *Decision variables, dependent variables and their optimal values for Ben-Tal 5 problem.*

Decision variables (with optimal values in brackets)	Dependent variables (with optimal values in brackets)
r_{11} (0.0); r_{21} (0.0); r_{31} (0.0)	q_{11} (0.0); q_{21} (0.0); q_{31} (0.0); q_{41} (1.0)
r_{12} (1.0); r_{22} (0.05); r_{32} (0.0)	q_{12} (1.0); q_{22} (0.0); q_{32} (0.0); q_{42} (0.0)
r_{13} (1.0); r_{23} (0.0); r_{33} (0.005)	q_{13} (1.0); q_{23} (0.0); q_{33} (0.0); q_{43} (0.0)
R_{11} (0.33); R_{12} (1.00); R_{13} (0.67); R_{14} (0.67); R_{15} (0.67)	Y_{11} (33.14); Y_{12} (200.00); Y_{13} (66.67); Y_{14} (66.67); Y_{15} (66.67); X_{11} (0.00); X_{21} (0.00); X_{31} (0.00); X_{41} (433.14)
R_{21} (0.98); R_{22} (0.73); R_{23} (1.00); R_{24} (0.67); R_{25} (1.00)	Y_{21} (65.47); Y_{22} (0.0); Y_{23} (33.33); Y_{24} (22.41); Y_{25} (33.33); X_{12} (154.55); X_{22} (0.0); X_{32} (0.0); X_{42} (0.0)
R_{31} (0.79); R_{32} (0.98); R_{33} (0.94); R_{34} (1.00); R_{35} (0.85)	Y_{31} (1.1); Y_{32} (0.0); Y_{33} (0.0); Y_{34} (10.93); Y_{35} (0.0); X_{13} (12.02); X_{23} (0.0); X_{33} (0.0); X_{43} (0.0)
R_{41} (1.00); R_{42} (0.91); R_{43} (1.00); R_{44} (1.00); R_{45} (1.00)	Y_{41} (0.29); Y_{42} (0.0); Y_{43} (0.0); Y_{44} (0.0); Y_{45} (0.0); X_{54} (0.29)

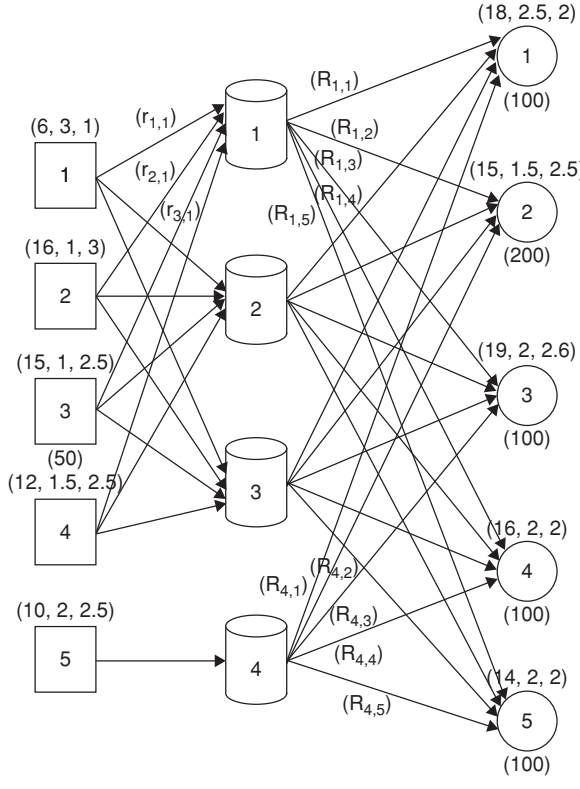

Figure 2.3 *Pooling network with variables for the* r*-formulation: Ben-Tal 5 problem; for clarity, only some decision variables are shown.*

Equations 2.2–2.7, and are not shown here for conciseness but they can be found in the Excel file on the accompanying web site for the book at http://booksupport.wiley.com. The objective function and constraints are:

$$\min \underbrace{\left\{\begin{array}{l} 6(X_{11}+X_{12}+X_{13})+16(X_{21}+X_{22}+X_{23}) \\ +15(X_{31}+X_{32}+X_{33})+12(X_{41}+X_{42}+X_{43})+10X_{54} \end{array}\right\}}_{\text{cost}} \\ - \underbrace{\left\{\begin{array}{l} 18(Y_{11}+Y_{21}+Y_{31}+Y_{41})+15(Y_{12}+Y_{22}+Y_{32}+Y_{42}) \\ +19(Y_{13}+Y_{23}+Y_{33}+Y_{43})+16(Y_{14}+Y_{24}+Y_{34}+Y_{44}) \\ +14(Y_{15}+Y_{25}+Y_{35}+Y_{45}) \end{array}\right\}}_{\text{revenue}} \tag{2.14a}$$

$$s.t. \quad \left.\begin{array}{l} (3q_{11}+1q_{21}+1q_{31}+1.5q_{41})Y_{11}+(3q_{12}+1q_{22}+1q_{32}+1.5q_{42})Y_{21} \\ +(3q_{13}+1q_{23}+1q_{33}+1.5q_{43})Y_{31}+2Y_{41} \le 2.5(Y_{11}+Y_{21}+Y_{31}+Y_{41}) \\ (3q_{11}+1q_{21}+1q_{31}+1.5q_{41})Y_{12}+(3q_{12}+1q_{22}+1q_{32}+1.5q_{42})Y_{22} \\ +(3q_{13}+1q_{23}+1q_{33}+1.5q_{43})Y_{32}+2Y_{42} \le 1.5(Y_{12}+Y_{22}+Y_{32}+Y_{42}) \\ (3q_{11}+1q_{21}+1q_{31}+1.5q_{41})Y_{13}+(3q_{12}+1q_{22}+1q_{32}+1.5q_{42})Y_{23} \\ +(3q_{13}+1q_{23}+1q_{33}+1.5q_{43})Y_{33}+2Y_{43} \le 2(Y_{13}+Y_{23}+Y_{33}+Y_{43}) \\ (3q_{11}+1q_{21}+1q_{31}+1.5q_{41})Y_{14}+(3q_{12}+1q_{22}+1q_{32}+1.5q_{42})Y_{24} \\ +(3q_{13}+1q_{23}+1q_{33}+1.5q_{43})Y_{34}+2Y_{44} \le 2(Y_{14}+Y_{24}+Y_{34}+Y_{44}) \\ (3q_{11}+1q_{21}+1q_{31}+1.5q_{41})Y_{15}+(3q_{12}+1q_{22}+1q_{32}+1.5q_{42})Y_{25} \\ +(3q_{13}+1q_{23}+1q_{33}+1.5q_{43})Y_{35}+2Y_{45} \le 2(Y_{15}+Y_{25}+Y_{35}+Y_{45}) \end{array}\right\} \textit{Quality 1 requirments of products} \tag{2.14b}$$

$$\left.\begin{array}{l} (1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{11}+(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{21} \\ +(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{31}+2.5Y_{41} \le 2(Y_{11}+Y_{21}+Y_{31}+Y_{41}) \\ (1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{12}+(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{22} \\ +(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{32}+2.5Y_{42} \le 2.5(Y_{12}+Y_{22}+Y_{32}+Y_{42}) \\ (1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{13}+(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{23} \\ +(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{33}+2.5Y_{43} \le 2.6(Y_{13}+Y_{23}+Y_{33}+Y_{43}) \\ (1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{14}+(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{24} \\ +(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{34}+2.5Y_{44} \le 2(Y_{14}+Y_{24}+Y_{34}+Y_{44}) \\ (1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{15}+(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{25} \\ +(1q_{11}+3q_{21}+2.5q_{31}+2.5q_{41})Y_{35}+2.5Y_{45} \le 2(Y_{15}+Y_{25}+Y_{35}+Y_{45}) \end{array}\right\} \textit{Quality 2 requirments of products} \tag{2.14c}$$

$$X_{31}+X_{32}+X_{33} \le 50 \quad \textit{Availability of the source stream 3} \tag{2.14d}$$

The first set of constraints (Equation 2.14b) is for quality 1 requirements on products, and the second set of constraints (Equation 2.14c) is for quality 2 requirements on products. The third constraint (Equation 2.14d) is the availability of the source stream 3 only. The Ben-Tal 5 problem has a multiple global minimum of −3500; the values of decision variables and dependent variables at one global minimum are given in Table 2.1.

2.3 *ε*-Constraint Method and IDE Algorithm

Previous studies on pooling problems have all focused on SOO. In this chapter, pooling problems are solved for two objectives to find Pareto-optimal solutions that provide insights into tradeoffs between objectives and many optimal choices for the decision-maker.

A classical approach to solving MOO problems is by transforming the multi-objective problem into a sequence of parameterized single-objective problems such that the solution of each single-objective problem corresponds to one Pareto-optimal solution. The common transformation method is the ε-constraint method, proposed by Haimes *et al.* (1971). It is based on scalarization where one of the objective functions is optimized while treating the other objectives as inequality constraints in order to keep them within reasonable limits.

Consider the bi-objective optimization problem:

$$\text{Objective 1: Min. } f_1(\boldsymbol{x}) \tag{2.15a}$$

$$\text{Objective 2: Min. } f_2(\boldsymbol{x}) \tag{2.15b}$$

$$\text{Subject to: } g(\boldsymbol{x}) \leq 0 \tag{2.15c}$$

$$h(\boldsymbol{x}) = 0 \tag{2.15d}$$

where "$\boldsymbol{x}$" is the set of decision variables, and $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are the inequality and equality constraints, respectively. For solving this bi-objective problem by the ε-constraint method, one objective—say, $f_2(\boldsymbol{x})$—is converted into an inequality constraint as follows:

$$\text{Objective: Min. } f_1(\boldsymbol{x}) \tag{2.16a}$$

$$\text{Subject to: } g(\boldsymbol{x}) \leq 0 \tag{2.16b}$$

$$h(\boldsymbol{x}) = 0 \tag{2.16c}$$

$$f_2(\boldsymbol{x}) \leq \varepsilon \tag{2.16d}$$

In the above, ε is a suitable value in order to keep the objective $f_2(\boldsymbol{x})$ small enough. The solution of the above single-objective problem for each ε value gives one Pareto-optimal solution. A series of Pareto-optimal solutions can be obtained by varying ε from a large to a small value. The large value for ε can be based on the solution of the single-objective problem in Equation 2.16 without the constraint on $f_2(\boldsymbol{x})$, and the small value for ε is the minimum of $f_2(\boldsymbol{x})$ subject to $g(\boldsymbol{x}) \leq 0$ and $h(\boldsymbol{x}) = 0$ (i.e., ignoring $f_1(\boldsymbol{x})$), which can be obtained by SOO. Alternatively, the range of ε can be based on the physical knowledge of the optimization problem; this strategy is used for bi-objective optimization of pooling problems in the next section.

The concept of an ε-constraint method is simple, and one can use one of many available SOO techniques (e.g., the Solver tool provided in the MS-Excel file; see Lee *et al.*, 2008). However, the method requires the solution of the SOO problem numerous times, particularly for more than two objectives. The additional constraints introduced for the objectives can make the SOO problem very difficult to solve. Further, Pareto-optimal solutions obtained from numerous SOO problems may not be well distributed. Despite these limitations, the ε-constraint method is used in this chapter for illustration, because it is a classical approach and popular for bi-objective problems. Here, SOO problems arising in the ε-constraint method are solved using the stochastic global optimization method, namely, integrated differential evolution (IDE) (Zhang and Rangaiah, 2012). The IDE algorithm integrates several strategies with differential evolution (DE), proposed by Storn and Price (1997). Differential evolution is a population-based global optimization method, its principle and steps are easy to understand, and it has relatively fast convergence and high reliability in finding the global optimum (Price *et al.*, 2005; Srinivas and Rangaiah, 2006).

The IDE algorithm incorporates the following strategies: (i) a self-adaptive strategy and learning to tune values of algorithm parameters and select a suitable mutation strategy; (ii) the tabu concept and a list of tabu searches to prevent revisiting the same search area and increasing the population diversity; (iii) self-adaptive constraint handling method to handle equality and inequality constraints; (iv) a novel stopping criterion based on the number of rejections; and (v) local optimization after the global search to find the precise optimum efficiently. The better reliability and effectiveness of IDE have been shown on many challenging benchmark functions with equality and inequality constraints by comparison with other stochastic optimization methods. See Zhang and Rangaiah (2012) for more details on IDE.

2.4 Application to Pooling Problems

In this study, product qualities, Equation 2.9 of the *r*-formulation is used as the second objective function (f_2) which is transformed as an inequality constraint shown in Equation 2.17 while retaining the objective function (f_1) as the single objective function and other constraints (Equations 2.10–2.12) unchanged.

$$\sum_{i=1}^{N}\sum_{j=1}^{P}\lambda_{i,w}q_{i,j}Y_{j,k} \le \varepsilon_{k,w} \quad k = 1, 2, \ldots, K; w = 1, 2, \ldots, W \tag{2.17}$$

The f_2 is included as an inequality constraint such that its value is not more than ε at the optimal solution of the transformed problem. The SOO problem will have to be solved for a range of ε values in order to find many Pareto-optimal solutions.

In this study, range of ε values is calculated based on the quality of the products at the global optimum of the original pooling problem with single objective of cost and using the following equation.

$$\varepsilon_{k,w} = \left(\eta_{k,w}\sum_{j=1}^{P}Y_{j,k}\right) \times Q_R \quad k = 1, 2, \ldots, K; w = 1, 2, \ldots, W \tag{2.18}$$

Here, Q_R is the ratio of the product quality. $Q_R = 1$ corresponds to the original pooling problem with a single objective, $Q_R > 1$ implies lower product quality, and $Q_R < 1$ refers to better product quality. Since the overall cost of the process increases with product quality, these two objective functions are conflicting and one can expect many Pareto-optimal solutions. In this study, the range of Q_R is set from 0.8 to 2.0 after preliminary testing on selected pooling problems.

The ε-constraint method with IDE for solving the resulting SOO problems is employed for optimizing 13 benchmark pooling problems for cost and product-quality objectives. In these problems employing *r*-formulation, the number of variables is in the range 5 to 152, and the number of inequality constraints is 2 to 24. Table 2.2 summarizes the basic details of these pooling problems with single and multiple qualities. The global optimum for each problem is also given in Table 2.2, which is the same solution when Q_R is equal to 1 in MOO approach.

Table 2.2 *Basic details of 13 benchmark pooling problems used for MOO; see Adhya* et al. *(1999) for further details.*

Problem name	Input streams	Pools + bypasses	Products	Quality	Decision variables[a]	Inequality constraints[a]	Global optimum
Haverly 1	3	1 + 1	2	1	5	2	−400
Haverly 2	3	1 + 1	2	1	5	2	−600
Haverly 3	3	1 + 1	2	1	5	2	−750
Foulds 2	6	2 + 2	4	1	18	4	−1100
Foulds 3	11	8 + 0	16	1	152	16	−8
Foulds 4	11	8 + 0	16	1	152	16	−8
Foulds 5	11	4 + 0	16	1	92	16	−8
Ben-Tal 4	4	1 + 1	2	1	6	3	−450
Ben-Tal 5	5	3 + 1	5	2	29	11	−3500
Adhya 1	5	2 + 0	4	4	11	16	−549.8
Adhya 2	5	2 + 0	4	6	11	24	−549.8
Adhya 3	8	3 + 0	4	6	17	24	−561.05
Adhya 4	8	2 + 0	5	4	16	20	−877.65

[a]Number of decision variables and number of inequality constraints are for *r*-formulation.

For solving MOO of pooling problems by the ε-constraint method, the following parameter values were used in IDE: population size, NP = 50, learning period, LP = 50, tabu list size = 50 and tabu radius= min (0.7, 0.02×Number of Variables). Stopping criterion is the satisfaction of either the maximum number of rejections, $N_{max} = 8$ or maximum number of function evaluations, $G_{max} = 10000 \times$ number of variables. The latter is used as another stopping criterion to avoid infinite loops. The local optimizer used in IDE is "fmincon" from the MATLAB toolbox. These parameter values are selected from those in Zhang and Rangaiah (2012) and are based on preliminary testing.

For each Q_R value, the SOO problem is solved 20 times using IDE, in order to ensure that the global optimum is obtained; the best results obtained from the 20 runs are presented and discussed in the next section. All calculations for MOO of pooling problems are performed in MATLAB. The computer system employed in this study is the Intel Core 2 (Duo CPU 3.10 GHz, 3.16 GB RAM) for which MFlops (million floating point operations per second) for the LINPACK benchmark program (at www.netlib.org/) for a matrix of order 500 are 1074 CPU time for one run of SOO by IDE varies from 3 to 20 s depending on the pooling problem.

2.5 Results and Discussion

First, the Ben-Tal 4 and 5 problems with stream data shown in Figures 2.2 and 2.3 have been optimized for loss (whose minimization is equivalent to profit maximization) and product quality simultaneously. The Ben-Tal 4 problem has four input streams, one pool, one bypass, two product streams and a single quality; it has six decision variables ($r_{i,j}$ and $R_{j,k}$; see Equation 2.13 and description below it). The Ben-Tal 5 problem has five

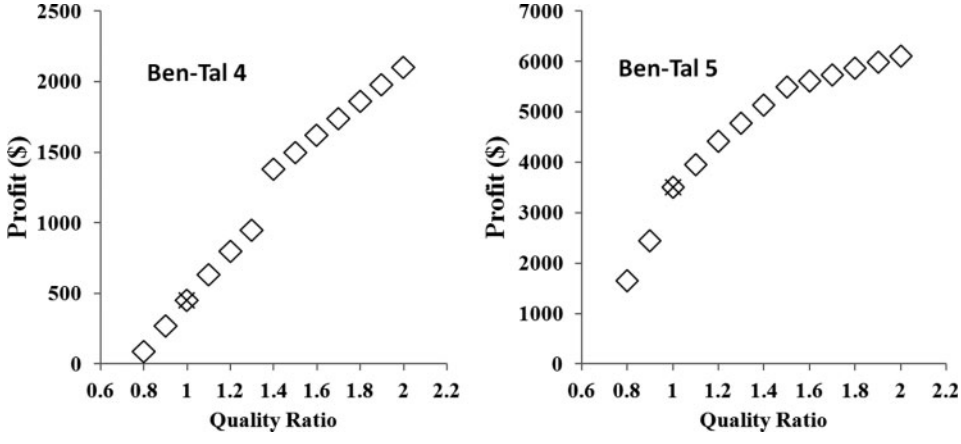

Figure 2.4 *Pareto-optimal solutions obtained by the ε-constraint method for Ben-Tal 4 and 5 problems.*

input streams, four pools, one bypass, five product streams and two qualities; it has 29 decision variables (see Table 2.1). Figure 2.4 shows the two-objective optimization results obtained for the Ben-Tal 4 and 5 problems. The *x*-axis is the second objective, namely, product quality ratio. The product quality is higher when the quality ratio is lower. For example, sulfur content in gasoline represents the gasoline quality; lower sulfur content in the gasoline means higher product quality. The *y*-axis is the first objective: profit (i.e., negative of loss) from the pooling network.

Figure 2.4 shows that the Pareto-optimal front for Ben-Tal 4 and 5 problems is well distributed except for a possible discontinuity. This indicates that the ε-constraint method requires many trials for a smooth Pareto-optimal front. As expected, profit and product quality are conflicting in nature (i.e., an increase in product quality is accompanied by a decrease in profit). For the Ben-Tal 4 problem, the solution obtained with the quality ratio at 1 is exactly the solution presented in section 2.2.2 for the SOO problem with quality constraints (Equation 2.13). This particular solution is identified with × in Figure 2.4 and subsequent figures. The profit increases linearly with the quality ratio from 0.8 to 1.3, shows a sudden jump from $950 to $1350, and then increases linearly with the quality ratio from 1.4 to 2.0. This indicates that a small decrease in product quality near the sudden jump can increase profit substantially.

Compared to the Ben-Tal 4 problem, the Ben-Tal 5 problem is more difficult to solve because of greater complexity and multiple qualities; the latter problem also has multiple global optimum solutions. For the Ben-Tal 5 problem, the profit decreases sharply from 3500 to 1653 when the quality ratio is changed from 1.0 to 0.8 (Figure 2.4). At a higher quality ratio (i.e., lower product quality), the profit increase becomes relatively small. The point "×" indicates the global optimum reported in the literature for SOO at a quality ratio of 1.

The Pareto-optimal fronts obtained by the ε-constraint method for the Haverly 1, 2 and 3 problems are well distributed (Figure 2.5). For these problems, the profit increases almost linearly as the quality ratio increases from 0.8 to 1.3. Then, as the quality ratio increases

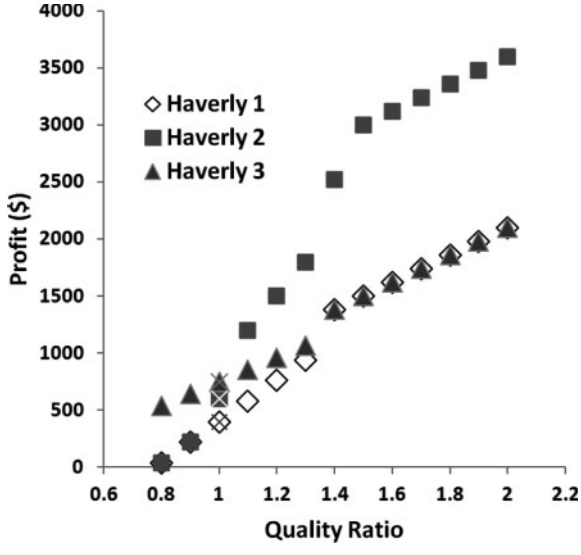

Figure 2.5 *Pareto-optimal solutions obtained by the ε-constraint method for Haverly 1, 2 and 3 problems.*

from 1.3 to 1.4, the profit increases sharply, particularly for the Haverly 2 problem. This is because the product 1 demand in Haverly 2 is higher than that in the Haverly 1 and 3 problems. Subsequently, the profit increases linearly until a it reaches a quality ratio of 2.0. For the Haverly 1 and 2 problems, the profit becomes zero when the quality ratio reduces to 0.8, whereas the profit is 540 for the Haverly 3 problem when the quality ratio is 0.8. Results in Figure 2.5 show that the Haverly 2 problem has a substantially higher profit at a lower product quality (i.e., higher quality ratio). On the other hand, the Haverly 3 problem has a higher profit at high quality. Interestingly, the Haverly1 and 3 problems have exactly the same profit for quality ratio from 1.4 to 2.0. The solutions shown as "×" in Figure 2.5 refer to those obtained by IDE when the quality ratio is equal to 1; IDE is able to obtain the global optimum solution reported in the literature for Haverly problems (Table 2.2).

When the number of decision variables and pools are larger, the pooling problem is even more difficult to solve. The ε-constraint method with IDE for SOO is applied to larger pooling problems: Foulds 2 to 5 with large numbers of pools and products and consequently a large number of decision variables, from 18 to 152, as shown in Table 2.2. Figure 2.6 shows that the Pareto-optimal front for each of Foulds problems is well distributed. For the Foulds 2 problem, the profit increases almost linearly with increasing quality ratio. This is not so for Foulds 3, 4 and 5, where the profit does not increase much for quality ratio more than 1.8. Interestingly, Foulds 3, 4 and 5 have similar Pareto-optimal fronts even though they are different in number of pools, input stream data and decision variables.

The Adhya 1 to 4 problems have multiple qualities—up to six—which indicates their complexity. The MOO results obtained for these problems by the ε-constraint method with

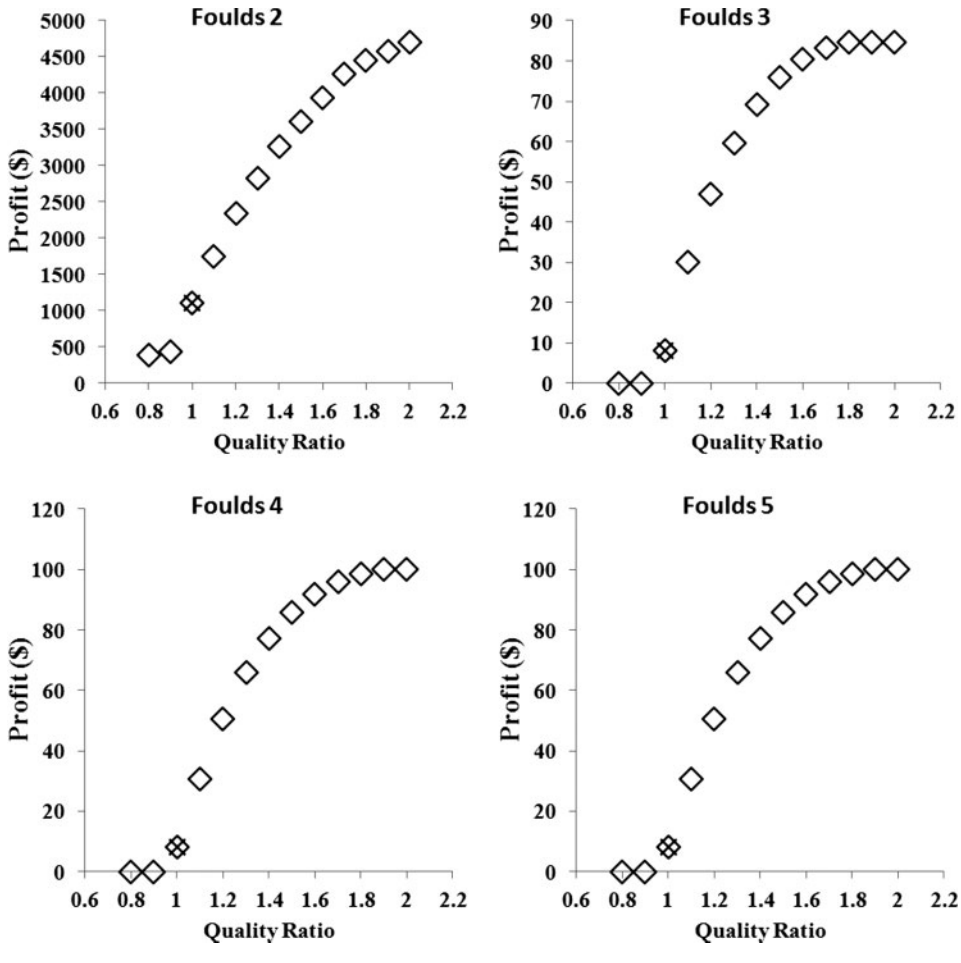

Figure 2.6 *Pareto-optimal solutions obtained by the ε-constraint method for Foulds 2, 3, 4 and 5 problems.*

IDE for SOO are shown Figure 2.7. These results indicate that a slight increase in product quality can lead to a large decrease in profit in Adhya problems. In fact, when the quality ratio is at 0.8, the profit is 0 for the Adhya 1, 2 and 3 problems. For Adhya 1, comparable profit is obtained at $837.2 and $854.2 for quality ratio of 1.2 and 1.3 respectively. Similar solutions can be seen for the Adhya 2 problem in Figure 2.7, where the profit is $615.5 and $626.3 for quality ratio of 1.1 and 1.2 respectively. For the Adhya 3 problem, the Pareto-optimal front is well distributed in the quality ratio range from 1.2 to 2.0. A somewhat different Pareto-optimal front is obtained for Adhya 4. The profit is $105 when the quality ratio is at 0.8 and 0.9, and it is $877.65 when the quality ratio is 1. Increasing the quality ratio above 1.2 does not increase profit. These trends are useful in decision making for situations where the product quality can be varied for different markets.

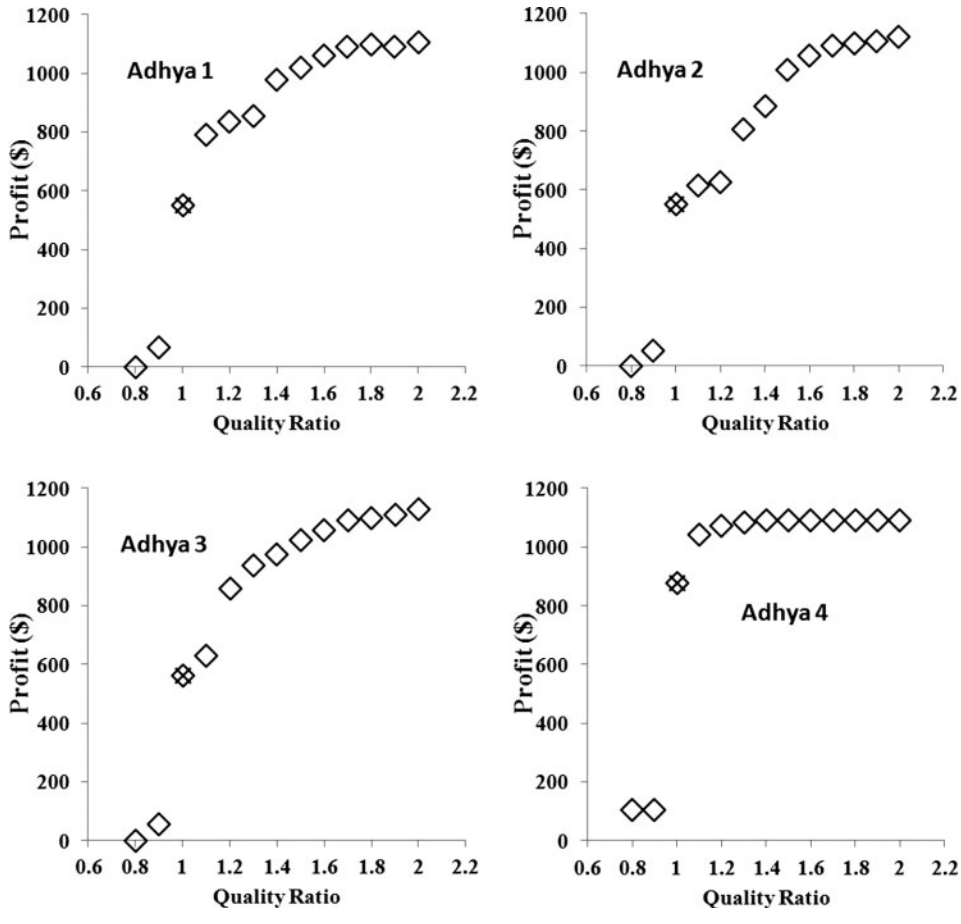

Figure 2.7 *Pareto-optimal solutions obtained by the ε-constraint method for Adhya 1, 2, 3 and 4 problems.*

2.6 Conclusions

In this chapter, pooling problems were optimized for two objectives, for the first time, using the ε-constraint method along with IDE for SOO. In addition, the *r*-formulation for pooling problems was described and used; this formulation reduces the number of decision variables and constraints as well as search space. The approach of using the ε-constraint method, IDE for SOO and the *r*-formulation for pooling problems is successful in optimizing 13 benchmark pooling problems from the literature for two objectives. The results obtained demonstrate the usefulness of MOO for understanding the tradeoff between objectives and in having many optimal solutions for selection. Given the large scale of operations in petroleum refineries, even minor improvements in solutions can lead to substantial savings in costs. In the present study, product prices are taken to be the same as those in the literature; these are likely to vary with product quality, location and time. The ε-constraint

method, IDE for SOO and the *r*-formulation for pooling problems can easily be applied to new pooling problems. It is also possible to use methods such as NSGA-II or MOO toolbox in MATLAB, which find many non-dominated solutions in a single run, instead of the ε-constraint method (see Exercise 2.3).

Exercises

2.1. For the pooling problems solved in this chapter, obtain the results presented using ε-constraint method and a global optimizer (e.g., IDE or BARON in GAMS). Confirm your results with those presented in this chapter.

2.2. IDE algorithm includes both global and local optimizers, where the global optimizer is based on DE and the local optimizer is sequential quadratic programming (SQP) method. Solve the problems in this chapter using ε-constraint method with a local optimizer only (e.g., using Solver in Excel or fmincon in MATLAB). Compare the results obtained with those in this chapter, and discuss the differences, if any.

2.3. Optimize pooling problems studied in this chapter, for two objectives using a MOO method such as NSGA-II or MOO toolbox in MATLAB. Compare and discuss the results obtained with those presented in this chapter.

2.4. For MOO in this chapter, the two objectives are profit and product quality. It is possible to choose other objectives (e.g., product demand). Will there be Pareto-optimal solutions or single optimal solution if the two objectives are profit and product demand? Briefly explain.

References

Adhya, N., Tawarmalani, M., Sahinidis, N.V., A Lagrangian approach to the pooling problem. *Ind. Eng. Chem. Res.* **1999**, 38, 1965–1972.

Bagajewicz, M. A review of recent design procedures for water networks in refineries and process plants. *Comput. & Chem. Eng.*, **2000**, 24, 2093–2113.

Baker, T.E., Lasdon, L.S., Successive linear programming at Exxon. *Manage. Sci.* **1985**, 31, 264–274.

Ben-Tal, A., Eiger, G., Gershovitz, V., Global minimization by reducing the duality gap. *Math. Program.* **1994**, 63, 193–212.

Cheah, K.S., Rangaiah, G.P., Multi-objective optimization in food engineering. In: Erdogdu, F. (ed.), *Optimization in Food Engineering.* Taylor and Francis/CRC Press, **2009**.

DeWitt, C.W., Lasdon, L.S., Waren, A.D., Brenner, D.A. and Melham, S., OMEGA: An improved gasoline blending system for Texaco. *Interfaces*, **1989**, 19 (1), 85–101.

Exler, O., Antelo, L.T., Egea, J.A., Alonso, A.A., Banga, J.R., A taboo search-based algorithm for mixed-integer nonlinear problems and its application to integrated process and control system design. *Comput. & Chem. Eng.*, **2008**, 32, 1877–1891.

Floudas, C.A., Gounaris, C.E., A review of recent advances in global optimization. *J. Glob. Optim.* **2009**, 45, 3–38.

Foulds, L.R., Haugland, D., Jornsten, K., A bilinear approach to the pooling problem. *Optimization* **1992**, 24, 165–180.

Haimes, Y., Lasdon, L., Wismer, D. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Trans. Systems, Man, and Cybernetics* **1971**, 1, 296–297.

Haverly, C.A., Studies of the behaviour of recursion for the pooling problem. *ACM SIGMAP Bull.* **1978**, 25, 29–32.

Jaimes, A.L., Coello Coello, C.A., Multi-objective evolutionary algorithms: a review of the state-of-the-art and some of their applications in chemical engineering. In Rangaiah, G.P. (ed.), *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*. World Scientific, **2009**.

Khosla, D.K., Gupta, S.K., Saraf, D.N., Multi-objective optimization of fuel oil blending using the jumping gene adaptation of genetic algorithm. *Fuel Process. Technol.*, **2007**, 88, 51–63.

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.

Lee, S.Q.E., Ang, Y.W.A, Rangaiah, G.P., Optimize your process plant for more than one objective. *Chem. Eng.*, **2008**, September, 58–64.

Mashinchi, M.H., Orgun, M.A., Pedrycz, W., Hybrid optimization with improved tabu search. *Appl. Soft Comput.* **2011**,11, 1993–2006.

Masuduzzaman, Rangaiah, G.P. Multi-objective optimization applications in chemical engineering. In Rangaiah, G.P. (ed.), *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*, World Scientific, **2009**.

Misener, R., Floudas, C.A., Advances for the pooling problem: modeling, global optimization, and computational studies. *Appl. Comput. Math.* **2009**, 8, 3–22.

Nocedal, J., Wright., S.J., *Numerical Optimization* Springer, **2006**.

Price, K., Storn, R., Lampinen, J., *Differential Evolution: A Practical Approach to Global Optimization*. Springer, **2005**.

Rangaiah, G.P. (ed.), *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*. World Scientific, **2009**.

Rigby, B., Lasdon, L.S., Waren, A.D. The evolution of Texaco's blending systems: from OMEGA to StarBlend. *Interfaces* **1995**, 25, 64–83.

Srinivas, M., Rangaiah, G.P., A study of differential evolution and tabu search for benchmark, phase equilibrium and phase stability problems. *Comput. & Chem. Eng.*, **2006**, 31, 760–772.

Storn, R., Price, K., Differential evolution – a simple and efficient heuristic for global optimization over continuous space. *J. Glob. Optim.*, **1997**, 11 (4), 341–359.

Weise, T., *Global Optimization Algorithm: Theory and Application*, **2008**, www.it-weise.de/projects/book.pdf (accessed December 4, 2012).

Zhang, H., Rangaiah, G.P., An efficient constraint handling method with integrated differential evolution for numerical and engineering optimization. *Comp. & Chem. Eng.*, **2012**, 37, 74–88.