Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 28 (2004) 1169–1192
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Retrospective on optimization

Lorenz T. Biegler*, Ignacio E. Grossmann

Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Abstract

In this paper, we provide a general classification of mathematical optimization problems, followed by a matrix of applications that shows the areas in which these problems have been typically applied in process systems engineering. We then provide a review of solution methods of the major types of optimization problems for continuous and discrete variable optimization, particularly nonlinear and mixed-integer nonlinear programming (MINLP). We also review their extensions to dynamic optimization and optimization under uncertainty. While these areas are still subject to significant research efforts, the emphasis in this paper is on major developments that have taken place over the last 25 years.

Keywords: Process optimization; Nonlinear programming; Mixed integer nonlinear programming; Linear programming; Dynamic optimization; Stochastic optimization

1. Introduction

When the two authors were asked to provide retrospective and perspective articles in the area of optimization, we decided that writing the two papers jointly would offer a better fit, given the breadth of the optimization area and our complementary interests. Our objective in this first paper is to provide a general review on optimization, emphasizing the strategies that have been applied or studied more extensively, namely, nonlinear programming (NLP), mixed-integer nonlinear programming (MINLP), dynamic optimization, and optimization under uncertainty. In the second paper we outline future directions of research that are motivated by the current barriers and limitations that are being experienced. These include global and logic-based optimization, large-scale computation, and advanced scientific computation.

Optimization has become a major enabling area in process systems engineering. It has evolved from a methodology of academic interest into a technology that has and continues to make significant impact in industry. Before we discuss the applications of optimization, it is useful to present a classification of problem types. It should be noted that this classification is independent of the solution methods. As shown in Fig. 1, optimization problems can first be classified in terms of continuous and of discrete variables. The major problems for continuous optimization include linear programming (LP) and NLP. An important subclass of LP is the linear complementarity problem (LCP), while for the NLP it includes quadratic programming (QP) and semidefinite programming (SP). For the latter, an important distinction is also whether the NLP problem is convex or nonconvex, since the latter may give rise to multiple local optima. Another important distinction is whether the problem is assumed to be differentiable or not. As for discrete problems, they are first classified into mixed-integer linear programming (MILP) and MINLP.

For the former an important particular case is when all the variables are integer, which gives rise to an integer programming (IP) problem. This problem in turn can be classified into many special problems (e.g. assignment, traveling salesman, etc.), which we do not show in Fig. 1. The MINLP problem also gives rise to special problems, although here the main distinction, like in the NLP problem, is whether its relaxation is convex or nonconvex.

Regarding their formulation, discrete/continuous optimization problems when represented in algebraic form, correspond to mixed-integer optimization problems that have the following general form:

$$\min Z = f(x, y) \text{ s.t. } \begin{cases} h(x, y) = 0 \\ g(x, y) \leq 0 \\ x \in X, y \in \{0, 1\}^m \end{cases} \quad \text{(MIP)}$$

where $f(x, y)$ is the objective function (e.g. cost), $h(x, y) = 0$ are the equations that describe the performance of the system (material balances, production rates), and $g(x, y) \leq 0$ are

* Corresponding author. Tel.: +1-412-268-2232; fax: +1-412-268-7139.
E-mail address: biegler@cmu.edu (L.T. Biegler).

0098-1354/$ – see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2003.11.003

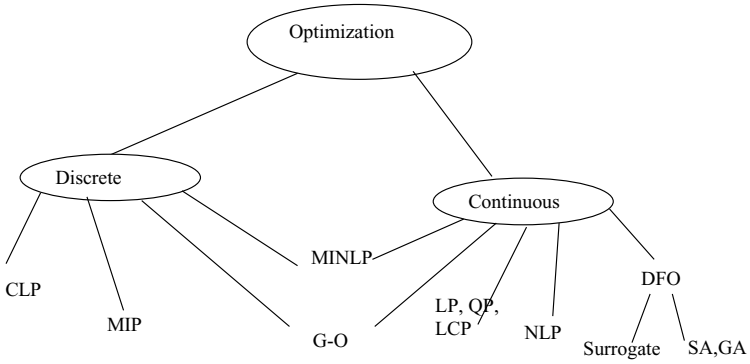

Fig. 1. Tree of classes of optimization problem.

inequalities that define the specifications or constraints for feasible plans and schedules. The variables x are continuous and generally correspond to state variables, while y are the discrete variables, which generally are restricted to take 0–1 values to define for instance the assignments of equipment and sequencing of tasks. Problem (MIP) corresponds to a mixed-integer nonlinear program (MINLP) when any of the functions involved are nonlinear. If all functions are linear it corresponds to a MILP. If there are no 0–1 variables, the problem (MIP) reduces to a NLP or LP depending on whether or not the functions are linear.

It should be noted that (MIP) problems, and their special cases, may be regarded as steady-state models. Hence, one important extension is the case of dynamic models, which in the case of discrete time models gives rise to multiperiod optimization problems, while for the case of continuous time it gives rise to optimal control problems that generally involve differential–algebraic equation (DAE) systems. Another important extension includes problems under uncertainty, which give rise to stochastic optimization problems.

1.1. Applications matrix

Mathematical programming, and optimization in general, have found extensive use in process systems engineering. A major reason for this is that in these problems there are often many alternative solutions, and hence, it is often not easy to find the optimal solution. Furthermore, in many cases, the economics is such that finding the optimum solution translates into large savings. Therefore, there might be a large economic penalty to just sticking to suboptimal solutions. In summary, optimization has become a major technology that helps companies to remain competitive.

As for specific areas, process design problems tend to give rise to NLP and MINLP problems, while scheduling and planning problems tend to give rise to LP and MILP problems. The reason for this is that design problems tend to rely more heavily on predictions of process models, which are nonlinear, while in scheduling and planning the physical predictions tend to be less important, since most operations are described through time requirements and activities. In the case of process control the split is about even.

In Table 1, we indicate what specific types of models have been formulated for a number of applications in process systems engineering. As seen in Table 1, design and synthesis have been dominated by NLP and MINLP models due to the need for the explicit handling of performance equations, although simpler targeting models give rise to LP and MILP problems. Operations problems, in contrast, tend to be dominated by linear models, LP and MILP, for planning,

Table 1
Applications of mathematical programming in process systems engineering

	LP	MILP	QP, LCP	NLP	MINLP	Global	SA/GA
Design and synthesis							
HENS	×	×		×	×	×	×
MENS	×	×		×	×	×	×
Separations		×			×		
Reactors	×			×	×	×	
Equipment Design				×	×		×
Flowsheeting				×	×		
Operations							
Scheduling	×	×			×		×
Supply chain	×	×			×		
Real-time optimization	×		×	×			
Control							
Linear MPC	×		×				
Nonlinear MPC				×		×	
Hybrid		×		×	×		

scheduling and supply chain problems. NLP, however, plays a crucial role at the level of real time optimization. Control has traditionally relied on LP and NLP models, although MILPs are being increasingly used for hybrid systems. Finally, note that global optimization has concentrated more on design than on operations problems, since nonconvexities in the design problems are more likely to yield suboptimal solutions since the corresponding bounds for the variables are rather loose in these problems. It is also worth noting that the applications listed in Table 1 have been facilitated not only by progress in optimization algorithms, but also by the advent of modeling techniques (Williams, 1985) and systems such as GAMS (Brooke, Kendrick, Meeraus, & Raman, 1998) and AMPL (Fourer, Gay, & Kernighan, 1992).

Several other papers in this special issue discuss applications of optimization in process engineering. Instead, this paper will emphasize optimization methods and concepts as a core area for research in process systems engineering. As a result, this review serves as a complement to detailed optimization models in specific applications areas that are presented in other papers in this issue. In the next section, we present an overview of linear and nonlinear programming methods for optimization problems with continuous variables, including simulated annealing (SA) and genetic algorithms (GA). Section 3 then extends to mixed-integer problems and provides a review of MINLP methods. Section 4 provides a survey of methods for optimization problems that include differential–algebraic equations and Section 5 discusses optimization under uncertainty. Finally, Section 6 provides a summary and sets the stage for future work discussed in our companion Perspectives paper.

2. Continuous variable optimization

For continuous variable optimization we consider (MIP) without discrete variables y. The general problem (NLP) is presented below:

$$\text{Min } f(x) \text{ s.t. } \begin{cases} h(x) = 0 \\ g(x) \leq 0 \end{cases} \quad \text{(NLP)}$$

A key characteristic of problem (NLP) is whether it is convex or not, i.e., it has a convex objective function and a convex feasible region. Convex feasible regions require $g(x)$ to be convex and $h(x)$ to be linear.[1] If (NLP) is a convex problem, than any local solution is also a global solution to (NLP). Moreover, if the objective function is strictly convex, this solution is unique. On the other hand, the KKT conditions can only satisfy local optimality for nonconvex problems and, as discussed in our companion perspectives paper, a more rigorous (and expensive) search procedure is required to find a global solution.

Further specializations of the problem can be made if the constraint and objective functions satisfy certain properties, and specialized algorithms can be constructed for these cases. In particular if the objective and constraint functions in (NLP) are linear then the following linear program:

$$\text{Min } c^{\mathrm{T}}x \text{ s.t. } \begin{cases} Ax = b \\ Cx \leq d \end{cases} \quad \text{(LP)}$$

can be solved in a finite number of steps. The standard method to solve (LP) is the simplex method, developed in the late 1940s (see Dantzig, 1963), although interior point methods have become quite advanced and competitive for highly constrained problems (Wright, 1996). Methods to solve (LP) are widespread and well implemented. Currently, start of the art LP solvers can handle millions of variables and constraints and the application of further decomposition methods leads to the solution of problems that are two or three orders of magnitude larger than this. Because these methods are so widely known, further mention of the simplex method will not be described here (see the standard references: Edgar, Himmelblau, & Lasdon, 2001; Hillier & Lieberman, 1974 for more details). Also, the interior point method is described below from the perspective of more general nonlinear problems.

Quadratic programs represent a slight modification of (LP) and can be stated as

$$\text{Min } c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Qx \text{ s.t. } \begin{cases} Ax = b \\ Cx \leq d \end{cases} \quad \text{(QP)}$$

If the matrix Q is positive semi-definite (positive definite) when projected into the null space of the active constraints, then (QP) is (strictly) convex and the (QP) has a unique minimum (minimizer). Otherwise, local solutions exist for (QP) and more extensive global optimization methods are needed to obtain the global solution. Convex QPs can also be solved in a finite number of steps. Here, a number of active set strategies have been created that solve the KKT conditions of the QP and incorporate efficient updates of active constraints. Popular methods include null space algorithms (Gill, Murray, & Wright, 1981), range space methods and Schur complement methods. As with LPs, QP problems can also be solved with interior point methods (see Wright, 1996). Structures of large-scale QPs can be exploited quite efficiently with interior and Schur complement methods.

2.1. Solving the NLP problem

To introduce solution techniques for (NLP) we first consider solvers based on successive quadratic programming (SQP) as they allow the construction of a number of NLP algorithms based on Newton steps. Moreover, these solvers have been shown to require the fewest function evaluations to solve NLPs (Binder et al., 2001; Schittkowski, 1987) and

[1] The function $\phi(x)$ is convex over $x \in X$ if: $\phi(\alpha x_1 + (1-\alpha)x_2) \leq \alpha\phi(x_1) + (1-\alpha)\phi(x_2)$ holds for all $a \in (0,1)$ and $x_1, x_2 \in X$. Strict convexity requires that this inequality be strict.

they can be tailored to a broad range of process engineering problems with different structure.

SQP applies the equivalent of a Newton step to the KKT conditions of the nonlinear programming problem and this leads to a fast rate of convergence. An informal derivation proceeds as follows. At a stationary point of (NLP), x^*, the first-order KKT conditions for this problem are given by

$$\nabla f(x^*) + A(x^*)\lambda + C(x^*)v = 0 \quad \text{(KKTa)}$$

$$h(x^*) = 0 \quad \text{(KKTb)}$$

$$g(x^*) + s = 0 \quad \text{(KKTc)}$$

$$SVe = 0 \quad \text{(KKTd)}$$

$$(s, v) \geq 0 \quad \text{(KKTe)}$$

where $e = [1, 1, \ldots, 1]^T$, λ is the multiplier vector of the equalities, v the multipliers of the inequalities, $A(x) = \nabla h(x)$, $C(x) = \nabla g(x)$, $S = \text{diag}\{s\}$, $V = \text{diag}(v)$.

SQP methods find solutions that satisfy (KKT) by generating Newton-like search directions at iteration k. In general, one can classify SQP methods by the following categories:

- *Active set versus barrier methods* to handle bounds and inequality constraints in generating search directions.
- *Second-order information* can be provided in a number of ways and problem structure can be exploited for Newton-like steps.
- *Line search versus trust region* methods to enforce global convergence of the SQP iterations.

2.1.1. Active set versus barrier methods

The complementarity conditions (KKTd and KKTe) present a key difficulty in solving the KKT conditions as a set of equations. At the solution, the equations (KKTd) and active bounds (KKTe) are dependent and serve to make the KKT system ill-conditioned near the solution. SQP algorithms treat these conditions in two ways. In the active set strategy, discrete decisions are made regarding the active constraint set, $i \in I = \{i | g_i(x^*) = 0\}$, (KKTd) is replaced by $s_i = 0$, $i \in I$, and $v_i = 0$, $i \notin I$ and determining the active set is a combinatorial problem. A relatively inexpensive way to determine an estimate of the active set (and also satisfy (KKTe)) is to formulate, at a point x^k, and to solve the quadratic programming (QP) problem at iteration k, given by

$$\begin{aligned} \text{Min } & \nabla\phi(x^k)^T p + \tfrac{1}{2} p^T W(x^k, \lambda^k, v^k) p \\ \text{s.t. } & h(x^k) + A(x^k)^T p = 0 \\ & g(x^k) + C(x^k)^T p + s = 0,\ s \geq 0 \end{aligned} \quad \text{(SQP)}$$

The KKT conditions of (SQP) are given by

$$\nabla\phi(x^k) + W(x^k, \lambda^k, v^k) p + A(x^k)\lambda + C(x^k)v = 0 \quad \text{(QPKKTa)}$$

$$h(x^k) + A(x^k)^T p = 0 \quad \text{(QPKKTb)}$$

$$g(x^k) + C(x^k)^T p + s = 0 \quad \text{(QPKKTc)}$$

$$SVe = 0 \quad \text{(QPKKTd)}$$

$$(s, v) \geq 0 \quad \text{(QPKKTe)}$$

where $W(x, \lambda, v) = \nabla^2(f(x) + h(x)^T\lambda + g(x)^T v)$; $A(x^k) = \nabla h(x^k)$ and $C(x^k) = \nabla g(x^k)$ is the Hessian of the Lagrange function. It is easy to show that ((QPKKTa)–(QPKKTc)) correspond to a linearization of ((KKTa)–(KKTc)) at iteration k. Also, selection of the active set is now handled at the QP level in satisfying the conditions ((QPKKTd), (QPKKTe)). To evaluate and change candidate active sets, QP algorithms apply inexpensive matrix updating strategies to the KKT matrix associated with (SQP). Details of this approach can be found in (Fletcher, 1987; Nocedal & Wright, 1999).

To avoid the combinatorial problem of selecting the active set, barrier methods modify the NLP problem (1–3) to form:

$$\text{Min } \phi(x^k) - \mu \sum_i \ln s_i \quad \text{s.t.} \begin{cases} h(x^k) = 0 \\ g(x^k) + s = 0 \end{cases} \quad \text{(IP)}$$

where the solution to this problem has $s > 0$ for the penalty parameter $\mu > 0$, and decreasing μ to zero leads to solution of problem (NLP). The KKT conditions for this problem can be written as

$$\begin{aligned} & \nabla\phi(x^*) + A(x^*)\lambda + C(x^*)v = 0 \\ & h(x^*) = 0 \\ & g(x^*) + s = 0 \\ & SVe = \mu e \end{aligned} \quad \text{(IPKKT)}$$

and for $\mu > 0$, $s > 0$, and $v > 0$, Newton steps generated to solve (IPKKT) are well-behaved and analogous to (QPKKT), with a modification on the right hand side of (QPKKTd). Moreover, if W^k is positive definite in the null space of $A(x^k)^T$, the Newton step can be written as the following QP subproblem:

$$\begin{aligned} \text{Min } & \nabla\phi(x^k)^T p + \tfrac{1}{2} p^T W(x^k, \lambda^k, v^k) p \\ & -\mu(S^k)^{-1} e^T \Delta s + \tfrac{1}{2} \Delta s^T (S^k)^{-1} v^k \Delta s \\ \text{s.t. } & h(x^k) + A(x^k)^T p = 0 \\ & g(x^k) + C(x^k)^T p + s^k + \Delta s = 0 \end{aligned} \quad \text{(IPQP)}$$

where $s = s^k + \Delta s$. This QP can be further simplified if the inequality constraints take the form of simple bounds. Note that the complementarity conditions are now replaced by penalty terms in the objective function. The optimality conditions for this QP can now be written as a set of linear equations and the combinatorial problem of selecting the active set disappears.

In comparing these approaches, both methods possess clear trade-offs. Barrier methods may require more iterations to solve (IP) for various values of μ, while active set

methods require the solution of a more expensive QP subproblem (SQP). Thus, if there are few inequality constraints or an active set is known (say from a good starting guess, or the warm-start QP solution from a previous iteration) then solving (SQP) is not expensive and the active set method is favored. On the other hand, for problems with many inequality constraints, barrier methods are often faster, as they avoid the combinatorial problem of selecting the active set. This is especially the case for large-scale problems when a large number of bounds are active, as this approach eliminates the necessity of choosing an active set. Examples that demonstrate the performance of these approaches include the solution of linear model predictive control (MPC) problems (Rao, Rawlings, & Wright, 1998) and nonlinear MPC problems (Albuquerque, Gopal, Staus, Biegler, & Ydstie, 1997) using interior point QP solvers, as well as the solution of large optimal control problems using barrier NLP solvers. For instance, an efficient implementation of IPOPT allows the solution of problems with more than 2,000,000 variables and 4500 degrees of freedom (see Wächter, 2002).

2.1.2. Providing second-order information

With the development and increasing application of automatic differentiation tools, there are a number of modeling and simulation platforms where accurate first and second derivatives can be accessed for optimization. If second derivatives are available for the objective or constraint functions, they can be used to construct the Hessian, W^k, for the above QP subproblems. However, to obtain a unique solution for these QPs, the active constraint gradients must still be full rank and W^k must be positive definite when projected into the null space of the active constraint gradients. These properties may not hold far from the solution or for problems that do not satisfy sufficient second-order conditions, and corrections to the Hessian in (SQP) may be necessary (see Fletcher, 1987).

If second derivatives are not available, positive definite quasi-Newton approximations to the reduced Hessian (such as BFGS) are often quite successful. Here, if we define the *nd* vector $d^{\mathrm{T}} = [p^{\mathrm{T}}\ \Delta s^{\mathrm{T}}]$, an *nc* vector $c^{\mathrm{T}} = [h^{\mathrm{T}}\ (g+s)^{\mathrm{T}}]$ and

$$H = \begin{bmatrix} A^{\mathrm{T}} & 0 \\ C^{\mathrm{T}} & I \end{bmatrix},$$

then we can partition $d = Zd_z + Yd_y$ where $Z \in \Re^{nd\times(nd-nc)}$, $Y \in \Re^{nd\times nc}$, $HZ = 0$ and $[\ Z \quad Y\ ]$ is a nonsingular matrix. If we write the QPs (IPQP) or (SQP), without the bound constraint in the following form:

$$\mathrm{Min}_d\ a^{\mathrm{T}}d + \tfrac{1}{2}d^{\mathrm{T}}Qd \quad s.t.\ c + Hd = 0 \qquad \text{(QP1)}$$

then substituting the partition for d into (QP1) and simplifying leads to

$$d_y = -(\ H \quad Y\ )^{-1}c \qquad \text{(RD)}$$

and

$$\mathrm{Min}_{dz}(Z^{\mathrm{T}}a + Z^{\mathrm{T}}QYd_y)^{\mathrm{T}}d_z + \tfrac{1}{2}d_z^{\mathrm{T}}(Z^{\mathrm{T}}QZ)d_z. \qquad \text{(ND)}$$

With this decomposition, (RD) is often a sparse linear system of equations of order *nc* while (ND) has only ($nd - nc$) variables. If there are only a few degrees of freedom ($nd - nc$), then the quantities $(Z^{\mathrm{T}}QZ)$ and $(Z^{\mathrm{T}}QY)d_y$ are inexpensive to approximate with quasi-Newton update formulae and finite difference formulae, respectively. Moreover, a stabilized BFGS update approximation for $(Z^{\mathrm{T}}QZ)$ leads to a positive definite reduced Hessian in (ND) and a unique solution for the QP subproblem.

Finally, for problems with quadratic objective functions, as in data reconciliation, parameter estimation, and model predictive control, one can often assume that the value of the objective function and its gradient at the solution are vanishingly small. Under these conditions, one can show that the multipliers (λ, ν) also vanish and W can be substituted by $\nabla^2\phi(x^*)$. This *Gauss–Newton approximation* has been shown to be very efficient for the solution of least squares problems.

2.1.3. Line search versus trust region methods

To promote convergence from poor starting points, two types of globalization strategies, line search and trust region methods, are commonly used for the search directions calculated from the above QP subproblems. In a trust region approach, the constraint, $||d|| \leq \Delta$ is added to the QP. The step, $x^{k+1} = x^k + d$, is taken if there is sufficient reduction of a merit function (e.g., the objective function weighted with some measure of the constraint violations). Popular merit functions for SQP methods include the augmented Lagrangian function (of the form: $\phi(x) + \lambda^{\mathrm{T}}h(x) + \nu^{\mathrm{T}}g(x) + \rho||g(x)_+,\ h(x)||^2$) or exact penalty functions (of the form: $\phi(x) + \rho||g(x)_+,\ h(x)||$). Also the size of the trust region Δ is adjusted based on the agreement of the reduction of the actual merit function compared to its predicted reduction from the QP (see Nocedal & Wright, 1999). However, for values of Δ sufficiently small, (QP1) may have no solution. Instead, trust region constraints can be applied to subproblems for d_y and d_z, which replace (RD) and (ND), respectively. This approach has been applied in the KNITRO code (Byrd, Hribar, & Nocedal, 1997). Such methods have strong global convergence properties and are especially appropriate for ill-conditioned NLPs.

On the other hand, line search methods can be more efficient on problems with reasonably good starting points and well-conditioned QP subproblems, as in real time optimization. Typically, once the QP search direction is calculated from (SQP) or from (IPQP) a step size $\alpha \in [0, 1]$ is chosen so that $x^k + \alpha d$ leads to a sufficient decrease of a merit function. As a recent alternative, a novel line search strategy has been developed based on a bi-criterion minimization, with the objective function and constraint infeasibility as competing objectives. Termed the *filter* approach, this method, also developed for trust regions (Fletcher, Gould, Leyffer, Toint, &

Table 2
Representative SQP-type NLP solvers

Method	Inequality constraints	Globalization	Full vs. reduced space	Second-order information
IPOPT (Wächter & Biegler, 2002)	Barrier	Line search	Full or reduced space	Exact or quasi-Newton
KNITRO (Byrd et al., 1997)	Barrier	Trust region	Reduced space	Exact or quasi-Newton
LOQO (Vanderbei & Shanno, 1997)	Barrier	Line search	Full space	Exact
NPSOL (Gill et al., 1990)	Active set	Line search	Full space	Quasi-Newton
rSQP (Ternet & Biegler, 1996)	Active set	Line search	Reduced space	Quasi-Newton
SNOPT (Gill, Murray, & Saunders, 1998)	Active set	Line search	Reduced space	Quasi-Newton
SOCS (Betts, 2001)	Active set	Line search	Full space	Exact
SRQP (PSE, 2002)	Active set	Line search	Reduced space	Quasi-Newton
TRICE (Dennis, Heinkenschloss, & Vicente, 1998)	Barrier	Trust region	Reduced space	Exact or quasi-Newton

Waechter, 2001), usually leads to better performance than line searches based on merit functions. A detailed description of the filter line search method can be found in (Wächter & Biegler, 2002).

Table 2 presents a sampling of available codes of SQP-type solvers that represent the above classifications.

2.2. Other gradient-based NLP solvers

In addition to SQP methods, several NLP solvers have been developed and adapted for large scale problems. Generally, these methods require more function evaluations than SQP methods but they perform very well when interfaced to optimization modeling platforms such as AMPL, CUTE or GAMS, where function evaluations are cheap. All of these can be derived from the perspective of applying Newton steps to portions of the KKT conditions of (NLP).

LANCELOT (Conn, Gould, & Toint, 2000) is based on the solution of bound constrained subproblems. Here an augmented Lagrangian is formed from (NLP) and the following subproblem is solved:

$$\begin{aligned}&\text{Min } f(x) + \lambda^{\mathrm{T}} h(x) + \nu^{\mathrm{T}}(g(x)+s) \\ &\quad + \tfrac{1}{2}\rho \|h(x),\ g(x)+s\|^2 \text{ s.t. } s \geq 0\end{aligned} \tag{AL}$$

The above subproblem can be solved very efficiently for fixed values of the multipliers, λ and ν, and penalty parameter ρ. Here, a gradient projection, trust region algorithm is applied. Once subproblem (AL) is solved, the multipliers and penalty parameter are updated in an outer loop and the cycle repeats until the KKT conditions for (NLP) are satisfied. LANCELOT works best when exact second derivatives are available. This promotes a fast convergence rate in solving each subproblem and allows the trust region method to exploit directions of negative curvature in the Hessian matrix.

MINOS (Murtagh & Saunders, 1987) is a well-implemented package with a number of similarities to SQP-type methods. Here, the quadratic programming subproblem is replaced by a linearly constrained NLP, with an augmented Lagrangian objective function:

$$\begin{aligned}&\text{Min } f(x) + \lambda^{\mathrm{T}} h(x) + \nu^{\mathrm{T}}(g(x)+s) \\ &\quad + \tfrac{1}{2}\rho \|h(x),\ g(x)+s\|^2 \\ &\text{s.t. } h(x^k) + A(x^k)^{\mathrm{T}} p = 0 \\ &\quad g(x^k) + C(x^k)^{\mathrm{T}} p + s = 0,\ s \geq 0\end{aligned} \tag{LCNLP}$$

At the current iterate, x^k, MINOS selects an active set and applies a reduced space decomposition to (LCNLP). In fact, the decomposition is implemented in such a way that MINOS naturally defaults to the LP simplex method if the objective and constraint functions are linear. Upon eliminating the dependent and bounded variables, an unconstrained quasi-Newton method is applied to the remaining variables. At the solution of this subproblem, the constraints are relinearized and the cycle repeats until the KKT conditions of (NLP) are satisfied. For problems with few degrees of freedom, this leads to an extremely efficient method even for very large problems. Note that the augmented Lagrangian function is used in (LCNLP) in order to penalize movement away from the feasible region. MINOS has been interfaced to both GAMS and AMPL and enjoys widespread use. It performs especially well on problems with few nonlinear constraints.

Finally the generalized reduced gradient (GRG) methods, GRG2, CONOPT, and SOLVER, consider the same subproblem as in MINOS, but also ensure that the constraints are always satisfied at each iterate of (LCNLP) (Edgar et al., 2001). GRG methods select an active set and applies a reduced space decomposition. Upon eliminating the dependent and bounded variables, an unconstrained quasi-Newton method is applied to the remaining variables, along with a constraint restoration step. Note that since x^k is always feasible, the augmented Lagrangian function in (LCNLP) simply becomes $f(x)$. Among all of the gradient based NLP solvers, the GRG methods are the most popular; the SOLVER code has been incorporated into MS Excel and optimization is now a widely used option in Excel. In the GAMS and AMPL modeling environments, CONOPT is an efficient and widely used code that is usually more reliable than MINOS.

2.3. Optimization without derivatives

We now consider a broad class of optimization strategies that do not require derivative information. These methods have the advantage of easy implementation and little prior knowledge of the optimization problem. In particular, such methods are well-suited for 'quick and dirty' optimization studies that explore the scope of optimization for new problems, prior to investing effort for more sophisticated modeling and solution strategies. Most of these methods are derived from heuristics that naturally spawn numerous variations. As a result, a very broad literature describes these methods. Here, we discuss only a few important trends in this area and describe the mostly widely used methods using the following classifications.

2.3.1. Classical direct search methods

Developed in the 1960s and 1970s these methods include one-at-a-time search, a variety of pattern searches, and methods based on experimental designs (EVOP). In fact, when *Computers and Chemical Engineering* was founded 25 years ago, direct search methods were the most popular optimization methods in chemical engineering. Methods that fall into this class include the conjugate direction method of Powell (1964), simplex and complex searches, in particular Nelder and Mead (1965), the adaptive random search methods of Luus and Jaakola (1973), Goulcher and Cesares Long (1978) and Banga and Seider (1996). All of these methods are based on well defined search methods that require only objective function values for unconstrained minimization. Associated with these methods are numerous studies with successful results on a wide range of process problems. Moreover, many of these methods include heuristics that prevent premature termination (e.g., directional flexibility in the complex search as well as random restarts and direction generation).

2.3.2. Simulated annealing

This strategy derives from a class of heuristics with analogies to the motion of molecules in the cooling and solidification of metals (Laarhoven & Aarts, 1987). Here, a 'temperature' parameter, T, can be raised or lowered to influence the probability of accepting points that do not improve the objective function. The method starts with a base point, x, and objective value, $f(x)$. The next point x' is chosen at random from a distribution. If $f(x') < f(x)$, the move is accepted with x' as the new point. Otherwise, x' is accepted with probability $p(T, x', x)$. Options include the Metropolis distribution,

$$p(T, x, x') = \exp\left(-\frac{f(x') - f(x)}{T}\right)$$

and the Glauber distribution,

$$p(T, x, x') = \frac{\exp(-(f(x') - f(x))/T)}{1 + \exp(-(f(x') - f(x)))/T}$$

Temperature is then reduced and the method continues until no further progress is made. Simulated annealing has been employed on a wide variety of process examples including separation synthesis (Floquet, Pibouleau, & Domenech, 1994), heat exchanger network synthesis (Dolan, Cummings, & Levan, 1989) and batch process scheduling (Das, Cummings, & LeVan, 1990).

2.3.3. Genetic algorithms

This approach, first proposed in (Holland, 1975) is based on the analogy of improving a population of solutions through modifying their gene pool. Two forms of genetic modification, crossover or mutation, are used and the elements of the optimization vector, x, are represented as binary strings. Crossover deals with random swapping of vector elements (among parents with highest objective function values or other rankings of population) or any linear combinations of two parents. Mutation deals with the addition of a random variable to elements of the vector. Genetic algorithms (GAs) have seen widespread use in process engineering and a number of codes are available. For instance, Edgar, Himmelblau, and Lasdon (2002) describe a related GA algorithm described that is available in Excel. Case study examples in process engineering include batch process scheduling (Jung, Lee, & Lee, 1998), Loehl, Schulz, and Engell (1998), sensor network design (Sen, Narasimhan, & Deb, 1998), and mass exchanger network synthesis (Garrard & Fraga, 1998).

2.3.4. Derivative free optimization (DFO)

In the past decade, the availability of parallel computers and faster computing hardware and the need to incorporate complex simulation models within optimization studies, have led a number of optimization researchers to reconsider classical direct search approaches. In particular, Dennis and Torczon (1991) developed a multidimensional search algorithm that extends the simplex approach of Nelder and Mead. They note that the Nelder–Mead algorithm fails as the number of variables increases, even for very simple problems. To overcome this, their multidimensional simplex approach combines reflection, expansion and contraction steps that act as line search algorithms for a number of linear independent search directions. This approach is easily adapted to parallel computation and the method can be tailored to the number of processors available. Moreover, Torczon (1991) showed that this approach converges to locally optimal solutions for unconstrained problems and observed an unexpected performance synergy when multiple processors are used. It should be noted that even EVOP and Hooke–Jeeves may be amenable to this convergence analysis, although the multidimensional search is much more efficient. The work of Dennis and Torczon (1991) has spawned considerable research on the analysis and code development for DFO methods. For instance, Conn, Scheinberg and Toint (1997) construct a multivariable DFO algorithm that uses a surrogate model

for the objective function within a trust region method. Here points are sampled to obtain a well-defined quadratic interpolation model and descent conditions from trust region methods enforce convergence properties. A number of trust region methods that rely on this approach are reviewed in Conn et al. (1997). Moreover, a number of DFO codes have been developed that lead to black box optimization implementations for large, complex simulation models. These include the DAKOTA package at Sandia National Lab (Eldred, 2002) and FOCUS developed at Boeing Corporation (Booker, Dennis, Frank, & Serafini, 1998).

All of the above methods are easy to apply to a wide variety of problem types and optimization models. Moreover, because their termination criteria are not based on gradient information and stationary points, these methods are often more likely to favor the search for global rather than locally optimal solutions. These methods can also be adapted easily to include integer variables. However, no rigorous convergence properties to globally optimal solutions have yet been discovered.

Derivative free methods are best suited for unconstrained problems or for problems with simple bounds. Otherwise, they may have difficulties in handling constraints, as the only options open for handling constraints are equality constraint elimination or addition of penalty functions for inequality constraints. Both approaches can be unreliable and may lead to failure of the optimization algorithm. Finally, the performance of derivative free methods scales poorly (and often exponentially) with the number of decision variables. While performance can be improved with the use of parallel computing, these methods are rarely applied to problems with more than a few dozen decision variables.

3. Discrete optimization

In many applications in process systems engineering it is required to model discrete decisions such as selection of units in a flowsheet or sequences in scheduling, or number of units or batches. The former are commonly represented with 0–1 variables, while the latter are represented with integer variables that are often approximated as continuous if they take large values. In the sections below we review the generalization of (NLP), or alternatively special cases of problem (MIP).

3.1. Mixed-integer linear programming

MILP problems have the general form:

$$\min Z = a^{\mathrm{T}}x + b^{\mathrm{T}}y \quad \text{s.t.} \begin{cases} Ax + By \le d \\ x \ge 0,\ y \in \{0,1\}^m \end{cases} \qquad \text{(MILP)}$$

For the case when no discrete variables y are involved, the problem reduces to a linear programming (LP) problem. MILP methods have been largely developed by operations researchers, and therefore we only provide a brief review. The major contribution of chemical engineers in this area has been to discover problems and applications that can be framed in the form of problem (MILP) (e.g. see Grossmann, Caballero, & Yeomans, 1999; Grossmann, Quesada, Raman, & Voudouris, 1996; Kallrath, 2000; Pinto & Grossmann, 1998).

MILP methods (Nemhauser & Wolsey, 1988) rely largely on the simplex LP-based branch and bound method (Dakin, 1965). This method consists of a tree enumeration in which the integer space is successively partitioned into relaxed LP subproblems that are solved at each node of the tree. The initial node in which the variables y in (MILP) are treated as continuous, yields an absolute lower bound (minimization case) to the optimal solution. If as is often the case, this solution exhibits one or more y variables with fractional values a tree search is performed according to pre-specified branching rules (e.g. depth first, minimum reduced cost). The LP solution of each node yields a lower bound to the solution of the descendant nodes. When a feasible integer solution is found this yields an upper bound. Nodes are eliminated based on these bounding properties, and the enumeration continues until the difference between the current lower and upper bounds lies within a tolerance.

In the worst case, the branch and bound method may end up enumerating most of the nodes in the tree, and therefore, not unexpectedly, MILP methods are NP-hard. To overcome the potential exponential computation in MILP problems two major developments have been the use of preprocessing and cutting planes. Pre-processing techniques rely on techniques for automatic elimination of variables and constraints, reduction of bound, fixing of integer variables, and reformulation of constraints. Cutting plane techniques are derived from theoretical analysis of the integer convex hull of either specialized problems (e.g. knapsack, network flows), or from general unstructured MILP problems. Cutting planes are generally generated from the LP relaxation and a separation problem that cuts off a portion of the relaxed continuous feasible region that does not contain the integer optimal solution. Cutting planes have usually the effect of producing tighter lower bounds for the LP relaxation. Recent trends in MILP include the development of branch-and-price (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998) and branch-and-cut methods such as the lift-and-project method by Balas, Ceria and Cornuejols (1993), in which cutting planes (e.g. Gomory, mixed-integer rounding cuts) are generated as part of the branch and bound enumeration. See also Johnson, Nemhauser, and Savelsbergh (2000) for a recent review on MILP.

MILP codes build on LP codes that are widely available. The best known include CPLEX (ILOG, 2000), XPRESS

(Dash Associates, 1999), and OSL (IBM, 1992) all which have achieved impressive improvements in their problem solving capabilities. It is worth noting that since MILP problems are NP-hard it is always possible to run into time limitations when solving problems with a large number of 0–1 variables, especially if the integrality gap (difference between optimal integer objective and optimal LP relaxation) is large. However, it is also important to emphasize that the improvements in solving capabilities for MILP problems have increased by tens of orders of magnitude over the last few years due to a combination of use of cutting planes, improved preprocessing and faster computers (Bixby, Fenelon, Gu, Rothberg, & Wunderling, 2002).

3.2. Mixed-integer nonlinear programming

MINLP models typically arise in synthesis and design problems, and in planning and scheduling problems. MINLP clearly provides much greater modeling flexibility for tackling a large variety of problems. While MILP methods have been largely developed outside process systems engineering, chemical engineers have played a prominent role in the development of MINLP methods.

Major methods for MINLP problems include Branch and Bound (BB) (Borchers & Mitchell, 1994; Gupta & Ravindran, 1985; Leyffer, 2001; Stubbs & Mehrotra, 1999), which is a direct extension of the linear case, except that NLP subproblems are solved at each node. Generalized benders decomposition (GBD) (Benders, 1962; Geoffrion, 1972), and Outer-Approximation (OA) (Duran & Grossmann, 1986; Ding-Mai & Sargent, 1992; Fletcher & Leyffer, 1994; Quesada & Grossmann, 1992; Yuan, Zhang, Piboleau, & Domenech, 1988), are iterative methods that solve a sequence of alternate NLP subproblems with all the 0–1 variables fixed, and MILP master problems that predict lower bounds and new values for the 0–1 variables. Finally, the Extended Cutting Plane Method (ECP) (Westerlund & Pettersson, 1995) is a variation that does not require the solution of NLP subproblems.

For the derivation of the above methods the MINLP problem is assumed to be given by

$$\min Z = f(x, y) \text{ s.t. } \begin{cases} g_j(x, y) \leq 0 \; j \in J \\ x \in X, \; y \in Y \end{cases} \tag{P1}$$

where $f(\cdot)$, $g(\cdot)$ are *convex*, *differentiable* functions, J is the index set of inequalities, and x and y are the continuous and discrete variables, respectively. The set X is commonly assumed to be a convex compact set, e.g. $X = \{x | x \in R^n, Dx \leq d, x^L \leq x \leq x^U\}$; the discrete set Y corresponds to a polyhedral set of integer points, $Y = \{y | y \in Z^m, Ay \leq a\}$, which in most applications is restricted to 0–1 values, $y \in \{0, 1, 1\}^m$. In most applications of interest the objective and constraint functions $f(\cdot)$, $g(\cdot)$ are linear in y (e.g. fixed cost charges and mixed-logic constraints):

$$f(x, y) = c^{\mathrm{T}} y + r(x), \qquad g(x, y) = By + h(x).$$

3.2.1. NLP subproblems

There are three basic NLP subproblems that can be considered for problem (P1).

3.2.1.1. NLP relaxation.

$$\min Z^k_{\mathrm{LB}} = f(x, y) \text{ s.t. } \begin{cases} g_j(x, y) \leq 0, \; j \in J \\ x \in X, \; y \in Y_{\mathrm{R}} \\ y_i \leq \alpha^k_i, \; i \in I^k_{\mathrm{FL}} \\ y_i \geq \beta^k_i, \; i \in I^k_{\mathrm{FU}} \end{cases} \tag{NLP1}$$

where Y_{R} is the continuous relaxation of the set Y, and I^k_{FL}, I^k_{FU} are index subsets of the integer variables y_i, $i \in I$, which are restricted to lower and upper bounds, α^k_i,β^k_i at the kth step of a branch and bound enumeration procedure. It should be noted that $\alpha^k_i = \lfloor y^l_i \rfloor$, $\beta^k_i = \lceil y^m_i \rceil$, $l < k$, $m < k$, where y^l_i, y^m_i, are noninteger values at a previous step, and are the floor and ceiling functions, respectively.

Also note that if $I^k_{\mathrm{FU}} = I^k_{\mathrm{FL}} = \emptyset$ ($k = 0$), (NLP1) corresponds to the continuous NLP relaxation of (P1). Except for few and special cases, the solution to this problem yields in general a noninteger vector for the discrete variables. Problem (NLP1) also corresponds to the kth step in a branch and bound search. The optimal objective function provides an absolute lower bound to (P1); for $m \geq k$, the bound is only valid for $I^k_{\mathrm{FL}} \subset I^m_{\mathrm{FL}}$, $I^k_{\mathrm{FU}} \subset I^m_{\mathrm{FL}}$.

3.2.1.2. NLP subproblem for fixed y^k.

$$\mathrm{Min}\, Z^k_U = f(x, y^k) \text{ s.t. } \begin{cases} g_j(x, y^k) \leq 0, \; j \in J \\ x \in X \end{cases} \tag{NLP2}$$

which yields an upper bound Z^k_{U} to (P1) provided (NLP2) has a feasible solution. When this is not the case, we consider the next subproblem.

3.2.1.3. Feasibility subproblem for fixed y^k.

$$\mathrm{Min}\, u \text{ s.t. } \begin{cases} g_j(x, y^k) \leq u \; j \in J \\ x \in X, \; u \in R^1 \end{cases} \tag{NLPF}$$

which can be interpreted as the minimization of the infinity-norm measure of infeasibility of the corresponding NLP subproblem. Note that for an infeasible subproblem the solution of (NLPF) yields a strictly positive value of the scalar variable u.

3.2.2. MILP cutting plane

The convexity of the nonlinear functions is exploited by replacing them with supporting hyperplanes, that are generally, but not necessarily, derived at the solution of the NLP subproblems. In particular, the new values y^k (or (x^k, y^k)) are

obtained from a cutting plane MILP problem that is based on the K points, (x^k, y^k), $k = 1, \ldots, K$ generated at the K previous steps:

$$\text{Min } Z_{\text{L}}^K = \alpha \text{ s.t. } \left\{ \begin{array}{c} \alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k)^{\text{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ g_j(x^k, y^k) + \nabla g_j(x^k, y^k)^{\text{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0, \ j \in J^k \\ x \in X, \ y \in Y \end{array} \right\} \quad k = 1, \ldots, K \text{(M-MIP)}$$

where $J^k \subseteq J$. When only a subset of linearizations is included, these commonly correspond to violated constraints in problem (P1). Alternatively, it is possible to include all linearizations in (M-MIP). The solution of (M-MIP) yields a valid lower bound Z_{L}^K to problem (P1). This bound is nondecreasing with the number of linearization points K. Note that since the functions $f(x, y)$ and $g(x, y)$ are convex, the linearizations in (M-MIP) correspond to outer-approximations of the nonlinear feasible region in problem (P1). Here it can be seen that the convex objective function is being underestimated, and the convex feasible region overestimated with these linearizations.

The different methods can be classified according to their use of the subproblems (NLP1), (NLP2) and (NLPF), and the specific specialization of the MILP problem (M-MIP). It should be noted that in the GBD and OA methods (case 3.2.1.2), and in the LP/NLP based branch and bound method (case 3.2.1.3), the problem (NLPF) is solved if infeasible subproblems are found. Each of the methods is explained next in terms of the basic subproblems.

Branch and bound. While the earlier work in BB was aimed at linear problems (Dakin, 1965), this method can also be applied to nonlinear problems (Borchers & Mitchell, 1994; Gupta & Ravindran, 1985; Leyffer, 2001; Nabar & Schrage, 1991; Stubbs & Mehrotra, 1999). The BB method starts by solving first the continuous NLP relaxation. If all discrete variables take integer values the search is stopped. Otherwise, a tree search is performed in the space of the integer variables y_i. $i \in I$. These are successively fixed at the corresponding nodes of the tree, giving rise to relaxed NLP subproblems of the form (NLP1) which yield lower bounds for the subproblems in the descendant nodes. Fathoming of nodes occurs when the lower bound exceeds the current upper bound, when the subproblem is infeasible or when all integer variables y_i take on discrete values. The latter yields an upper bound to the original problem.

The BB method is generally only attractive if the NLP subproblems are relatively inexpensive to solve, or when only few of them need to be solved. This could be either because of the low dimensionality of the discrete variables, or because the integrality gap of the continuous NLP relaxation of (P1) is small.

Outer-Approximation (Duran & Grossmann, 1986; Fletcher & Leyffer, 1994; Yuan et al., 1988). The OA method arises when NLP subproblems (NLP2) and MILP master problems (M-MIP) with $J^k = J$ are solved successively in a cycle of iterations to generate the points (x^k, y^k).

Since the master problem (M-MIP) theoretically requires for equivalence with (P1), the solution of all feasible discrete variables y^k, the following MILP relaxation is considered, assuming that the solution of K different NLP subproblems (where $K = |KFS \cup KIS|$, KFS is the set of solutions from (NLP2) and KIS is the set of solutions from (NLPF)) is available:

$$\text{Min } Z_{\text{L}}^K = \alpha \text{ s.t. } \left\{ \begin{array}{c} \alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k)^{\text{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ g_j(x^k, y^k) + \nabla g_j(x^k, y^k)^{\text{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0, \ j \in J \\ x \in X, \ y \in Y \end{array} \right\} \quad k = 1, \ldots, K \quad \text{(RM-OA)}$$

Given the assumption on convexity of the functions $f(x, y)$ and $g(x, y)$, the solution of problem (RM-OA), Z_{L}^K, corresponds to a lower bound of the solution of problem (P1). Also, since function linearizations are accumulated as iterations proceed, the master problems (RM-OA) yield a non-decreasing sequence of lower bounds, $Z_{\text{L}}^1 \cdots \leq Z_{\text{L}}^k \leq \cdots \leq Z_{\text{L}}^K$, since linearizations are accumulated as iterations k proceed.

The OA algorithm as proposed by Duran and Grossmann (1986) consists of performing a cycle of major iterations, $k = 1, \ldots, K$, in which (NLP1) is solved for the corresponding y^k, and the relaxed MILP master problem (RM-OA) is updated and solved with the corresponding function linearizations at the point (x^k, y^k), for which the corresponding subproblem NLP2 is solved. If feasible, the solution to that problem is used to construct the first MILP master problem; otherwise a feasibility problem (NLPF) is solved to generate the corresponding continuous point (Fletcher and Leyffer, 1994). The initial MILP master problem (RM-OA) then generates a new vector of discrete variables. The (NLP2) subproblems yield an upper bound that is used to define the best current solution, $\text{UB}^K = \text{Min}_k \{Z_{\text{U}}^k\}$.

The cycle of iterations is continued until this upper bound and the lower bound of the relaxed master problem, Z_L^K, are within a specified tolerance. One way to avoid solving the feasibility problem (NLPF) in the OA algorithm when the discrete variables in problem (P1) are 0–1, is to introduce the following integer cut whose objective is to make infeasible the choice of the previous 0–1 values generated at the K previous iterations (Duran & Grossmann, 1986):

$$\sum_{i\in B^k} y_i - \sum_{i\in N^k} y_i \le |B^k| - 1, \quad k = 1, \ldots, K \qquad \text{(ICUT)}$$

where $B^k = \{i | y_i^k = 1\}$, $N^k = \{i | y_i^k = 0\}$, $k = 1, \ldots, K$. This cut becomes very weak as the dimensionality of the 0–1 variables increases. However, it has the useful feature of ensuring that new 0–1 values are generated at each major iteration. In this way, the algorithm will not return to a previous integer point when convergence is achieved. Using the above integer cut the termination takes place as soon as $Z_L^K = \mathrm{UB}^K$.

The OA method generally requires relatively few cycles or major iterations. One reason is that the OA algorithm trivially converges in one iteration if $f(x, y)$ and $g(x, y)$ are linear. This property simply follows from the fact that if $f(x, y)$ and $g(x, y)$ are linear in x and y the MILP master problem (RM-OA) is identical to the original problem (P1).

It is also important to note that the MILP master problem need not be solved to optimality. In fact given the upper bound UB^k and a tolerance ε, it is sufficient to generate the new (y^k, x^k) by finding a mixed-integer solution to the MILP that lies below $\mathrm{UB}^K - \varepsilon$. In this case, the OA iterations are terminated when (RM-OAF) has no feasible solution.

Generalized benders decomposition (Geoffrion, 1972). The GBD method (see Flippo & Kan, 1993) is similar to the OA method. The difference arises in the definition of the MILP master problem (M-MIP). In the GBD method only active inequalities are considered $J^k = \{j | g_j(x^k, y^k) = 0\}$ and the set is disregarded. In particular, consider an outer-approximation given at a given point (x^k, y^k),

$$\begin{aligned} &\alpha \ge f(x^k, y^k) + \nabla f(x^k, y^k)^{\mathrm{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ &g(x^k, y^k) + \nabla g(x^k, y^k)^{\mathrm{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \le 0 \end{aligned} \qquad (\mathrm{OA}^k)$$

where for a fixed y^k the point x^k corresponds to the optimal solution to problem (NLP2). Making use of the Karush–Kuhn–Tucker conditions and eliminating the continuous variables x, the inequalities in (OA^k) can be reduced as follows (Quesada & Grossmann, 1992):

$$\begin{aligned} \alpha \ge\ & f(x^k, y^k) + \nabla_y f(x^k, y^k)^{\mathrm{T}} (y - y^k) \\ & + (\mu^k)^{\mathrm{T}} [g(x^k, y^k) + \nabla_y g(x^k, y^k)^{\mathrm{T}} (y - y^k)] \end{aligned} \qquad (\mathrm{LC}^k)$$

which is the Lagrangian cut projected in the y-space. This can be interpreted as a surrogate constraint of the equations in (OA^k), because it is obtained as a linear combination of these.

For the case when there is no feasible solution to problem (NLP2), then if the point x^k is obtained from the feasibility subproblem (NLPF), the following feasibility cut projected in y can be obtained using a similar procedure,

$$(\lambda^k)^{\mathrm{T}} [g(x^k, y^k) + \nabla_y g(x^k, y^k)^{\mathrm{T}} (y - y^k)] \le 0 \qquad (\mathrm{FC}^k)$$

In this way, the problem (M-MIP) reduces to a problem projected in the y-space

$$\mathrm{Min}\ Z_L^K = \alpha \ \mathrm{s.t.} \begin{cases} \alpha \ge f(x^k, y^k) + \nabla_y f(x^k, y^k)^{\mathrm{T}} (y - y^k) + (\mu^k)^{\mathrm{T}} [g(x^k, y^k) + \nabla_y g(x^k, y^k)^T (y - y^k)] \, k \in KFS \\ (\lambda^k)^{\mathrm{T}} [g(x^k, y^k) + \nabla_y g(x^k, y^k)^{\mathrm{T}} (y - y^k)] \le 0, \ k \in KIS, \ x \in X, \ \alpha \in R^1 \end{cases} \qquad \text{(RM-GBD)}$$

where KFS is the set of feasible subproblems (NLP2) and KIS is the set of infeasible subproblems whose solution is given by (NLPF). Also $|KFS \cup KIS| = K$.

Since the master problem (RM-GBD) can be derived from the master problem (RM-OA), in the context of problem (P1), GBD can be regarded as a particular case of the OA algorithm. In fact given the same set of K subproblems, the lower bound predicted by the relaxed master problem (RM-OA) can be shown to be greater or equal to the one predicted by the relaxed master problem (RM-GBD). This property follows from the fact that the Lagrangian and feasibility cuts, (LC^k) and (FC^k), are surrogates of the OAs (OA^k). Given the fact that the lower bounds of GBD are generally weaker, this method commonly requires a larger number of cycles or major iterations. As the number of 0–1 variables increases this difference becomes more pronounced. This is to be expected since only one new cut is generated per iteration. Therefore, user-supplied constraints must often be added to the master problem to strengthen the bounds. Also, it is sometimes possible to generate multiple cuts from the solution of an NLP subproblem in order to strengthen the lower bound (Magnanti & Wong, 1981). As for the OA algorithm, the trade-off is that while it generally predicts stronger lower bounds than GBD, the computational cost for solving the master problem (M-OA) is greater since the number of constraints added per iteration is equal to the number of nonlinear constraints plus the nonlinear objective. Sahinidis and Grossmann (1991) have shown that if problem (P1) has zero integrality gap, the GBD algorithm converges in one iteration once the optimal (x^*, y^*) is found. This property implies that the only case one can expect the GBD method to terminate in one iteration, is when the initial discrete vector is the optimum, and when the objective value of the NLP relaxation of problem (P1) is the same as the objective of

the optimal mixed-integer solution. One further property that relates the OA and GBD algorithms is that a cut obtained from performing one Benders iteration on the MILP master (RM-OA) is equivalent to the cut obtained from the GBD algorithm. By making use of this property, instead of solving the MILP (RM-OA) to optimality, for instance by LP-based branch and bound, one can generate a GBD cut by simply performing one Benders (1962) iteration on the MILP.

Extended cutting plane (Westerlund & Pettersson, 1995). The ECP method, which is an extension of Kelly's cutting plane algorithm for convex NLP (Kelley, 1960), does not rely on the use of NLP subproblems and algorithms. It relies only on the iterative solution of the problem (M-MIP) by successively adding a linearization of the most violated constraint at the predicted point (x^k, y^k): $J^k = \{\hat{j} \in \arg\{\mathrm{Max}_{j\in J}\ g_j(x^k, y^k)\}$. Convergence is achieved when the maximum constraint violation lies within the specified tolerance. The optimal objective value of (M-MIP) yields a non-decreasing sequence of lower bounds. It is of course also possible to either add to (M-MIP) linearizatons of all the violated constraints in the set J^k, or linearizations of all the nonlinear constraints $j \in J$. In the ECP method, the objective must be defined as a linear function, which can easily be accomplished by introducing a new variable to transfer nonlinearities in the objective as an inequality.

Note that since the discrete and continuous variables are converged simultaneously, the ECP method may require a large number of iterations. However, this method solves in one iteration (as does the OA method) in the limiting case when all the functions are linear.

LP/NLP based branch and bound (Quesada & Grossmann, 1992). This method is similar in spirit to a branch and cut method, and avoids the complete solution of the MILP master problem (M-OA) at each major iteration. The method starts by solving an initial NLP subproblem, which is linearized as in (M-OA). The basic idea consists then of performing an LP-based branch and bound method for (M-OA) in which NLP subproblems (NLP2) are solved at those nodes in which feasible integer solutions are found. By updating the representation of the master problem in the current open nodes of the tree with the addition of the corresponding linearizations, the need of restarting the tree search is avoided.

This method can also be applied to the GBD and ECP methods. The LP/NLP method commonly reduces quite significantly the number of nodes to be enumerated. The trade-off, however, is that the number of NLP subproblems may increase. Computational experience has indicated that often the number of NLP subproblems remains unchanged. Therefore, this method is better suited for problems in which the bottleneck corresponds to the solution of the MILP master problem. Leyffer (1993) has reported substantial savings with this method.

3.3. Extensions of MINLP methods

In this subsection, we present an overview of some of the major extensions of the methods presented in the previous section.

Quadratic master problems. For most problems of interest, problem (P1) is linear in y: $f(x, y) = \phi(x) + c^{\mathrm{T}}y$, $g(x, y) = h(x) + By$. When this is not the case Fletcher and Leyffer (1994) suggested to include in the feasibility version of (RMIP-OA) a quadratic approximation $\nabla^2 L(x^k, y^k)$ of the Hessian of the Lagrangian of the last NLP subproblem, which yields a mixed-integer quadratic programming (MIQP) problem. Ding-Mai and Sargent (1992), found that the quadratic approximations can help to reduce the number of major iterations since an improved representation of the continuous space is obtained. Note also that for convex $f(x, y)$ and $g(x, y)$ using an MIQP leads to rigorous solutions since the outer-approximations remain valid. Also, if the function $f(x, y)$ is nonlinear in y, and y is a general integer variable, Fletcher and Leyffer (1994) have shown that the original OA algorithm may require a much larger number of iterations to converge than when the master problem (M-MIQP) is used. This, however, comes at the price of having to solve an MIQP instead of an MILP. Of course, the ideal situation is the case when the original problem (P1) is quadratic in the objective function and linear in the constraints, as then (M-MIQP) is an exact representation of such a mixed-integer quadratic program.

Reducing dimensionality of the master problem in OA. The master problem (RM-OA) can involve a rather large number of constraints, due to the accumulation of linearizations. One option is to keep only the last linearization point, but this can lead to nonconvergence even in convex problems, since then the monotonic increase of the lower bound is not guaranteed. A rigorous way of reducing the number of constraints without greatly sacrificing the strength of the lower bound can be achieved in the case of the "largely" linear MINLP problem:

$$\mathrm{Min}\ Z = a^{\mathrm{T}}w + r(v) + c^{\mathrm{T}}y \ \mathrm{s.t.} \begin{cases} Dw + t(v) + Cy \le 0 \\ Fw + Gv + Ey \le b \\ w \in W, v \in V, y \in Y \end{cases} \quad \text{(PL)}$$

where (w, v) are continuous variables and $r(v)$ and $t(v)$ are nonlinear convex functions. As shown by Quesada and Grossmann (1992), linear approximations to the nonlinear objective and constraints can be aggregated with the following MILP master problem:

$$\mathrm{Min}\ Z = a^{T}w + r(v) + c^{T}y \quad \mathrm{s.t.} \begin{cases} \beta \ge r(v^k) + (\lambda^k)^{\mathrm{T}}[Dw + t(v^k) + Cy] - (\mu^k)^{\mathrm{T}}(G(v - v^k) \quad k = 1, \ldots, K \\ Fw + Gv + Ey \le b \\ w \in W,\ v \in V,\ y \in Y,\ \beta \in R^1 \end{cases} \quad \text{(M-MIP)}$$

Numerical results have shown that the quality of the bounds is not greatly degraded with the above MILP as might happen if GBD is applied to (PL).

Handling of equalities. For the case when linear equalities of the form $h(x, y) = 0$ are added to (P1) there is no major difficulty since these are invariant to the linearization points. If the equations are nonlinear, however, there are two difficulties. First, it is not possible to enforce the linearized equalities at K points. Second, the nonlinear equations may generally introduce nonconvexities, unless they relax as convex inequalities (see Bazaraa, Sherali, & Shetty, 1994). Kocis and Grossmann (1987) proposed an equality relaxation strategy in which the nonlinear equalities are replaced by the inequalities,

$$T^k \nabla h(x^k, y^k)^{\mathrm{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0$$

where and $T^k = \{t_{ii}^k\}$, and $t_{ii}^k = \mathrm{sign}\{\lambda_i^k\}$ in which λ_i^k is the multiplier associated to the equation $h_i(x, y) = 0$. Note that if these equations relax as the inequalities $h(x, y) \leq 0$ for all y, and $h(x, y)$ is convex, this is a rigorous procedure. Otherwise, nonvalid supports may be generated. Also, note that in the master problem of GBD, (RM-GBD), no special provision is required to handle equations since these are simply included in the Lagrangian cuts. However, similar difficulties as in OA arise if the equations do not relax as convex inequalities.

Handling of nonconvexities. When $f(x, y)$ and $g(x, y)$ are nonconvex in (P1), or when nonlinear equalities, $h(x, y) = 0$, are present, two difficulties arise. First, the NLP subproblems (NLP1), (NLP2), (NLPF) may not have a unique local optimum solution. Second, the master problem (M-MIP) and its variants (e.g. M-MIPF, M-GBD, M-MIQP), do not guarantee a valid lower bound Z_{L}^K or a valid bounding representation with which the global optimum may be cut off. One possible approach to circumvent this problem is reformulation. This, however, is restricted to special cases, most notably in geometric programming constraints (polynomials) in which exponential transformations, $u = \exp(x)$, can be applied for convexification.

One general solution approach for handling nonconvexities is to develop rigorous global optimization algorithms, that assume specific forms of the nonlinearities (e.g. bilinear, linear fractional, concave separable) as will be discussed in the Perspectives article. The other option for handling nonconvexities is to apply a heuristic strategy to try to reduce as much as possible the effect of nonconvexities. While not being rigorous, this requires much less computational effort. We will describe here an approach for reducing the effect of nonconvexities at the level of the MILP master problem.

Viswanathan and Grossmann (1990) proposed to introduce slacks in the MILP master problem to reduce the likelihood of cutting-off feasible solutions. This master problem "augmented penalty/equality relaxation" (APER) has the form:

$$\mathrm{Min}\ Z^K = \alpha + \sum_{k=1}^{K} [w_p^k p^k + w_q^k q^k] \quad \text{s.t.} \begin{Bmatrix} \alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k)^{\mathrm{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ T^k \nabla h(x^k, y^k)^{\mathrm{T}} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq p^k \\ g(x^k, y^k) + \nabla g(x^k, y^k)^{T} \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq q^k \\ \sum_{i \in B^k} y_i - \sum_{i \in N^k} y_i \leq |B^k| - 1 \\ x \in X,\ y \in Y,\ \alpha \in R^1,\ p^k, q^k \geq 0 \end{Bmatrix} \quad k = 1, \ldots, K \qquad \text{(M-APER)}$$

where w_p^k, w_q^k are weights that are chosen sufficiently large (e.g. 1000 times magnitude of Lagrange multiplier). Note that if the functions are convex then the MILP master problem (M-APER) predicts rigorous lower bounds to (P1) since all the slacks are set to zero.

It should also be noted that another modification to reduce the undesirable effects of nonconvexities in the master problem is to apply global convexity tests followed by a suitable validation of linearizations. One possibility is to apply the tests to all linearizations with respect to the current solution vector (y^k, x^k) (Grossmann & Kravanja, 1997).

3.4. Computer codes for MINLP

The number of computer codes for solving MINLP problems is still rather small. The program DICOPT (Viswanathan & Grossmann, 1990) is an MINLP solver that is available in the modeling system GAMS (Brooke et al., 1998). The code is based on the master problem (M-APER) and the NLP subproblems (NLP2). This code also uses the relaxed (NLP1) to generate the first linearization for the above master problem, with which the user need not specify an initial integer value. Also, since bounding properties of (M−APER) cannot be guaranteed, the search for nonconvex problems is terminated when there is no further improvement in the feasible NLP subproblems. This is a heuristic that works reasonably well in many problems. Codes that implement the branch-and-bound method using subproblems (NLP1) include the code MINLP_BB that is based on an SQP algorithm (Leyffer, 2001) and is available in AMPL, the code BARON (Sahinidis, 1996) that also implements global optimization capabilities, and the code

SBB which is available in GAMS (Brooke et al., 1998). The code α-ECP implements the extended cutting plane method by Westerlund and Pettersson (1995), including the extension by Pörn and Westerlund (2000). Finally, the code MINOPT (Schweiger & Floudas, 1998) also implements the OA and GBD methods, and applies them to mixed-integer dynamic optimization problems. It is difficult to make general remarks on the efficiency and reliability of all these codes and their corresponding methods since no systematic comparison has been made. However, one might anticipate that branch and bound codes are likely to perform better if the relaxation of the MINLP is tight. Decomposition methods based on OA are likely to perform better if the NLP subproblems are relatively expensive to solve, while GBD can perform with some efficiency if the MINLP is tight, and there are many discrete variables. ECP methods tend to perform well on mostly linear problems.

4. Dynamic optimization

4.1. DAE optimization—problem statement

Interest in dynamic simulation and optimization of chemical processes has increased significantly during the last two decades. Chemical processes are modeled dynamically using DAEs, consisting of differential equations that describe the dynamic behavior of the system, such as mass and energy balances, and algebraic equations that ensure physical and thermodynamic relations. Typical applications include control and scheduling of batch processes; startup, upset, shutdown and transient analysis; safety studies and the evaluation of control schemes. We state a general differential–algebraic optimization problem (DAOP) as follows:

$$\text{Min } \Phi(z(t_f);\, y(t_f);\, u(t_f);\, t_f;\, p) \quad \text{s.t.} \begin{cases} F(\mathrm{d}z/\mathrm{d}t;\, z(t);\, y(t);\, u(t);\, t;\, p) = 0,\; z(0) = z_0 \\ G_s(z(t_s);\, y(t_s);\, u(t_s);\, t_s;\, p)) = 0 \\ z^L \le z(t) \le z^U \\ y^L \le y(t) \le y^U \\ u^L \le u(t) \le u^U \\ p^L \le p \le p^U \\ t_f^L \le t_f \le t_f^U \end{cases} \qquad \text{(DAOP)}$$

where Φ is a scalar objective function at final time, t_f, F are DAE constraints, G_s's are additional point conditions at times t_s's, z's are differential state profile vectors, y's are algebraic state profile vectors, u's are control state profile vectors and p is a time-independent parameter vector.

We assume, without loss of generality, that the index of the DAE system is one, consistent initial conditions are available and that the objective function is in the Mayer form. Otherwise, it is easy to reformulate problems to this form. Problem (DAOP) can be solved either by the variational approach or by applying some level of discretization that converts the original continuous time problem into a discrete problem. Early solution strategies, known as indirect methods, were focused on solving the classical variational conditions for optimality. On the other hand, methods that discretize the original continuous time formulation can be divided into two categories, according to the level of discretization. Here, we distinguish between the methods that discretize only the control profiles (partial discretization) and those that discretize the state and control profiles (full discretization). Basically, the partially discretized problem can be solved either by dynamic programming or by applying a nonlinear programming (NLP) strategy (direct-sequential). A basic characteristic of these methods is that a feasible solution of the DAE system, for given control values, is obtained by integration at every iteration of the NLP solver. The main advantage of these approaches is that, for the NLP solver, they generate smaller discrete problems than full discretization methods.

Methods that fully discretize the continuous time problem also apply NLP strategies to solve the discrete system and are known as direct-simultaneous methods. These methods can use different NLP and discretization techniques but the basic characteristic is that they solve the DAE system only once, at the optimum. In addition, they have better stability properties than partial discretization methods, especially in the presence of unstable dynamic modes. On the other hand, the discrete problem is larger and requires large-scale NLP solvers.

With this classification we take into account the degree of discretization used by the different methods. Below we briefly present the description of the variational methods, followed by methods that partially discretize the dynamic optimization problem, and finally we consider full discretization methods for problem (DAOP).

4.2. Variational methods

These methods are based on the solution of the first order necessary conditions for optimality that are obtained from Pontryagin's Maximum Principle (Pontryagin, Boltyanskii, Gamkredlidze, & Mishchenko, 1962). If we consider a version of (DAOP) without bounds, the optimality conditions are formulated as a set of DAEs:

$$\frac{\partial F(z, y, u, p, t)}{\partial z'}\lambda' = \frac{\partial H}{\partial z} = \frac{\partial F(z, y, u, p, t)}{\partial z}\lambda \qquad \text{(VCa)}$$

$$F(z, y, u, p, t) = 0 \qquad \text{(VCb)}$$

$$G_f(z, y, u, p, t_f) = 0 \quad \text{(VCc)}$$

$$G_s(z, y, u, p, t_s) = 0 \quad \text{(VCd)}$$

$$\frac{\partial H}{\partial y} = \frac{\partial F(z, y, u, p, t)}{\partial y}\lambda = 0 \quad \text{(VCe)}$$

$$\frac{\partial H}{\partial u} = \frac{\partial F(z, y, u, p, t)}{\partial u}\lambda = 0 \quad \text{(VCf)}$$

$$\int_0^{t_f} \frac{\partial F(z, y, u, p, t)}{\partial p}\lambda \, dt = 0 \quad \text{(VCg)}$$

where the Hamiltonian, H, is a scalar function of the form $H(t) = F(z, y, u, p, t)^T\lambda(t)$ and $\lambda(t)$ is a vector of adjoint variables. Boundary and jump conditions for the adjoint variables are given by

$$\begin{aligned} &\frac{\partial F}{\partial z'}\lambda(t_f) + \frac{\partial \Phi}{\partial z} + \frac{\partial G_f}{\partial z}v_f = 0 \\ &\frac{\partial F}{\partial z'}\lambda(t_s^-) + \frac{\partial G_s}{\partial z}v_s = \frac{\partial F}{\partial z'}\lambda(t_s^+) \end{aligned} \quad \text{(VBC)}$$

where v_f, v_s are the multipliers associated with the final time and point constraints, respectively. The most expensive step lies in obtaining a solution to this boundary value problem. Normally, the state variables are given as initial conditions and the adjoint variables as final conditions. This formulation leads to boundary value problems (BVPs) that can be solved by a number of standard methods including single shooting, invariant embedding, multiple shooting or some discretization method such as collocation on finite elements or finite differences. Also the point conditions lead to an additional calculation loop to determine the multipliers v_f and v_s. On the other hand, when bound constraints are considered, the above conditions are augmented with additional multipliers and associated complementarity conditions. Solving the resulting system leads to a combinatorial problem that is prohibitively expensive except for small problems.

4.3. Partial discretization

These strategies consider a discretization of the control profile $u(t)$ in (DAOP). Two strategies are usually considered, one based on dynamic programming and the other based on nonlinear programming.

4.3.1. Dynamic programming

Iterative dynamic programming (IDP) for the solution of dynamic optimization problems has been limited to small problems. However, this approach can be made efficient (Bojko & Luus, 1992; Luus, 1993) by allowing a coarse solution grid, which in some cases can be accurate enough to represent a solution to (DAOP). Although the IDP algorithm is slower than most gradient-based algorithms, it can be useful to crosscheck results of relatively small problems and it may avoid local solutions. Here the probability of obtaining the global optimum is usually high if the grid is well chosen (Dadebo & McAuley, 1995). For these techniques the time horizon is divided into P time stages, each of length L. Then, the control variables are usually represented as piecewise constant or piecewise linear functions in each interval.

The functions in each interval $(t_i; t_{i+1})$, usually take the form: $u(t) = u_i + ((u_{i+1} - u_i)/L)(t_{i+1} - t_i)$ where u_i and u_{i+1} are the values of u at t_i and t_{i+1}, respectively. The dynamic optimization problem is to find u_i; $i = 1, \ldots, P$ that minimize a given objective function. The basic search algorithm mimics the classical dynamic programming algorithm starting at the last stage with a discrete set of control values. For a set of input states, the best control is chosen at each stage and the algorithm proceeds forward to a previous stage. Once all of the stages are considered, the discrete set of control values is narrowed around the best set of values and the process repeats. More details of this approach can be found in (Dadebo & McAuley, 1995; Luus, 1993). The IDP algorithm works well when the dynamic optimization problem does not include bounds on state variables. In order to include such bounds, a penalty term has to be added into the objective function to penalize the constraint violation. This can be done by adding a state variable for each inequality that measures the constraint violation over time (Mekarapiruk & Luus, 1997) or by computing the constraint violation at given points in time (Dadebo & McAuley, 1995).

4.3.2. Direct sequential methods

With partial discretization methods (also called sequential methods or control vector parameterization), only the control variables are discretized. Given the initial conditions and a given set of control parameters, the DAE system is solved with a differential algebraic equation solver at each iteration. This produces the value of the objective function, which is used by a NLP solver to find the optimal parameters in the control parameterization, υ. The sequential method is reliable when the system contains only stable modes. If this is not the case, finding a feasible solution for a given set of control parameters can be difficult. The time horizon is divided into time stages and at each stage the control variables are represented with a piecewise constant, a piecewise linear or a polynomial approximation (Feehery & Barton, 1998; Vassiliadis, 1993). A common practice is to represent the controls as a set of Lagrange interpolation polynomials.

For the NLP solver, gradients of the objective and constraint functions with respect to the control parameters can be calculated with the sensitivity equations of the DAE system, given by

$$\begin{aligned} &\frac{\partial F}{\partial z'}^T s_k' + \frac{\partial F}{\partial z}^T s_k + \frac{\partial F}{\partial y}^T w_k + \frac{\partial F}{\partial q_k}^T = 0, \\ &s_k(0) = \frac{\partial z(0)}{\partial q_k}, \quad k = 1, \ldots, N_q \end{aligned} \quad \text{(SE)}$$

where $s_k(t) = \partial z(t)/\partial q_k$, $w_k(t) = \partial y(t)/\partial q_k$ and $q^T = [p^T, \upsilon^T]$. As can be inferred from (SE), the cost of obtaining

these sensitivities is directly proportional to N_q, the number of decision variables in the NLP. Alternately, gradients can be obtained by integration of the adjoint equations (VCa, VCe, VBC) (Bryson & Ho, 1969; Hasdorff, 1976; Sargent & Sullivan, 1979) at a cost independent of the number of input variables and proportional to the number of constraints in the NLP.

The methods that are based on this approach cannot treat directly the bounds on state variables because the state variables are not included in the nonlinear programming problem. Instead, most of the techniques for dealing with inequality path constraints rely on defining a measure of the constraint violation over the entire horizon, and then penalizing it in the objective function, or forcing it directly to zero through an end-point constraint (Vassiliadis, Sargent, & Pantelides, 1994). Other techniques approximate the constraint satisfaction by introducing an exact penalty function (Bloss, Biegler & Schiesser, 1999; Sargent & Sullivan, 1979) or a Kreisselmeier–Steinhauser function (Bloss et al., 1999) into the problem.

Finally, initial value solvers that handle path constraints directly have been developed in Feehery and Barton (1998). The main idea is to use an algorithm for constrained dynamic simulation so that any admissible combination of the control parameters produces an initial value problem that is feasible with respect to the path constraints. The algorithm proceeds by detecting activation and deactivation of the constraints during the solution, and solving the resulting high-index DAE system and their related sensitivities.

4.4. Full discretization

Full discretization methods explicitly discretize all the variables of the DAE system and generate a large scale nonlinear programming problem that is usually solved with a successive quadratic programming (SQP) algorithm. These methods follow a simultaneous approach (or infeasible path approach); that is, the DAE system is not solved at every iteration, it is only solved at the optimum point. Because of the size of the problem, special decomposition strategies are used to solve the NLP efficiently. Despite this characteristic, the simultaneous approach has advantages for problems with state variable (or path) constraints and for systems where instabilities occur for a range of inputs. In addition, the simultaneous approach can avoid intermediate solutions that may not exist, are difficult to obtain, or require excessive computational effort. There are mainly two different approaches to discretize the state variables explicitly, multiple shooting (Bock & Plitt, 1984; Leineweber et al., 1997) and collocation on finite elements (Betts, 2001; Biegler, Cervantes, & Wächter, 2002; Cuthrell & Biegler, 1987).

4.4.1. Multiple shooting

With multiple shooting, time is discretized into P stages and control variables are parameterized using a finite set of control parameters in each stage, as with partial discretization. The DAE system is solved on each stage, $i = 1, \ldots, P$ and the values of the state variables $z(t_i)$ are chosen as additional unknowns. In this way a set of relaxed, decoupled initial value problems (IVP) is obtained, as follows:

$$F(\mathrm{d}z/\mathrm{d}t; z(t); y(t); v_i; p) = 0, \quad t \in [t_{i-1}, t_i], \; z(t_{i-1}) = z_i$$
$$z_{i+1} - z(t_i; z_i; \upsilon_i; p) = 0, \quad i = 1, \ldots, P-1$$

Note that continuity between stages is treated through equality constraints, so that the final solution satisfies the DAE system. With this approach, inequality constraints for states and controls can be imposed directly at the grid points, but path constraints for the states may not be satisfied between grid points. This problem can be avoided by applying penalty techniques to enforce feasibility, like the ones used in the sequential methods.

The resulting NLP is solved using SQP-type methods, as described above. At each SQP iteration, the DAEs are integrated in each stage and objective and constraint gradients with respect to p, z_i and υ_i are obtained using sensitivity equations, as in (SE). Compared to sequential methods, the NLP contains many more variables but efficient decompositions have been proposed (Leineweber et al., 1997) and many of these calculations can be performed in parallel.

4.4.2. Collocation methods

In this formulation, the continuous time problem is converted into an NLP by approximating the profiles as a family of polynomials on finite elements. Various polynomial representations are used in the literature, including Lagrange interpolation polynomials for the differential and algebraic profiles (see Cuthrell & Biegler, 1987). In Betts (2001), a Hermite–Simpson collocation form is used while Cervantes and Biegler (1998) and Tanartkit and Biegler (1995) use a monomial basis representation (Bader & Ascher, 1987) for the differential profiles. All of these representations stem from implicit Runge–Kutta formulae and the monomial representation is recommended because of smaller condition numbers and smaller rounding errors. On the other hand, control and algebraic profiles are approximated using Lagrange polynomials.

Discretizations of (DAOP) using collocation formulations lead to the largest NLP problems but these can be solved efficiently using large-scale NLP solvers, such as IPOPT and by exploiting the structure of the collocation equations. Biegler et al. (2002) provide a review of dynamic optimization methods using simultaneous methods. These methods offer a number of advantages for challenging dynamic optimization problems, including:

- Control variables can be discretized at the same level of accuracy as the differential and algebraic state variables. Finite elements allow for discontinuities in control profiles.
- Collocation formulations allow problems with unstable modes to be handled in an efficient and well-conditioned manner. The NLP formulation inherits stability properties

of boundary value solvers. Moreover, an element-wise decomposition has been developed that pins down unstable modes in (DAOP).
- Collocation formulations have been proposed with moving finite elements. This allows the placement of elements both for accurate breakpoint locations of control profiles as well as accurate DAE solutions.

Dynamic optimization using collocation methods has been used for a number of process applications including batch process optimization (Bhatia & Biegler, 1996), nonlinear model predictive control (Albuquerque et al., 1997), grade transitions and process changeovers (Cervantes, Tonelli, Brandolin, Bandoni, & Biegler, 2002) and reactor design and synthesis (Lakshmanan, Rooney, & Biegler, 2000; Ierapetritou, 2001).

4.5. Extensions for dynamic optimization

Here, we briefly summarize a few issues that emerge for dynamic optimization. These extend the methods presented so far to larger and more challenging applications and include discrete decisions, the treatment of multistage dynamic systems and fundamental questions on the accuracy of discretized optimal control problems.

4.5.1. Discrete decisions in dynamic optimization

Along with the DAE models described in (2) and (3), it becomes important to consider the modeling of discrete events in many dynamic simulation and optimization problems. In chemical processes, examples of this phenomena include phase changes in vapor–liquid equilibrium systems, changes in modes in the operation of safety and relief valves, vessels running dry or overflowing, discrete decisions made by control systems and explosions due to accidents. These actions can be reversible or irreversible with the state profiles and should be modeled with appropriate logical constraints. An interesting presentation on modeling discrete events can be found in Allgor and Barton (1999). The simulation of these events is often triggered by an appropriate discontinuity function which monitors a change in the condition and leads to a change in the state equations. These changes can be reformulated either by using complementarity conditions (with positive continuous variables x and y alternately set to zero) or as binary decision variables (Barton & Park, 1997). These additional variables can then be embedded within optimization problems. Here complementarity conditions can be reformulated through barrier methods (Raghunathan & Biegler, 2002) to yield an NLP while the incorporation of integer variables leads to mixed-integer optimization problems.

For the latter case, several studies have considered the solution of mixed-integer dynamic optimization (MIDO) problems. In particular, Avraam, Shah, and Pantelides (1998) developed a complete discretization of the state and control variables to form a MINLP. On the other hand, Allgor and Barton (1999) apply a sequential strategy and discretize only the control profile. In this case, careful attention is paid to the calculation of sensitivity information across discrete decisions that are triggered in time.

4.5.2. Multistage applications

The ability to solve large dynamic optimization problems and to model discrete decisions allows the integration of multiple dynamic systems for design and analysis. Here, different dynamic stages of operation can be considered with individual models for each dynamic stage. Multistage applications in process engineering include startups and transients in dynamic systems with different modes of operation, design and operation of periodic processes with different models (e.g., adsorption, regeneration, pressurization, in a dynamic cycle (Nilchan & Pantelides, 1998)), synthesis of chemical reactor networks (Lakshmanan & Biegler, 1995), changes in physical phenomena due to discrete changes (as seen above) and multiproduct and multiperiod batch plants where scheduling and dynamics need to be combined and different sequences and dynamic operations need to be optimized. For these applications each stage is described by separate state variables and models as in equations (2) and (3). These stages include an overall objective function with parameters linking among stages and control profiles that are manipulated within each stage. Moreover, multistage models need to incorporate transitions between dynamic stages. These can include logical conditions and transitions to multiple models for different operation. Moreover, the DAE models for each stage require consistent initializations across profile discontinuities, triggered by discrete decisions.

The solution of multistage optimization problems has been considered in a number of recent studies. Bhatia and Biegler (1996) consider the simultaneous design, operation and scheduling of a multiproduct batch plant by solving a large NLP. More recently, multistage problems have been considered as mixed-integer problems using sequential strategies as well as simultaneous strategies. These applications only represent the initial stages of dynamic systems modeling, in order to deal with an integrated analysis and optimization of large scale process models. With the development of more efficient decomposition and solution strategies for dynamic optimization, much more challenging and diverse multistage applications will continue to be considered.

4.5.3. Improved formulations for dynamic optimization

For optimal control problems where control variables are discretized at the same level as the state variables, there are a number of open questions related to convergence to the solution of the original variational problem. A number of studies have shown (e.g., Cuthrell & Biegler, 1989; Polak, 1997; Reddien, 1979; Schwartz, 1996) that the KKT conditions of the simultaneous NLP can be made consistent with the optimality conditions of the variational problem. Nevertheless, these consistency properties do not guarantee conver-

gence to solution of the infinite dimensional optimal control problem and several studies report stability problems due to poor discretizations, high index constraints and singular arcs. In particular, interesting stability questions arise regarding appropriate discretizations of control and state profiles. Empirical evidence of this instability and practical remedies have been given in Logsdon and Biegler (1989), Bausa and Tsatsaronis (2001) and special cases of these have been analyzed rigorously in Dontchev et al. (2000). In a recent thesis, Biehn (2001) showed that for continuous, convex optimal control problems, two simple simultaneous collocation formulations have desirable consistency properties. Moreover, his analysis has shown that these formulations remain stable in the presence of high index constraints, *even when sequential* (*initial value*) *solvers fail* on these problems. In related work, Schwartz (1996) developed consistency results for *explicit* Runge–Kutta discretizations that apply to more challenging optimal control problems with singular arcs and discontinuous profiles.

5. Optimization under uncertainty

All the previous optimization problems that we have reviewed are deterministic in nature. However, as one might expect, there is often significant uncertainty in application of optimization in the real world. Failure to account for the uncertainty of key parameters (e.g., technical coefficients, product demands) has the drawback that the solution of deterministic models can lead to non-optimal or infeasible decisions. This, however, does not mean that deterministic models are not useful. In fact, as will be seen, they are used as a basis in virtually any stochastic optimization method, or methods for flexibility analysis.

Considerable theoretical work has been reported in the *Operations Research* literature on the formulation and solution of linear stochastic optimization problems (see reviews by Birge, 1992; Dantzig, 1987; Dempster, 1980; Wets, 1989). We provide here only a very brief review. An excellent recent review can be found in Sahinidis (2003).

Extending deterministic models with probabilistic representations leads to the stochastic programming model. The most common linear model is the following two-stage (fixed recourse) stochastic LP:

$$\begin{aligned} \text{Min} \quad & z = c_1^{\mathrm{T}} x_1 + \sum_{k \in K} p_{2k} c_{2k}^{\mathrm{T}} x_{2k} \\ & A_1 x_1 = b_1 \\ \text{s.t.} \quad & B_1 x_1 + A_2 x_{2k} = b_{2k} \quad \forall k \in K \\ & 0 \le x_1 \le U_{k_1} \\ & 0 \le x_1 \le U_{k_1} \quad \forall k \in K \end{aligned} \tag{SLP}$$

where matrices B_1 and A_2 are fixed (i.e., $B_{1k} = B_1$ and $A_{2k} = A_2$ $\forall k \in K$). The term K denotes the set of possible stage 2 events defined on the finite, discrete probability space. This problem is important because: (i) it is representative of the multi-stage model in terms of probabilistic expansion of variables and constraints, and (ii) it is the key structural component to the multi-stage problem and is the key subproblem for the nested decomposition algorithms used to solve the multi-stage LP.

The study of the theory and solution of the multi-stage stochastic LP (MSLP) has paralleled the development of deterministic LP methods. Early references included seminal work on the formulation and problem structure (Dantzig, 1955; Dantzig, 1963; Dempster, 1965; Madansky, 1963; Rosen, 1963; Wets, 1966), but left questions concerning the solution to the general problem largely unanswered. Since the certainty equivalent LP, expanded to multi-stage as needed, is intractably large for all but the smallest problems (see Dantzig, 1987 for discussion of exponential expansion), current solution methods use Benders-based decomposition strategies (Benders, 1962; Geoffrion, 1972; Van Slyke and Wets, 1969). See Dantzig (1987) or Birge (1982a) for a discussion of the general multi-stage stochastic LP formulations. Comprehensive reviews of theory and solution practices are provided in the collections edited by Dempster (1980) and Ermoliev and Wets (1988). Spurred in part by the expansion in computing power, recent progress has been made in solving the two-stage stochastic linear programming problem using Benders-based schemes (see e.g., Dantzig & Glynn, 1989; Gassmann, 1990; Infanger, 1991; Wets, 1983; Wets, 1989). Extension to multi-stage problems via nested decomposition methods is conceptually straightforward. The multi-stage problem however remains intractable due to computational expense, arising from the nested structure of the problem and resultant exponential growth in the number of subproblems (see Birge, 1982a; Dantzig, 1987; Dempster, 1980; Gassmann, 1990; Louveaux, 1986). While a few specialized problems have been addressed (see Beale, Forrest, & Taylor, 1980; Bienstock and Shapiro, 1985; Dantzig, 1987; Karreman, 1963), general multi-stage linear problems remain computationally intractable. Multi-stage solution methods generally rely on nested decomposition strategies which involve solving series of two-stage subproblems (Birge, 1982a; Ermoliev & Wets, 1988; Gassmann, 1990). Hence, advances in the solution to two-stage models are applicable toward improving multi-stage solution methods. Conceptually the extension to nonlinear stochastic problems is similar as in the linear case. The extension to stochastic mixed-integer problems is considerably more difficult (see Sahinidis, 2003).

5.1. Process flexibility

In contrast to the stochastic optimization approach, considerable effort has been devoted in process systems engineering over the last 25 years to developing methods for evaluating and optimizing flexibility. The major goal has been to address nonlinear optimization problems under uncertainty, particularly design problems (see Grossmann &

Straub, 1991). The proposed approaches can be classified in two broad classes: (i) deterministic, in which the parameter uncertainty is typically described through bounds of expected deviations, and (ii) stochastic, that describes the uncertainty through a probability distribution function. Here, we only review the deterministic flexibility analysis.

The model of the process can be described, in the case where the topology is fixed, by a set of equations and inequalities involving continuous variables of the form:

$$\begin{aligned} h(d, z, x, \theta) &= 0 \\ g(d, z, x, \theta) &\leq 0 \end{aligned} \tag{F0}$$

where the variables are defined as follows: $d \in R^{n_d}$ denotes an n_d vector of stage 1 variables (e.g. design variables) that defines the structure and equipment sizes of the process, $z \in R^{n_z}$ denotes an n_z vector of stage 2 variables (e.g. control variables) that can be adjusted during plant operation (e.g. flows, pressures), $x \in R^{n_x}$ denotes an n_x vector of state variables that describes the behavior of the process (e.g. flows, pressures, temperatures, reactor conversions), $\theta \in R^{n_\theta}$ denotes an n_θ vector of uncertain parameters (e.g. feed composition, kinetic constants).

For simplicity in the presentation and consistency with the existing literature (Grossmann & Floudas, 1987), it is assumed that the state variables in (F0) are eliminated from the equations and thus the model reduces to

$$f_j(z, \theta, d) \leq 0, \quad j \in J$$

Note, however, that in the development of the proposed methodology this projection will not be necessary.

For a given design d, the first important question is to determine whether this design is feasible for a realization of the uncertain parameters θ also known as the *feasibility problem* (F1). The formulation of this problem (Halemane & Grossmann, 1983) is:

$$\psi(\theta, d) = \underset{z,u}{\text{Min}}\ u \text{ s.t. } f_j(z, \theta, d) \leq u,\ j \in J;\ u \in R^1 \tag{F1}$$

Note that problem (F1) is an optimization problem where the objective is to find a point z^*, for fixed d and θ, such that the maximum potential constraint violation is minimized. However, u is in principle a function of d and θ, and expressed in that form it represents the projected feasibility function. The *projected feasibility function* $\psi(\theta, d)$ is a key concept in the flexibility analysis and its construction is an important and challenging task. As can be deduced from (F1), $\psi \leq 0$ indicates feasibility and $\psi > 0$, infeasibility.

The problem of evaluating flexibility over a specified set T of uncertain parameters, also known as the *flexibility test*, corresponds to the finding the worst value of θ in the set T, which gives rise to the maximization problem,

$$\chi(d) = \max_{\theta \in T}\ \psi(d, \theta) \tag{F2}$$

which is also equivalent to the Max–Min–Max optimization problem (Halemane & Grossmann, 1983),

$$\chi(d) = \underset{\theta \in T}{\text{Max}}\ \underset{z}{\text{Min}}\ \underset{j \in J}{\text{Max}}\ f_j(d, z, \theta) \tag{F2$'$}$$

where a common description of T is $T = \{\theta | \theta^{\text{L}} \leq \theta \leq \theta^{\text{U}}\}$, where $\theta^{\text{L}}, \theta^{\text{U}}$ are lower and upper bounds, respectively. Other descriptions of T such as hypercircles or hyper-ellipsoids can also be easily used.

The more general problem of quantifying flexibility, also known as the *flexibility index problem* (F3), is to determine the maximum deviation Δ that a given design d can tolerate, such that every point θ in the uncertain parameter space, $T(\delta)$, is feasible. The most common choice is the hyper-rectangle parametric in δ, $T(\delta) = \{\theta | \theta^N - \delta\Delta\theta^- \leq \theta \leq \theta^N + \delta\Delta\theta^+\}$, where $\Delta\theta^+$ and $\Delta\theta^-$ are the expected deviations of uncertain parameters in the positive and negative direction. Other descriptions of $T(\delta)$, such as the parametric hyper-ellipsoid, are also possible (see Rooney & Biegler, 1999).

As shown by Swaney and Grossmann (1985a), the flexibility index can be determined from the formulation,

$$F = \text{Max } \delta \text{ s.t. } \begin{cases} \text{Max}_{\theta \in T(\delta)}\ \psi(\theta, d) \leq 0 \\ \delta \geq 0,\ \delta \in R^1 \end{cases} \tag{F3}$$

As seen from the implicit form of the *projected feasibility function* $\psi(\theta, d)$, problem (F3) cannot be directly solved unless ψ is determined. The simplest way around this problem (see Swaney & Grossmann, 1985b) is to determine the flexibility index in (F3) by vertex enumeration search in which the maximum displacement is computed along each vertex direction, thus avoiding the explicit construction of ψ. This vertex enumeration scheme relies on the assumption that the critical points θ^* lie at the vertices of $T(\Delta^*)$, which is valid for the case of a linear model and in general only if certain convexity conditions hold. The drawback with this approach, however, is that it requires the solution of $2n\theta$ optimization problems, and therefore, it scales exponentially with the number of uncertain parameters.

An alternative method for evaluating the flexibility index that does not rely on the assumption that critical points correspond to vertices, is the active set strategy by Grossmann and Floudas (1987). In this method the key idea is that the feasible region projected into the space of d and θ, can be expressed in terms of active sets of constraints $f_j(z, \theta, d) = u$, $j \in J_A^k$, $k = 1, n_{\text{AS}}$, where n_{AS} is the number of possible active sets of f_j. These active sets are defined by all subsets of non-zero multipliers that satisfy the Kuhn–Tucker conditions of (F1):

$$\sum_{j \in J_A^k} \lambda_j^k = 1$$

$$\sum_{j \in J_A^k} \lambda_j^k \frac{\partial f_j}{\partial z} = 0$$

By reformulating problem (F3) for evaluating the flexibility index, and using the above equations with 0–1 variables for the complementarity conditions and slacks, we get a mixed integer optimization problem that can explicitly solve (F3) without having to find a priori all the active sets.

$$
\begin{aligned}
F = \min_{\delta,\lambda_j,s_j,y_j} \quad & \delta \\
& f_j(d,z,\theta) + s_j = 0, \ \ j \in J \\
& \textstyle\sum_{j\in J} \lambda_j = 1 \\
& \textstyle\sum_{j\in J} \lambda_j \dfrac{\partial f_j}{\partial z} = 0 \\
\text{s.t.} \quad & s_j - U(1-y_j) \le 0, \ \ j \in J \\
& \lambda_j - y_j \le 0 \\
& \textstyle\sum_{j\in J} y_j = n_z + 1 \\
& \theta^N - \delta\Delta\theta^- \le \theta \le \theta^N + \delta\Delta\theta^+ \\
& \delta \ge 0,\ \lambda_j,\ s_j \ge 0,\ \ j \in J;\ y_j = 0,1\ j \in J
\end{aligned}
\tag{ASF}
$$

This model (ASF) gives rise to an MINLP problem (or MILP if all constraints are linear) with $n_f = \text{card}\{J\}$ binary variables.

As for nonlinear optimization problems under uncertainty they involve the selection of the stage 1 variables d (i.e. design variables) so as to minimize cost and either (a) satisfy the flexibility test (F2), or (b) maximize the flexibility index as given by (F3), where the latter problem gives rise to a multiobjective optimization problem.

Most of the previous work in design under uncertainty (Johns, Marketos, & Rippin, 1976; Malik & Hughes, 1979) has considered the effect of the continuous uncertain parameters θ for the design optimization through the minimization of the expected value of the cost using a two-stage strategy, similar as the one in problem (SLP), but for continuous distribution functions, is given by

$$
\min_d \ \mathop{E}_{\theta\in T(F)} [\min_z \ C(d,z,\theta) | f(d,z,\theta) \le 0] \tag{SNLP}
$$

The reason the above also requires a two-stage strategy is because the design variables d are chosen in stage 1 and remain fixed during stage 2 during which the control variables z are adjusted depending on the realizations of the parameters θ. In order to handle infeasibilities in the inner minimization, one approach is to assign penalties for the violation of constraints (e.g. $C(d,z,\theta) = \bar{C}$ if $f(d,z,\theta) > 0$. The other approach is to enforce feasibility for a specified flexibility index F (e.g. see Halemane and Grossmann, 1993) through the parameter set $T(F) = \{\theta | \theta^L - F\Delta\theta^- \le \theta \le \theta^U + F\Delta\theta^+\}$. In this case (SNLP) is formulated as

$$
\begin{aligned}
\min_d \ & \mathop{E}_{\theta\in T(F)} [\min_z \ C(d,z,\theta) | f(d,z,\theta) \le 0] \\
\text{s.t.} \ & \max_{\theta\in T(F)} \ \psi(d,\theta) \le 0
\end{aligned}
\tag{SNLPF}
$$

A particular case of (SNLP) occurs when only a discrete set of points θ^k, $k = 1,\ldots,K$ are specified which then gives rise to the optimal design problem,

$$
\begin{aligned}
\min_{d,z^1,\ldots z^K} \ & \sum_{k=1}^{K} w_k C(d,z^k,\theta^k) \\
\text{s.t.} \ & f(d,z^k,\theta^k) \le 0, \quad k = 1,\ldots,K
\end{aligned}
\tag{DSNLP}
$$

where w_k are weights that are assigned to each point θ^k, and $\sum_{k=1}^{K} w_k = 1$.

Problem (DSNLP) can be interpreted as problem under uncertainty with discrete probabilities, which is also equivalent to a multiperiod problem, which is also of great importance in the optimal design of flexible chemical plants (see Grossmann & Sargent, 1979; Varvarezos, Grossmann & Biegler, 1992, 1993). As shown by Grossmann and Sargent (1978) problem (DSNLP) can also be used to approximate the solution of (SNLPF). This is accomplished by selecting an initial set of points θ^k, solving problem (DSNLP) and verifying its feasibility over $T(F)$ by solving problem (F2) or (F3). If the design is feasible the procedure terminates. Otherwise the critical point obtained from the flexibility evaluation is included to the set of K points and the solution of (DSNLP) is repeated. Computational experience has shown that commonly one or two major iterations must be performed to achieve feasibility with this method. Ostrovsky, Volin, and Senyavinj (1997) has proposed an alternative method for the two-stage problem that simplifies the evaluation of flexibility.

Stochastic approaches for the evaluation of flexibility rely on the idea of using joint probability distribution functions, which are integrated over the feasible region in order to determine the probability that constraints be satisfied given that control variables can be manipulated (e.g. Straub & Grossmann, 1993; Pistikopoulos & Mazzuchi, 1990). For a recent review of stochastic flexibility see Pistikoploulos (2002).

6. Summary and conclusions

Research in the formulation, solution and analysis of mathematical programs has grown tremendously over the past 25 years. In 1980, optimization on engineering problems beyond linear programming was often viewed as a curious novelty without much benefit. Now optimization applications are essential in all areas of process systems engineering including design, identification, control, estimation, scheduling and planning. This paper offers a retrospective on relevant optimization methods that have been developed and applied over the past 25 years and reviews four broad areas. First, we deal with methods for continuous variable optimization and survey advances in nonlinear programming methods with and without derivative evaluations. Next we consider mixed-integer programming methods and cover a family of algorithms and extensions for MINLPs. Related to these two approaches is optimization with differential algebraic models. Over the past decade these challenging problems have been considered more frequently in the process industries through sequential and simultaneous methods. Finally, we survey methods to deal with the essential problem of optimization under uncertainty.

References

Albuquerque, J., Gopal, V., Staus, G., Biegler, L. T., & Ydstie, B. E. (1997). Interior point SQP strategies for large-scale structured process optimization problems. *Computers and Chemical Engineering, 23*, 283.

Allgor, R., & Barton, P. (1999). Mixed-integer dynamic optimization. I. Problem formulation. *Computers and Chemical Engineering, 23*(4–5), 457.

Avraam, M., Shah, N., & Pantelides, C. (1998). Modeling and optimization of general hybrid systems in continuous time domain. *Computers and Chemical Engineering, 22*, S221.

Bader, G., & Ascher, U. M. (1987). A new basis implementation for mixed order boundary value ODE solver. *SIAM Journal of Science and Computers, 8*, 483–500.

Balas, E., Ceria, S., & Cornuejols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming, 58*, 295–324.

Banga, J. R., & W. D. Seider (1996). Global optimization of chemical processes using stochastic algorithms. In C. Floudas & P. Pardalos (Eds.), *State of the art in global optimization* (p. 563). Dordrecht: Kluwer Academic Publishers.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*, 316–329.

Barton, P., & Park, T. (1997). Analysis and control of combined discrete/continuous systems: Progress and challenges in the chemical process industries. *AIChE Symposium Series, 93*(316), 102.

Bausa, J., & Tsatsaronis, G. (2001). Discretization methods for direct collocation of optimal control problems. *Computers and Chemical Engineering*, in press.

Bazaraa, M. S., & Sherali, H. D., & Shetty, C. M. (1994), *Nonlinear programming*. New York: Wiley.

Beale, E. M., Forrest, J. J. H., & Taylor, C. J. (1980). Multi-time-period stochastic programming. In M. A. H. Dempster (Ed.), *Stochastic programming* (pp. 387–402). New York: Academic Press.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerical Mathematics, 4*, 238–252.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathematic, 4*, 238–252.

Betts, J. T. (2001). Practical methods for optimal control using nonlinear programming. In *Advances in design and control* (Vol. 3). Philadelphia, USA: SIAM.

Betts, J. T. (2001). Practical methods for optimal control using nonlinear programming. *Advances in design and control* (Vol. 3), Philadelphia, USA: SIAM.

Bhatia, T., & Biegler, L. T. (1996). Dynamic optimization in the design and scheduling of multiproduct batch plants. *I&EC Research, 35*(7), 2234.

Biegler, L. T., Cervantes, A. M., & Wächter, A. (2002). Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science, 57*(4), 575–593.

Biehn, N. (2001). Ph.D. Thesis, Department of Mathematics, North Carolina State University.

Binder, T., Blank, L., Bock, H. G., Bulitsch, R., Dahmen, W., Diehl, M., Kronseder, T., Marquardt, W., Schloeder J., & von Stryk, O. (2001). Introduction to model based optimization of chemical processes on moving horizons. In Groetschel, M., Krumke, S. O., & Rambau, J. (Eds.), *Online optimization of large scale systems*. Berlin: Springer.

Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E., & Wunderling, R. (2002). *MIP: Theory and practice—closing the gap* (http://www.ilog.com/products/optimization/tech/research/mip.pdf).

Bloss, K. F., Biegler, L. T., & Schiesser, W. E. (1999). Dynamic process optimization through adjoint formulations and constraint aggregation. *Industrial Engineering in Chemical Research, 38*, 421–432.

Bock, H. G., & Plitt, K. J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC world congress, Budapest*.

Bojko, B., & Luus, R. (1992). Use of random admissible values for control in iterative dynamic programming. *Industrial Engineering in Chemical Research, 31*, 1308–1314.

Booker, A. J., Dennis Jr., J. E., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1998). A rigorous framework for optimization of expensive functions by surrogates. CRPC Technical Report 98739, Rice University.

Borchers, B., & Mitchell, J. E. (1994). An improved branch and bound algorithm for mixed-integer nonlinear programming. *Computers and Operations Research, 21*, 359–367.

Brooke, A., Kendrick, D., Meeraus, A., & Raman, R. (1998). *GAMS—A user's guide* (www.gams.com).

Bryson, A. E., & Ho, Y. C. (1969). *Applied optimal control: Optimization, estimation, and control*. Waltham, MA: Ginn and Company.

Byrd, R. H., Hribar, M. E., & Nocedal, J. (1997). *An interior point algorithm for large scale nonlinear programming*. Optimization Technology Center, Northwestern University.

Cervantes, A., & Biegler, L. T. (1998). Large-scale date optimization using simultaneous nonlinear programming formulations. *AIChE Journal, 44*, 1038.

Cervantes, A. M., Tonelli, S., Brandolin, A., Bandoni, J. A., & Biegler, L. T. (2002). Large-scale dynamic optimization for grade transitions in a low density polyethylene plant. *Computers and Chemical Engineering, 26*, 227.

Conn, A. R., Scheinberg, K., & Toint, P. (1997). Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming, Series B, 79*(3), 397.

Conn, A. R., Gould, N., & Toint, P. (2000). *Trust region methods*. Philadelphia, USA: SIAM.

Cuthrell, J. E., & Biegler, L. T. (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering, 13*(1–2), 49.

Cuthrell, J. E., & Biegler, L. T. (1987). On the optimization of differential–algebraic process systems. *AIChE Journal, 33*, 1257–1270.

Dadebo, S. A., & McAuley, K. B. (1995). Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Computers and Chemical Engineering, 19*(5), 513–525.

Dakin, R. J. (1965). A tree search algorithm for mixed-integer programming problems. *Computer Journal, 8*, 250–255.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton, NJ: Princeton University Press.

Das, H., Cummings, P., & LeVan, M. (1990). Scheduling of serial multiproduct batch processes via simulated annealing. *Computers and Chemical Engineering, 14*(12), 1351–1362.

Dempster, M. A. H. (1965). On stochastic programming, Ph.D. Thesis, Department of Mathematics, Carnegie Mellon University.

Dempster, M. A. H. (1980). Introduction to stochastic programming. In M. A. H. Dempster (Ed.), *Stochastic programming* (pp. 3–62). New York: Academic Press.

Dennis, J. E., & Torczon, V. (1991). Direct search methods on parallel machines. *SIAM Journal of Optics, 1*, 448.

Dennis, J. E., Heinkenschloss, M., & Vicente, L. N. (1998). Trust-region interior-point SQP algorithms for a class of nonlinear programming problems. *SIAM Journal on Control and Optimization, 36*, 1750–1794.

Ding-Mai, & Sargent (1992). A combined SQP and branch and bound algorithm for MINLP optimization. Internal Report, Centre for Process Systems Engineering, London.

Dolan, W. D., Cummings, P. T., & Levan, M. D. (1989). Process optimization via simulated annealing. *AIChE Journal, 35*, 725–736.

Duran, M. A., & Grossmann, I. E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming, 36*, 307.

Edgar, T. F., Himmelblau, D. M., & Lasdon, L. S. (2002). *Optimization of chemical processes*. New York: McGraw-Hill.

Edgar, T. F., Himmelblau, D. M., & Lasdon, L. S. (2001). *Optimization of chemical processes*. New York: McGraw-Hill.

Eldred, M. (2002). DAKOTA: A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis (http://endo.sandia.gov/DAKOTA/software.html).

Ermoliev, Y., & Wets, R. J.-B. (1988). *Numerical techniques for stochastic optimization*. New York: Springer-Verlag.

Feehery, W. F., & Barton, P. I. (1998). Dynamic optimization with state variable path constraints. *Computers and Chemical Engineering, 22*, 1241–1256.

Fletcher, R., Gould, N. I. M., Leyffer, S., Toint, Ph. L., & Waechter, A. (2001). Global convergence of a trust-region (SQP)-filter algorithms for general nonlinear programming. Department of Mathematics, University of Namur, Belgium (Revised).

Fletcher, R., & Leyffer, S. (1994). Solving mixed-integer nonlinear programs by outer approximation. *Mathematical Programming, 66*, 327.

Fletcher, R. (1987). *Practical methods of optimization*. Chichester: Wiley.

Flippo, O. E., & Kan, A. H. G. R. (1993). Decomposition in general mathematical programming. *Mathematical Programming, 60*, 361–382.

Floquet, P., Pibouleau, & Domenech, S. (1994). Separation system synthesis: How to use a simulated annealing procedure. *Computers and Chemical Engineering, 18*, 1141.

Fourer, R., Gay, D. M., & Kernighan, B. W. (1992). *AMPL: A modeling language for mathematical programming*. Belomont, CA: Duxbury Press.

Garrard, A., & Fraga, E. S. (1998). Mass exchange network synthesis using genetic algorithms. *Computers and Chemical Engineering, 22*, 1837.

Gassmann, H. I. (1990). MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming, 47* (pp. 407–423). Amsterdam: North-Holland.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of Optimization Theory and Applications, 10*(4), 237–260.

Gill, P. E., Murray, W., & Wright, M. (1981). *Practical optimization*. New York: Academic Press.

Gill, P. E., Murray, W., & Saunders, M. A. (1998). User's guide for SNOPT: A FORTRAN package for large-scale nonlinear programming. Technical report, Department of Mathematics, University of California, San Diego, USA, 1998.

Goulcher, R., & Cesares Long, J. (1978). The solution of steady state chemical engineering optimization problems using a random search algorithm. *Computers and Chemical Engineering, 2*, 23.

Grossmann, I. E., & Floudas, C. A. (1987). Active constraint strategy for flexibility analysis in chemical processes. *Computers and Chemical Engineering, 11*(6), 675–693.

Grossmann, I. E., & Straub, D. A. (1991). Recent developments in the evaluation and optimization of flexible chemical processes. In L. Puigjaner & A. Espuna (Eds.), *Proceedings of COPE-91* (pp. 49–59). Barcelona, Spain.

Grossmann, I. E., & Sargent, R. W. H. (1978). Optimum design of chemical plants with uncertain parameters. *AIChE Journal, 24*, 1021.

Grossmann, I. E., & Sargent, R. W. H. (1979). Optimum design of multipurpose chemical plants. *Industrial Engineering in Chemical Process, Design and Development, 18*, 343.

Grossmann, I. E., & Kravanja, Z. (1997). Mixed-integer nonlinear programming: A survey of algorithms and applications. In Biegler, Coleman, Conn, Santosa (Eds.), *The IMA volumes in mathematics and its applications: Large-scale optimization with applications* (Vol. 93). Part II. *Optimal design and control* (pp. 73–100), Berlin: Springer-Verlag.

Grossmann, I. E., Quesada, J., Raman, R., & Voudouris, V. (1996). Mixed-integer optimization techniques for the design and scheduling of batch processes. In G. V. Reklaitis, A. K. Sunol, D. W. T. Rippin, & O. Hortacsu (Eds.), *Batch processing systems engineering* (pp. 451–494). Berlin: Springer-Verlag.

Grossmann, I. E., Caballero, J. A., & Yeomans, H. (1999). Advances in mathematical programming for automated design, integration and operation of chemical processes. *Korean Journal of Chemical Engineering, 16*, 407–426.

Gupta, O. K., & Ravindran, V. (1985). Branch and bound experiments in convex nonlinear integer programming. *Management Science, 31*(12), 1533–1546.

Halemane, K. P., & Grossmann, I. E. (1983). Optimal process design under uncertainty. *AIChE Journal, 29*, 425.

Halemane, K. P., & Grossmann, I. E. (1993). Optimal process design under uncertainty. *AIChE Journal, 29*, 425.

Hasdorff, L. (1976). *Gradient optimization and nonlinear control*. New York, NY: Wiley/Interscience.

Hillier, F., & Lieberman, G. J. (1974). *Introduction to operations research*. San Francisco: Holden-Day.

Holland, J. H. (1975). *Adaptations in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Ierapetritou, M. G. (2001). A new approach for quantifying process feasibility: Convex and one dimensional quasi-convex regions. *AIChE Journal, 47*, 1407.

Johns, W. R., Marketos, G., & Rippin, D. W. T. (1976). The optimal design of chemical plant to meet time-varying demands in the presence of technical and commercial uncertainty. *Design Congress, 76*, F1.

Johnson, E. L., Nemhauser, G. L., & Savelsbergh, M. W. P. (2000). Progress in linear programming based branch-and-bound algorithms: Exposition. *INFORMS Journal of Computing, 12*.

Jung, J. H., Lee, C. H., & Lee, I.-B. (1998). A genetic algorithm for scheduling of multiproduct batch processes. *Computers and Chemical Engineering, 22*, 1725.

Kallrath, J. (2000). Mixed-integer optimization in the chemical process industry: Experience, potential and future. *Transactions of Industrial and Chemical Engineering, 78*(Part A), 809–822.

Kelley Jr, J.E. (1960). The cutting-plane method for solving convex programs. *Journal of SIAM, 8*, 703–712.

Kocis, G. R., & Grossmann, I. E. (1987). Relaxation strategy for the structural optimization of process flowsheets. *Industrial Engineering in Chemical Research, 26*, 1869.

Laarhoven, P. J. M., & van Aarts, E. H. L. (1987). *Simulated annealing: Theory and applications*. Dordrecht: Reidel.

Lakshmanan, A., & Biegler, L. T. (1995). Synthesis of optimal chemical reactor networks. *I&EC Research, 35*(4), 1344.

Leineweber, D. B. Bock, H. G., Schlöder, J. P., Gallitzendörfer, J. V., Schäfer, A., & Jansohn, P. (1997). A boundary value problem approach to the optimization of chemical processes described by DAE models. *Computers & Chemical Engineering* (also: IWR-Preprint 97-14, Universität Heidelberg, March 1997).

Leyffer, S., (1993), Deterministic methods for mixed-integer nonlinear programming. Ph.D. Thesis, Department of Mathematics and Computer Science, University of Dundee, Dundee.

Leyffer, S. (2001). Integrating SQP and branch and bound for mixed-integer noninear programming. *Computational Optimization and Applications, 18*, 295–309.

Loehl, T., Schulz, C., & Engell, S. (1998). Sequencing of batch operations for highly coupled production process: Genetic algorithms vs. mathematical programming. *Computers and Chemical Engineering, 22*, S579.

Logsdon, J. S., & Biegler, L. T. (1989). On the accurate solution of differential–algebraic optimization problems. *I&EC Research, 28*, 1628.

Louveaux, F. V. (1986), *Multistage stochastic programs with block-separable recourse, mathmatical programming study* (Vol. 28, pp. 48–62). Amsterdam: North-Holland.

Luus, R., & Jaakola, T. H. I. (1973). Direct search for complex systems. *AIChE Journal, 19*, 645–646.

Luus, R. (1993). Piecewise linear continuous optimal control by iterative dynamic programming. *Industrial Engineering in Chemical Research, 32*, 859–865.

Madansky, A. (1963). In R. L. Graves, & P. Wolfe (Eds.), *Linear programming under uncertainty, mathematical programming* (pp. 103–110). New York: McGraw-Hill.

Magnanti, T. L., & Wong, R. T. (1981). Accelerated benders decomposition: Algorithm enhancement and model selection criteria. *Operations Research, 29*, 464–484.

Malik, R. K., & Hughes, R. R. (1979). Optimal design of flexible chemical processes. *Computers and Chemical Engineering, 3*, 473.

Mekarapiruk, W., & Luus, R. (1997). Optimal control of inequality state constrained systems. *Industrial Engineering in Chemical Research, 36*, 1686–1694.

Murtagh, B. A., & Saunders, M. A. (1987). MINOS 5.1 user's guide. Technical Report SOL 83-20R, Stanford University.

Nabar, S., & Schrage, L. (1991). Modeling and solving nonlinear integer programming problems. In *Presented at annual AIChE meeting, Chicago*.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal, 7*, 308.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. New York: Wiley/Interscience.

Nilchan, S., & Pantelides, C. (1998). On the optimization of periodic adsorption processes. *Adsorption, 4*, 113.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. New York: Springer.

Ostrovsky, G. M., Volin, M., & Senyavinj, M. M. (1997). An approach to solving two-stage optimization problems under uncertainty. *Computers and Chemical Engineering, 21*, 311–325.

Pinto, J., & Grossmann, I. E. (1998). Assignment and sequencing models for the scheduling of chemical processes. *Annals of Operations Research, 81*, 433–466.

Pistikoploulos, E. N. (2002). *Review of Stochastic Flexibility*, in preparation.

Pistikopoulos, E. N., & Mazzuchi, T. A. (1990). A novel flexibility analysis approach for processes with stochastic parameters with E. Pistikopoulos. *Computers and Chemical Engineering, 14*, 991–1000.

Polak, E. (1997). *Optimization: Algorithms and consistent approximations*. New York: Springer.

Pontryagin, V. V., Boltyanskii, Gamkrelidge, R., & Mishchenko, E. (1962). *The mathematical theory of optimal processes*. New York, NY: Interscience.

Pörn, R., & Westerlund, T. (2000). A cutting plane method for minimizing pseudo-convex functions in the mixed-integer case. *Computers and Chemical Engineering, 24*, 2655–2665.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computing Journal, 7*, 155.

Quesada, I., & Grossmann, I. E. (1992). An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering, 16*, 937–947.

Raghunathan, A., & Biegler, L. T. (2002). *MPEC Formulations and Algorithms in Process Engineering*, submitted for publication.

Rao, C. V., Rawlings, J. B., & Wright, S. (1998). On the application of interior point methods to model predictive control. *Journal of Optimum Theory and Applications, 99*, 723.

Reddien, G. W. (1979). Collocation at Gauss points as a discretization in optimal control. *SIAM Journal on Control and Optimization, 17*, 298.

Rooney, W. R., & Biegler, L. T. (1999). Incorporating joint confidence regions into design under uncertainty. *Computers and Chemical Engineering, 23*(10), 1563–1575.

Sahinidis, N. V. (2003). Optimization under uncertainty: State-of-the-art and opportunities. In I. E. Grossmann, & C. M. McDonald (Eds.), *Proceedings of FOCAPO2003, Coral Springs*.

Sahinidis, N. V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization, 8*(2), 201–205.

Sahinidis, N. V., & Grossmann, I. E. (1991). Convergence properties of generalized benders decomposition. *Computers and Chemical Engineering, 15*, 481.

Sargent, R. W. H., & Sullivan, G. R. (1979). Development of feed changeover policies for refinery distillation units. *Industrial Engineering in Chemical Process, Design and Development, 18*, 113.

Schittkowski, K. (1987). *More test examples for nonlinear programming codes. Lecture notes in economics and mathematical systems* (Vol. 282). Berlin: Springer-Verlag.

Schwartz, A. (1996). Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California-Berkeley.

Schweiger, C. A., & Floudas, C. A. (Eds.), *Process synthesis, design and control: A mixed-integer optimal control framework* (pp. 189–194). *Proceedings of DYCOPS-5 on dynamics and control of process systems*.

Sen, S., Narasimhan, S., & Deb, K. (1998). Sensor network design of linear processes using genetic algorithms. *Computers and Chemical Engineering, 22*, 385.

Straub, D. A., & Grossmann, I. E. (1993). Design optimization of stochastic flexibility. *Computers and Chemical Engineering, 17*, 339.

Stubbs, R., & Mehrotra, S. (1999). A Branch-and-cut method for 0–1 mixed convex programming. *Mathematical Programming, 86*(3), 515–532.

Swaney, R. E., & Grossmann, I. E. (1985a). An index for operational flexibility in chemical process design. Part I. Formulation and theory. *AIChE Journal, 31*, 621.

Swaney, R. E., & Grossmann, I. E. (1985b). An index for operational flexibility in chemical process design. Part II. Computational algorithms. *AIChE Journal, 31*, 631.

Tanartkit, P., & Biegler, L. T. (1995). Stable decomposition for dynamic optimization. *Industrial Engineering in Chemical Research, 34*, 1253–1266.

Torczon, V. (1991). On the convergence of the multidimensional search algorithm. *SIAM Journal of Optics, 1*, 123.

Vanderbei, R. J., & Shanno, D. F. (1997). An interior point algorithm for non-convex nonlinear programming. Technical Report SOR-97-21, CEOR, Princeton University, Princeton, NJ.

Varvarezos, D. K., Grossmann, I. E., & Biegler, L. T. (1992). An outer approximation method for multiperiod design optimization. *Industrial Engineering and Chemical Research, 31*, 1466–1477.

Vassiliadis, V. R., Sargent, W. H., & Pantelides, C. (1994). Solution of a class of multistage dynamic optimization problems. *I&EC Research, 33*, 2123.

Vassiliadis, V. S. (1993). Computational solution of dynamic optimization problems with general differential-algebraic constraints. Ph.D. Thesis, University of London, London, UK.

Vassiliadis, V. S., Sargent, R. W. H., & Pantelides, C. C. (1994). Solution of a class of multistage dynamic optimization problems. 2. Problems with path constraints. *Industrial Engineering in Chemical Research, 33*, 2123–2133.

Viswanathan, J., & Grossmann, I. E. (1990). A combined penalty function and outer-approximation method for MINLP optimization. *Computers and Chemical Engineering, 14*, 769.

Wächter, A., & Biegler, L. T. (2002). Global and local convergence of line search filter methods for nonlinear programming. CAPD Technical Report B-01-09 (August 2001, revised May 2002), submitted for publication.

Wächter, A. (2002), An interior point algorithm for large-scale nonlinear optimization with applications in process engineering. Ph.D. Thesis, Carnegie Mellon University.

Westerlund, T., & Pettersson, F. (1995). A cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering, 19*, S131–S136.

Wets, R. J.-B. (1989). Stochastic programming, In G. L. Nemhauser, et al. (Eds.), *Handbooks in operations research and management science* (Vol. 1, pp. 573–629). Amsterdam: North-Holland.

Williams, H. P. (1985). *Mathematical building in mathematical programming*. Chichester: Wiley.

Wright, S. J. (1996). *Primal-dual interior point methods*. Philadelphia: SIAM.

Yuan, X., Zhang, S., Piboleau, L., & Domenech, S. (1988). Une Methode d'optimisation Nonlineare en Variables Mixtes pour la Conception de Procedes. *RAIRO, 22*, 331.