1 Nonlinear Optimization and Tools

Nonlinear Optimization, also known as nonlinear programming, has proven itself as a useful technique to reduce costs and to support other objectives, especially in the refinery industry. Nonlinear programming knocks to the companies' doors. Mixed integer nonlinear optimization, an area under continual development, is now establishing itself in many branches, *e.g.*, the process industry or financial services, and it certainly has much to offer for the future.

For $\mathbf{x}^T = (x_1, ..., x_{n_c})$ and $\mathbf{y}^T = (y_1, ..., y_{n_d})$, objective function $f(\mathbf{x}, \mathbf{y})$ and constraints $\mathbf{g}(\mathbf{x}, \mathbf{y})$ and $\mathbf{h}(\mathbf{x}, \mathbf{y})$ an optimization problem

$$\min \left\{ f(\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{ll} \mathbf{g}(\mathbf{x}, \mathbf{y}) = 0 & \mathbf{x} \in X \subseteq \mathbb{R}^{n_c} \\ \mathbf{h}(\mathbf{x}, \mathbf{y}) \geq 0 & \mathbf{y} \in U \subseteq \mathbb{Z}^{n_d} \end{array} \right\} \tag{1.1}$$

is called a *mixed integer nonlinear programming problem*, if the domain U is discrete, *e.g.*, $U = \mathbb{N}_0 = \{0, 1, 2, 3, ...\}$ and the functions $f(\mathbf{x}, \mathbf{y})$, $\mathbf{g}(\mathbf{x}, \mathbf{y})$ and $\mathbf{h}(\mathbf{x}, \mathbf{y})$ are nonlinear. The continuous variables in (1.1) could for instance describe the states (temperature, pressure, etc.), flow rates or design parameters of plant or chemical reactors. The discrete variables, often binary variables, may be used to describe the topology of a process network or to represent the existence or non-existence of plants.

In this article we try to give a glance on models, algorithms and software in the are of nonlinear optimization, and also indicate where to find support and consulting firms.

2 Models and Problems

The simplest nonlinear models and problems are quadratic programming (QP) problems. They contain a quadratic objective function $\mathbf{x}^{\mathrm{T}}\mathsf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}$ and linear constraints and are used, for instance, to describe location problems in the sense of the quadratic assignment problem. In some cases, QP problems can be solved as pure MILP problems (Kallrath and Wilson, 1997).If the matrix Q is positive semi-definite, the problem can be solved in polynomial time, and, because the problem is convex, the local optimum coincidences with the global one. If Q is negative semi-definite it is possible to exploit methods used in concave minimization. However, if Q indefinite the problem is NP hard and we have to expect the worst. It can be solved by a Branch&Bound method, which, as in Horst and Thoai (1996) uses, for instance, the radial simplex subdivision as the branching rule, and the solution of certain linear programming problems to derive lower bounds.

The next step is to allow not only for a quadratic objective function but also for quadratic constraints, usually inequalities. These may occur, for instance, if a binary variable $\delta \in \{0, 1\}$ is replaced by the inequality $\delta^2 - \delta \geq 0$, to compute the maximum radius r of n non-overlapping circles contained in

the unit square, in production planning and portfolio problems in which linear stochastic constraints are replaced by quadratic deterministic constraints, or in layout problems in integrated circuit design. Finally, in these class of problems we find an almost classical problem in nonlinear optimization, the *pooling problem* which we encounter as the fuel mixture problem in in refineries and in the petrochemical industry. The pooling problem refers to the intrinsic nonlinear problem of forcing the same (unknown) fractional composition of a multi-component streams emerging from a pool, *e.g.*, a tank or a splitter in a mass flow network. Structurally, this problem is dominated by indefinite bilinear terms of the form $\sum_i \sum_j A_{ij} x_i y_j$ appearing in equality constraints. The pooling problems occur in all multi-component network flow problems in which the conservation of both mass flow and composition is required.

In the chemical process industry reaction kinetics might lead to more general nonlinearities, such as exponential nonlinearities due to Arrhenius terms of the form $e^{-\Delta E/kT}$ describing the reaction kinetics.

Discrete variables enter the models, for instance, to select discrete capacities of pipelines and appropriate links in a network design problem, to interpolate between discrete modes of a steam cracker, or to model semi-continuous flow quantities. In most cases the discrete variables enter on linearly.

A variety of nonlinear mixed integer models in the chemical process industry are provided and discussed by Kallrath (1999): a production planning problem in BASF's petrochemical division, a tanker refinery scheduling problem at a refinery, a site analysis of one of BASF's bigger sites, and a process design problem.

The **first problem** leads to a MINLP model for describing a petrochemical network including several steam crackers and plants located at two different sites. The sites are interconnected by pipelines and ship transport. The sites consist of 2(1) steamcrackers, 3(2) downstream processing plants, recycles and 17(10) storage tanks. The model considers costs for transport, external purchases, raw material, utilities and inventory. Blending leads to nonlinear structures suitable for recursion. The model includes about 10 recursed streams, which lead to about 65 recursed matrix coefficients for each period. Binary variables are needed to select cracker operation modes, to interpolate between them, or representing semi-continuous shipping amounts. The yield coefficients in the steamcrackers, depending nonlinearly on temperature and pressure, are determined through interpolation.

The basic multi-period model for the larger site has six periods, includes distributive recursion with about 400 recursed coefficients and about 3500 variables; 30 of them are binary variables. The basic model is part of the multi-site-model. The problem is solved using `PIMS` (Module PPIMSXX and XPIMS) by AspenTech Corp. (Houston, US). Great effort had to be made in connection with convergence of the distributive recursion. The algorithm tends to cycle in multi-period-computations, shifting around the property errors between the periods instead of eliminating them, especially when using more than six (shorter) periods. Finally, we succeeded in improving the performance, especially regard-

ing the multi-sites model, speedup convergence of distributive recursion and in modelling the interpolation of more than three cracker modes.

In a **second project**, a *Tanker and Refinery Scheduling Problem* was to model and schedule the production and the storage of oil products of the medium size refinery for a period of two to four weeks. The data for supply of crude oil and for the delivery of the products were provided by a preceding production planning optimization. Since not just a single crude oil type is stored in the tanks but different types of different properties are mixed, the mathematical problem becomes nonlinear. The necessity of deciding the day of production, and of which crude oil tank oil is charged to which production unit in order to get a particular product requires the application of binary variables. To solve this MINLP problem two methods had been used:

the software package `GAMS` with the `DICOPT`-algorithm using a nonlinear solver `CONOPT` by ARKI in combination with a MILP-solver.

the software package `XPRESS-MP` by Dash Associates using its new B&B algorithm supporting recursion at each node. The size of the total problem was as following:

technically: 16 crude oils, 4 blending components, 17 crude oil and 8 intermediate tanks, 5 production units and 8 final products,

mathematically: 80000 constraints, 70000 variables,

2000 binary variables, 1300000 nonzeros

The **third problem**, a network design problem with about 6000 variables, leads to a MINLP problem dominated by pooling problems and about 900 binary variables.

The model describes a network of process units within one or more production units connected by a system of pipes. Some of the process units manufacture substances, others produce substances which can be used in other units, others do both. For all units the demand or the amount of products required is fixed. For some of those substances which are already used within the system an expensive re-processing is neccessary in order to get an optimal mixture and quality.

The actual situation is that nearly all raw material comes from external delivery points at high expenses. The idea is to make use of the products manufactured within the model system and to reduce the costs for raw material and for re-processing.

New plumbings may be constructed if the actual pipe system is not sufficient. In addition it might be possible that re-processing of substances or a mixture of substances in a small local re-processing unit is cheaper than getting them from outside. Therefore the construction of new re-processing units has to be decided.

One important aspect concerns the nonlinearities of the model. It becomes nonlinear because of mixing or blending substances where amount of solvent and concentrations are not known in advance *pooling problem* and only the resulting mixture has to fulfil some quality constraints. The decision on the construction of plumbings and re-processing units is modelled by binary variables. The constraints (about 6000) in the model describe the following features:

mass balance equations for substances and solvents

inequalities to fulfil quality constraints

inequalities which can force the construction of plumbings or re-processing units

objective function including all costs (raw material, re-processing, construction)

Solution Approach: The model is structured in several sub-models formulated in `GAMS` of which each is based upon the former, in order to support a homotopy method. Therefore there are purely nonlinear submodels including a very rough linear approximation which only provides initial values. Both are solved by the NLP-solver `CONOPT`. The MINLP problem is presolved by relaxing the binary variables which allows them to have any real value between 0 and 1. The main solution procedure afterwards is based upon the "*Outer Approximation*" (Viswanathan & Grossman, 1990) which is included in the solver program package `DICOPT` (Viswanathan & Grossman, Carnegie Mellon University). Solution times are of the order of one hour on a PC.

The **fourth problem** is concerned with a process design problem in which some process parameters and the optimal topology of a cascade of chemical reactors are computed w.r.t. optimizing total production, selectivity, energy, and costs.

Nonlinearities are related to the exponential terms for the reaction kinetics and mass fractions used to interpolate density and viscosity. Discrete features are required to model minimum flow rates between reactors, the number of reactors and their connections. The variables are the flow rates, and the number and size of reactors. The optimization model has been embedded into an attractive and easy to use user-interfaces. It helps the client in his daily production planning duties to adjust his plant immediately to current needs, *i.e.*, changes in costs, capacity fluctuations or to attributes of orders. The tool supports the process design phase and helps to lay out cascades and connections of a system of reactors. The new designs safe raw material, minimize waste material and increase the capacity of the reactor system. In the layout phase the tool support design and other changed constraints.

3 Algorithms

3.1 ... for NLP Problems

In nonlinear problems we have to distinguish between local optima and the global optimum (see our concluding comments on gloabl optimization). In practice it is observed that, for instance, the pooling problem usually has several local optima. Depending on the initial guesses the solver finds different local optima. Thus, solving models involving pooling problems requires great care and deep understanding of the underlying real-world problems. Let us give at least one bit of general advice: one should certainly try to avoid setting initial guess variables to zero. Once a good solution is found one should keep the values of the recursed variables and use them as initial values in further computations.

Algorithms to solve (1.1) are found, for instance, in Gill *et al.* (1981) or Fletcher (1987). Most are based on linearization techniques. Inequalities are included, for instance, by applying active set methods. The most powerful nonlinear optimization algorithms are the *Generalized Reduced Gradient algorithm* (*GRG*) and *Sequential Quadratic Programming (SQP) methods* and *Interior Point Methods* (IPM) [see, for instance, Bazaraa *et al.* (1993) or Wright (1996)] for problems involving many inequalities. The GRG algorithm was first developed by Abadie and Carpenter (1969) [more recent information is contained in Abadie (1978), Lasdon *et al.* (1978), and Lasdon and Waren (1978) , but see also Gill *et al.* (1981, Section 6.3)]. While it is frequently used to solve nonlinear constrained optimization problems, it is rarely used to solve least-squares problems. A similar remark holds for IPM. A special class of this method includes inequalities by adding logarithmic penalties terms to the objective function. Then the problem can solved as a nonlinear optimization problem with possible equations but no inequalities.

3.1.1 Sequential Quadratic Algorithms

SQP methods belong to the most powerful and frequently used nonlinear optimization algorithms [23] to solve problem (1.1). The basic idea is to solve (1.1) by a sequence of quadratic programming subproblems. The subproblem in iteration k appears as

$$\min_{\Delta\mathbf{x}} \left\{ \tfrac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathsf{H}_k\Delta\mathbf{x} + \nabla f(\mathbf{x}_k)^{\mathrm{T}}\Delta\mathbf{x} \right\}, \quad \Delta\mathbf{x} \in \mathbb{R}^n \tag{3.2}$$

$$\begin{aligned} \mathsf{J}_2(\mathbf{x}_k)^{\mathrm{T}}\Delta\mathbf{x} + \mathbf{F}_2(\mathbf{x}_k) &= 0, \\ \mathsf{J}_3(\mathbf{x}_k)^{\mathrm{T}}\Delta\mathbf{x} + \mathbf{F}_3(\mathbf{x}_k) &\geq 0, \end{aligned}$$

where the subscript k refers to quantities known prior to iteration k, and $\Delta\mathbf{x}$ is the correction vector to be determined. This subproblem [*cf.* Gill *et al.* (1981, Section 6.5.3)] is obtained by linearizing the constraints and terminating the Taylor serious expansion of the objective function of (3.2) after the quadratic

term; the constant term $f(\mathbf{x}_k)$ has been dropped. The necessary condition derived from the Lagrangian function associated with (3.2) is

$$\mathsf{H}_k \Delta\mathbf{x} + \nabla f(\mathbf{x}_k) - \mathsf{J}_2(\mathbf{x}_k)\tilde{\lambda}_{k+1} - \mathsf{J}_3(\mathbf{x}_k)\tilde{\mu}_{k+1} = \mathbf{0}. \tag{3.3}$$

If, furthermore, λ_k denotes the vector of Lagrange multipliers (for convenience we do not distinguish between λ and μ for equations and inequalities) known prior to iteration k, and $\Delta\mathbf{x}_k$, $\tilde{\lambda}_k$, and $\tilde{\mu}_k$ represent the solution of (3.2), then the next iteration follows as

$$\begin{pmatrix} \mathbf{x}_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k \\ \lambda_k \\ \mu_k \end{pmatrix} + \alpha_k \begin{pmatrix} \Delta\mathbf{x}_k \\ \Delta\lambda_k \\ \Delta\mu_k \end{pmatrix}, \quad \begin{pmatrix} \Delta\lambda_k = \tilde{\lambda}_k - \lambda_k \\ \Delta\mu_k = \tilde{\mu}_k - \mu_k \end{pmatrix}, \tag{3.4}$$

where α_k is a damping factor.

For the solution of the quadratic subproblems the reader is referred to Gill *et al.* (1981, Section 5.3.2) or Fletcher (1987, Chapter 10).

3.2 Interior Point Methods

Interior point methods, sometimes, as for example in Gill *et al.* (1981, Section 6.2), also known as barrier methods, have the origins in nonlinear programming. They are special homotopy algorithms for the solution of general nonlinear constrained optimisation problems. But initiated by the work of Karmarkar (1984), a large variety of *interior-point methods* (IPMs) has been developed [see for instance Gonzaga (1992), Lustig *et al.* (1992)], and Mehrotra's so called *primal-dual second-order predictor-corrector methods* [21] for solving linear programming problems as well. In linear programming, IPMs are well suited especially for large, sparse problems or those, which are highly degenerate. Here considerable computing-time gains can be achieved. With respect to the solution strategy most of these algorithms can be classified [see for instance Freund and Mizuno (1996)] as *affine scaling methods*, *potential reduction methods*, *central trajectory methods*

The idea of IPMs is to proceed from an initial interior point $\mathbf{x} \in S$ satisfying $\mathbf{h}(\mathbf{x}) > 0$, towards an optimal solution without touching the boundary of the feasible set S. The condition $\mathbf{h}(\mathbf{x}) > 0$ is (in the second and third method) guaranteed by adding a penalty term to the objective function.

To explain the essential characteristics of central trajectory interior-point methods, let us consider the *logarithmic barrier method* in detail. The primal problem is mapped to a sequence of nonlinear programming problems

$$P^{(k)} : \min \left\{ f(\mathbf{x}) - \mu \cdot \sum_{j=1}^{n} \ln\ h_j(\mathbf{x}) \;\middle|\; \begin{array}{l} \mathbf{g}(\mathbf{x}) = 0 \\ \mathbf{h}(\mathbf{x}) > 0 \end{array} \quad , \quad \mu = \mu^{(k)} \right\} \tag{3.5}$$

with *homotopy parameter* μ where we replaced the non-negativity constraint on the variables with the logarithmic penalty term.

At every iteration step k, μ is newly chosen. The penalty term, and therefore the objective function, increases to infinity if the inequalities becomes active. By suitable reduction of the parameter $\mu > 0$, the weight of the penalty term, which gives the name logarithmic barrier problem to this methods, is successively reduced and the sequence of points obtained by solving the perturbed problems, converges to the optimal solution of the original problem. So, through the choice of $\mu^{(k)}$ a sequence $P^{(k)}$ of minimisation problems is constructed, where the relation

$$\lim_{k\to\infty} \mu^{(k)} \cdot \sum_{j=1}^{n} \ln\ h_j(\mathbf{x}) = 0 \tag{3.6}$$

has to be valid, viz.

$$\lim_{k\to\infty} argmin(P^{(k)}) = argmin(NLP) = x^* \tag{3.7}$$

where the function *argmin* returns an optimal solution vector of the problem.

Applying the Karush-Kuhn-Tucker (KKT) conditions [Karush (1939) and Kuhn and Tucker (1951)], these are the necessary or the sufficient conditions for the existence of local optima in NLP problems, we get a system of nonlinear equations which can be solved with the Newton-Raphson algorithm as shown below. The good news is that the problems $P^{(k)}$ or systems of nonlinear equations they produce need not to be solved exactly in practice, but one is satisfied with the solution achieved after one iteration.

3.3 ... for MINLP Problems

MINLP problems such as are the most difficult optimisation problems of all. They belong to the class $\mathcal{NP}$-complete problems.

MILP problems are combinatorial optimisation problems for which, often, the B&B algorithm based on LP relaxation proves to be sufficiently efficient. A similar statement holds for QP problems. LP and QP problems are special cases of nonlinear programming (NLP) problems. Usually, NLP problems cannot be solved in a finite number of steps but only by iterations. Nevertheless, solving NLP problems is usually easier than solving MILP problems. The reason for that, well-known to numerical analysts, is that many NLP problems can be solved using sequential quadratic programming (a similar technique to sequential linear programming) and have convergence rates of second order. This property may allow us to determine the solution quickly.

Unfortunately, MINLP problems combine all the difficulties of both its subclasses: MILP and NLP. Even worse, in addition they have properties absent in NLP or MILP. While for convex NLP problems a local minimum is identical to the global minimum, we find that this result does not hold for MINLP problems.

At present there exists no algorithm which could solve (1.1) in its general form exactly. Therefore, only special instances of (1.1) are investigated. A significant assumption is convexity. Solution algorithms in discrete optimisation belong to two classes: *deterministic* and *heuristic methods*. All deterministic

methods use implicit enumeration and tree search rules. They try to avoid analyzing sub-trees. Deterministic methods can decide whether a given solution is optimal or not. Heuristic methods lack this feature. Unfortunately, for non-convex problems no exact methods are known.

A simple deterministic method is to list all combinations U of discrete variables $\mathbf{y}_i$. Each binary vector $\mathbf{y}_i$ generates an NLP in the continuous variable vector $\mathbf{x}$. If we solve this NLP problem, it is either infeasible or yields a solution, *i.e.*, a pair $(\mathbf{x}_i, z_i = f(\mathbf{x}_i, \mathbf{y}_i))$. After having solved all NLP problems, we choose the pair with smallest z_i (let us refer to it using the index i^*). Thus, the solution is given by the triple $(\mathbf{x}^* = \mathbf{x}_{i^*}, \mathbf{y} = \mathbf{y}_{i^*}, z_i^* = f(\mathbf{x}_{i^*}, \mathbf{y}_{i^*}))$. This method, of course, works only if U has a limited number of elements and if the NLP subproblems allow us to determine their global minima. Although the convexity assumption is fulfilled for convex (continuous) problems the method is of no practical use because of the prohibitively high numerical effort.

3.4 Deterministic Methods for Solving MINLP Problems

Deterministic methods for solving (convex) MINLP problems fall into three classes:

1. *Branch & Bound* (Gupta and Ravindran, 1985),
2. *Generalised Benders Decomposition* (Geoffrion, 1972) and
3. *Outer Approximation* (Duran and Grossmann, 1986).

The B&B algorithm for MINLP problems by Gupta and Ravindran (1985) is based on the same ideas as the B&B algorithm for solving MILP problems. The first step is to solve the problem generated by relaxing the integrality condition on the variables. If the solution of that problem fulfils all integrality conditions the whole problems is solved. Otherwise, in a minimisation problem the relaxed problem provides a lower bound (of course only if the global minimum can be determined) and the search tree is built up. A feasible integer solution provides an upper bound. A major drawback of the B&B algorithm applied to MINLP problems is that nodes deeper in the tree cannot benefit so greatly from information available at previous nodes as is the case in MILP B&B algorithms using the dual simplex algorithm.

The Generalized Benders Decomposition (GBD) method divides the variables into two sets: complicating and non-complicating variables. In MINLP models the class of complicating variables is made up by the discrete (usually binary) variables. Then the algorithm generates a sequence of NLP sub-problems (produced by fixing the binary variables $\mathbf{y}^k$) and solves the so-called MILP Master problems in the space of the complicating variables. The NLP sub-problems yield upper bounds for the original problem while the MILP Master problems yield additional combination of binary variables $\mathbf{y}^k$ for subsequent NLP sub-problems. Under convexity assumptions the Master problems generate a sequence of lower bounds increasing monotonically. The algorithm terminates if lower and upper bounds equal or cross each other.

Outer Approximation (Duran and Grossmann, 1986) also consists of a sequence of NLP sub-problems (produced by fixing the binary variables $\mathbf{y}^k$) gen-

erated by MILP Master problems. The significant difference is how the Master problems are defined. Algorithms based on Outer Approximation (OA) describe the feasible region as the intersection of an infinite collection of sets with a simpler structure, *e.g.*, polyhedra. In Outer Approximation the Master problems are generated by "outer approximations" (linearisations, or Taylor series expansions) of the nonlinear constraints in *those* points which are the optimal solutions of the NLP subproblems; that is, a finite collection of sets. The key idea of the algorithm by Duran and Grossmann (1986) is to solve the MINLP with a much smaller set of points, *i.e.* tangential planes. In convex MINLP problems, a superset of the feasible region is established. Thus, the OA Master problems (MILP problem in both discrete and continuous variables) produce a sequence of lower bounds monotonically increasing. The termination criterion is the same as above.

While the GBD Master problems have fewer variables and constraints, the OA algorithm provides tighter bounds and needs less iterations for convergence.

Both GBD and OA algorithms have heuristic extensions for non-convex MINLP. In many instances they are even capable of proving optimality.

4 Software

4.1 Software Packages for Refineries

There are many packages available for production planning problems in refineries: `Gamma2000` (Bonner & Moore), `Haverly Systems` (Haverly Systems), `PIMS` (AspenTech). These packages are based on models including the pooling problem. In terms of the solution algorithms they use sequential linear programming, or certain variants of it (recursion, distributive recursion, etc.). Successive Linear Programming may be thought of as a technique for dealing with problems where some of the coefficients (so called recursed coefficients) in an otherwise linear program are functions of the LP variables. The basic idea is to guess the values of the LP variables, calculate the recursed coefficients, and solve the LP again. Repeat this process until the solution converges. In some circumstances convergence to a solution cannot be guaranteed, but where the original guesses are quite good - which is often the case in reality - then the technique can be very fast and reliable. If refinery models include discrete variable, the packages mentioned above usually solve one NLP, fix the fractions, solve one MILP, fix the discrete variables, and solve another NLP, and terminate. The LP and MILP subproblems are most frequently solved by the state-of-the-art LP and MILP solvers `OSL` (IBM Corporation), `CPLEX` (ILOG, F'rance) and `XPRESS-MP` (Dash Associates, England). Beyond it LP and MILP features `XPRESS-MP` has a built in 'recursion' (Successive Linear Programming) feature.

4.2 Languages for NLP Models Formulation

Several languages support the NLP model formulation.

The most popular are AMPL Plus, GAMS and LINGO.

They convert nonlinear expressions and data into an input stream for nonlinear optimizers.

Data Interface AMPL Plus interfaces with Microsoft Excel worksheets. It also interfaces with the following databases: Btrieve, dBASE IV, Microsoft Access/FoxPro, Paradox, and Oracle.

LINGO interfaces with some spreadsheets: Lotus 1-2-3, Microsofts Excel, QuattroPro and Simphony. It also interfaces with the following databases: DB/2, Microsoft Access, Oracle, and Paradox

GAMS interfaces with Lotus 123, Microsoft Excel, and Quattro Pro spreadsheets.

NLP Solvers By applying a specific driver, AMPL interfaces with CONOPT, DONLP2, FSQP, GRG, LSGRG, LANCELOT,

LOQO, MINOS, NPSOL, SNOPT.

LINGO uses inhouse implemented nonlinear programming Generalized Reduced Gradient (GRG) algorithm.

It also incorporates Successive Linear Programming (SLP).

Some NLP solvers have been hooked up to GAMS: CONOPT, DICOPT, and MINOS

4.3 DICOPT & MINOPT

DICOPT (Viswanathan and Grossmann, 1990) is the only commercial software available for solving the MINLP problem (1.1) of realistic size. It uses Outer Approximation with some extensions for non-convex problems. This program is most conveniently used in `GAMS`.

Another software package, for mixed integer nonlinear optimisation is `MINOPT` by Schweiger *et al.* (1996), developed at the Department of Chemical Engineering of Princeton University. This package can even handle MINLP problems with differential constraints or mixed integer optimal control problems.

5 Support and Consulting Firms

Since nonlinear, and especially mixed integer nonlinear optimisation problems are difficult to solve, one might expect that universities are an appropriate address to contact. In some cases that might be worthwhile to try.

Another, natural idea is to contact the software and tool developers such as

AMPL, Bell Labs (http://www.bell-labs.com/)

CPLEX, ILOG Consulting Group (http://www.ilog.fr/corporate/support/)

GAMS Inc., Washington D.C., US (http://www.gams.com),

LINDO Systems Inc. (http://www.lindo.com)

MathPro 2000 (http://sundown-vmp.com/mathpro)

OSL, IBM Business Consulting (http://www.ibm.com/services/buscon/)

XPRESS-MP, Dash Associates, Ltd., England, (http://www.dash.co.uk),

In addition, and alternatively, there exists consulting firms who specialize on projects related to optimization, *e.g.*,

ARKI Consulting & Development A/S, Denmark (http://www.arki.com)

MαBOS Mathematical Business Optimization Services GmbH, Germany (http://www.mabos.com)

Mathesis GmbH, Germany (http://www.mathesis.de)

TechnoLogix Decision Sciences Inc. (http://www.technologix.ca)

These firms have clearly the advantage that they are not biased in terms of tools and software packages.

6 Conclusions

Mixed integer nonlinear optimization has been considered as an approach to solve complex production planning and design problems. The problems discussed are very demanding in terms of the mathematical modeling, and appropriate tuning of the algorithms. In all cases special heuristics had been constructed to provide reasonable initial values to the solver. So, the lesson to be learned is that each MINLP problem is different from others and requires special treatment and techniques. One common features seems to be the problem of getting good initial values to start the solver, which, can be overcome by homotopy techniques.

The problem of the existence of multiple local optima in nonlinear optimization is treated in an mathematical discipline of its own: global optimization, a field including theory, methods and applications of optimization techniques aimed at detecting a global optimum of nonlinear problem. The methods are quite different from the classical concepts of gradients and Hessian, and have more in common what is used in discrete optimization. So far, the problems which can be solved at present are specially structured and are usually small involving only up to, say a hundred of variables or constraints but the field is growing and is worthwile to get into contact with it; it may knock at your door tomorrow anyway. The *Journal of Global Optimization* or the books [Horst and Tuy (1996), Horst *et al.* (1996), or Horst and Pardalos (1995) are good and recommended starting points.

Let us conclude with the observation that nonlinear optimization is a good example that mathematical methods and techniques can support human inventiveness and decisions. Especially, they can ensure that less intuitive solutions are not lost, and can provide a quantitative basis for decisions and allow coping most successfully with complex problems.

Josef Kallrath, Ludwigshafen (Germany) & Gainesville (FL, US)
kallrath@astro.ufl.edu

References

[1] J. Abadie. The GRG Method for Nonlinear Programming. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, pages 335–363. Sijthoff and Noordhoff, The Netherlands, 1978.

[2] J. Abadie and J. Carpenter. Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints. In R. Fletcher, editor, *Optimization*, pages 37–47. Academic Press, New York, 1969.

[3] Mokhtar Bazaraa, Hanif D. Sheraldi, and C. M. Shetly. *Nonlinear Programming*. Wiley, Chichester, UK, 2nd edition, 1993.

[4] Marco A. Duran and Ignacio E. Grossmann. An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programms. *Mathematical Programming*, 36:307–339, 1986.

[5] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, UK, 2nd edition, 1987.

[6] A. M. Geoffrion. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[7] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[8] C. L. Gonzaga. Path-Following Methods for Linear Programming. *SIAM Review*, 34:167–224, 1992.

[9] O. K. Gupta and V. Ravindran. Branch and Bound Experiments in Convex Nonlinear Integer Programming. *Management Science*, 31:1533–1546, 1985.

[10] Reiner Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, Holland, 1995.

[11] Reiner Horst, P. M. Pardalos, and Nguyen Van Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, Holland, 1996.

[12] Reiner Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer, New York, 3rd edition, 1996.

[13] Reiner Horst and Nguyen van Thoai. A New Algorithm for Solving the General Quadratic Programming Problem. *Computational Optimization Applications*, 5:39–48, 1996.

[14] Josef Kallrath. Mixed-Integer Nonlinear Programming Applications. In Ellis L. Johnson Tito A. Ciriani, Stefano Gliozzi and Roberto Tadei, editors, *Operational Research in Industry*, pages 42–76. Macmillan, Houndsmill, Basingstoke, 1999.

[15] Josef Kallrath and John M. Wilson. *Business Optimisation Using Mathematical Programming*. Macmillan, UK, 1997.

[16] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:375–395, 1984.

[17] W. Karush. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. Master thesis, Department of Mathematics, University of Chicago, Chicago, 1939.

[18] L. S. Lasdon and A. D. Waren. Generalized Reduced Gradient Method for Linearly and Nonlinearly Constrained Programming. In H. J. Greenberg, editor, *Design and Implementation of Optimization Software*, pages 363–397. Sijthoff and Noordhoff, The Netherlands, 1978.

[19] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner. Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming. *ACM Trans. Math. Software*, 4:34–50, 1978.

[20] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's Predictor-Corrector Interior-Point Method for Linear Programming. *SIAM Journal of Optimisation*, 2:435–449, 1992.

[21] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[22] C. A. Schweiger, A. Rojnuckarin, and C. A. Floudas. *MINOPT: A Software Package for Mixed-Integer Nonlinear Optimization*. Dept. of Chemical Engineering, Princeton University, Princeton, NJ 08544, 1996.

[23] J. Stoer. Foundations of Recursive Quadratic Programming Methods for Solving Nonlinear Programs. In K. Schittkowski, editor, *Computational Mathematical Programming*, number 15 in NATO ASI Series, Heidelberg, Germany, 1985. Springer.

[24] J. Viswanathan and Ignacio E. Grossmann. A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization. *Comp. Chem. Eng.*, 14(7):769–782, 1990.

[25] S. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.