# Polylithic Modeling and Solution Approaches Using Algebraic Modeling Systems

Josef Kallrath

Department of Astronomy
University of Florida
Gainesville, FL 32611, USA
kallrath@astro.ufl.edu

**Abstract.** Based on the Greek *term monolithos* (stone consisting of one single block) Kallrath (2009) introduced the term *polylithic* for modeling and solution approaches in which mixed integer or non-convex nonlinear optimization problems are solved by tailor-made methods involving several models and/or algorithmic components, in which the solution of one model is input to another one. This can be exploited to initialize certain variables, or to provide bounds on them (problem-specific preprocessing). Mathematical examples of polylithic approaches are decomposition techniques, or hybrid methods in which constructive heuristics and local search improvement methods are coupled with exact MIP algorithms

Tailor-made polylithic solution approaches with thousands or millions of solve statements are challenges on algebraic modeling languages. Local objects and procedural structures are almost necessary. Warm-start and hot-start techniques can be essential. The effort of developing complex tailor-made polylithic solutions is awarded by enabling us to solve real-world problems far beyond the limits of monolithic approaches and general purpose solvers.

**Keywords:** algebraic modeling languages, branch&price, column generation, decomposition, hybrid methods, monolithic, primal methods, polylithic, problem-specific preprocessing

## 1 Introduction

Instead of solving difficult optimization problems directly as monolithic problems, they can be solved equivalently or approximately by solving sequences of models leading to polylithic modeling and solution approaches among them: column generation, branch-and-price, exploiting auxiliary problems to generate safe bounds for the original problem which then makes the original problems more tractable, or hybrid methods, *i.e.*, constructive heuristics and local search on subsets of difficult discrete variables leaving the remaining variables and constraints in tractable mixed integer problems. We provide an overview on such approaches and discuss requirements they put on algebraic modeling languages.

We connect the methods briefly covered in this paper with illustrative literature examples from the paper and metals industries, scheduling in the process industry, and planning of hydro-thermal plants in the energy industry.

### 1.1 Optimization Problems

The optimization problem class we have in mind are mixed integer programming (MIP) problems specified by the augmented vector $\mathbf{x}_{\oplus}^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}} \oplus \mathbf{y}^{\mathrm{T}}$ established by vectors $\mathbf{x}^{\mathrm{T}} = (x_1, ..., x_{n_c})$ and $\mathbf{y}^{\mathrm{T}} = (y_1, ..., y_{n_d})$ of $n_c$ continuous and $n_d$ discrete variables, an objective function $f(\mathbf{x}, \mathbf{y})$, $n_e$ equality constraints $\mathbf{h}(\mathbf{x}, \mathbf{y})$ and $n_i$ inequality constraints $\mathbf{g}(\mathbf{x}, \mathbf{y})$. The problem

$$\min \left\{ f(\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{l} \mathbf{h}(\mathbf{x}, \mathbf{y}) = 0 \;\; \mathbf{h} : X \times U \to \mathbb{R}^{n_e} \;\; \mathbf{x} \in X \subseteq \mathbb{R}^{n_c} \\ \mathbf{g}(\mathbf{x}, \mathbf{y}) \geq 0, \mathbf{g} : X \times U \to \mathbb{R}^{n_i}, \mathbf{y} \in U \subseteq \mathbb{Z}^{n_d} \end{array} \right\} \tag{1}$$

is called *Mixed Integer Nonlinear Programming* (MINLP) problem, if at least one of the functions $f$, $\mathbf{g}$ or $\mathbf{h}$ is nonlinear. The vector inequality, $\mathbf{g}(\mathbf{x}, \mathbf{y}) \geq 0$, is to be read component-wise. Any vector $\mathbf{x}_{\oplus}^{\mathrm{T}}$ satisfying the constraints of (1) is called a *feasible point* of (1). Any feasible point, whose objective function value is less or equal than that of all other feasible points is called an *optimal solution*. Depending on the functions $f$, $\mathbf{g}$, and $\mathbf{h}$ in (1) we get the problems types

| *acronym* | *type of optimization* | $f(\mathbf{x}, \mathbf{y})$ | $\mathbf{h}(\mathbf{x}, \mathbf{y})$ | $\mathbf{g}(\mathbf{x}, \mathbf{y})$ | $n_d$ |
|---|---|---|---|---|---|
| LP | Linear Programming | $\mathbf{c}^{\mathrm{T}}\mathbf{x}$ | $\mathsf{A}\mathbf{x} - \mathbf{b}$ | $\mathbf{x}$ | 0 |
| QP | Quadratic Programming | $\mathbf{x}^{\mathrm{T}}Q\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}$ | $\mathsf{A}\mathbf{x} - \mathbf{b}$ | $\mathbf{x}$ | 0 |
| NLP | Nonlinear Programming | | | | 0 |
| MILP | Mixed Integer LP | $\mathbf{c}^{\mathrm{T}}\mathbf{x}_{\oplus}$ | $\mathsf{A}\mathbf{x}_{\oplus} - \mathbf{b}$ | $\mathbf{x}_{\oplus}$ | $\geq 1$ |
| MIQP | Mixed Integer QP | $\mathbf{x}_{\oplus}^{\mathrm{T}}\mathsf{Q}\mathbf{x}_{\oplus} + \mathbf{c}^{\mathrm{T}}\mathbf{x}_{\oplus}$ | $\mathsf{A}\mathbf{x}_{\oplus} - \mathbf{b}$ | $\mathbf{x}_{\oplus}$ | $\geq 1$ |
| MINLP | Mixed Integer NLP | | | | $\geq 1$ |

with a matrix $\mathsf{A} \in \mathcal{M}(m \times n, \mathbb{R})$ of $m$ rows and $n$ columns, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, and $n = n_c + n_d$. MIPs without continuous variables are integer programs (IPs). Sometimes, the problems can be equivalently formulated in several classes, e.g., Rebennack et al. (2009) derive an exact algorithm from a continuous, bilinear formulation of the fixed charge network flow problem. The authors reformulate this classical MILP problem with a continuous QP programming formulation.

### 1.2 Difficult Optimization Problem

While LP problems can be solved relatively easily (the number of iterations, and thus the effort to solve LP problems with $m$ constraints grows approximately linearly in $m$), the computational complexity of MILP and MINLP grows exponentially with $n_d$ but depends strongly on the structure of the problem. Numerical methods to solve NLP problems work iteratively and the computational problems are related to questions of convergence, multiple local optima and availability of good initial solutions. Global optimization techniques, cf. Horst and Pardalos (1995), Floudas (2000), or Floudas and Gounaris (2009), can be applied to both NLP and MINLP problems and its complexity increases exponentially in the number of all variables entering nonlinearly into the model.

From a practical point of view, we call optimization problems difficult when they cannot be solved to optimality, or within a reasonable integrality gap or any guaranteed bound by any standard solver within a reasonable time limit. Problem structure, size, or both can produce difficult problems. From a theoretical point of view it is relevant to observe that solving MILP problems and the problem of finding appropriate bounds is often $\mathcal{NP}$-hard, which makes these problems hard to solve. A consequence of this structural property is that these problems scale badly. If the problem can be solved to optimality for a given instance, this might not be so if the size is increased slightly.

## 2 General Concepts

Based on the Greek *term monolithos* (stone consisting of one single block) Kallrath (2009) introduced the corresponding term *polylithic* for modeling and solution approaches in which mixed integer or nonconvex nonlinear optimization problems are solved by tailor-made methods involving several models and/or solve statements or algorithmic components.

### 2.1 Monolithic Models and Solution Approaches

A *monolithic model* is just *one* model with data, a set of variables and a set of constraints and *one* solve statement calling a general purpose LP, MILP, NLP, or MINLP solver (gpS) – the standard for solving many problems. It keeps the structure of a model and its solution process relatively easy and clean. Bounds on variables are set only once, and the output of a model and solve statement is taken and converted into a reasonable looking report.

### 2.2 Polylithic Models and Solution Approaches (PMSAs)

In contrast, a *polylithic model* contains a set of models which are connected in their data flow of input and output data, *i.e.*, the solution of one model is input to another one. This can be exploited to initialize certain variables, or to provide bounds on them. Examples of polylithic solution approaches are decomposition approaches, column generation as in Gilmore and Gomory (1961)] and Branch&Price [see, for instance, Barnhart et al. (1998)] or hybrid techniques [see, for instance, Pochet and Wolsey (2006)] in which constructive heuristics and local search improvement methods are coupled with exact MIP algorithms to produce feasible points and tight lower and upper bounds. Thus, we observe that the sub-models of polylithic models are often connected in such a way that they represent a tailor-made algorithm. This may create several problems:

1. Bounds and initial values of variables need careful tracing and updating.
2. Output variables, which are declared as binary or integer variables, are usually returned with small deviations from being integral requiring rounding techniques to avoid numerical problems in subsequent problems to be solved.

3. Variables and constraints in different models must either be declared *local*, or otherwise carefully distinguished from each other in the naming conventions.
4. Maintenance efforts are usually much higher than in monolithic models and adding new model features may be not straightforward.

PMSAs have a broad range of applications in which they extend the scope of real-world problems which we can solve. We find them useful in problem-specific preprocessing, in generic mathematical algorithms, and in structural primal heuristics and hybrid methods. Tailor-made polylithic solution approaches with thousands or millions of solve statement to be executed put an extreme challenge on algebraic modeling languages. Hot-start techniques avoiding that the whole problem is retranslated and regenerated become essential.

PMSAs are recommended if we cannot solve our problem directly – they are a last resort. They can also seen as some kind of bridging technology which might be replaced once the solver technology makes another significant step forward.

## 3 Problem-Specific Preprocessing

*Preprocessing* methods [presolve (arithmetic tests on constraints and variables, bound tightening), disaggregation of constraints, coefficient reduction, and clique and cover detection.] introduce changes to LPs, MILPs, NLP, or MINLPs to speed up the algorithms leaving the problem's feasible region unchanged. Johnson et al. (1985) gives a selection of preprocessing algorithms, another overview of simple and advanced preprocessing techniques is by Savelsbergh (1994).

Commercial vendors exploit these methods in their solvers and do their own presolving but vendors are usually not very specific about this topic. Besides the lack of transparency, the modeler might know more about his problem. Thus, it is sometimes very useful to exploit problem-specific structure and properties.

### 3.1 Dynamic Computation and Reduction of Big-M Coefficients

The relevance of minimal Big-M coefficients in activating or deactivation inequalities such as $x \leq X + M(1 - \delta)$ with a continuous variable $x$ and a binary variable $\delta$, the iterative reduction of $M$ in a sequence of models, and an illustrative production planning example fully implemented in the modeling language `GAMS` are discussed by Kallrath (2009). The number of nodes in Branch&Bound algorithms can be reduced significantly. In general, this method is effective when the pre-solving techniques of the commercial solver are not able to tighten the formulation, but only a sequence of tailor-made auxiliary problems can do so.

### 3.2 Computing Tight Bounds on Integer Variables

Another PMSA to tighten models is to derive tight upper bounds on integer variables. The range reduction applied to the number of batches in Lin et al. (2005, Table 4), in which a reactor portfolio problem is solved, is an example

of this technique. The objective function in the auxiliary problems to be solved was to maximize the number of batches. Especially in the context of integer variables, this technique is very much recommended, as it significantly improves the lower bound of the overall problem. This can be understood similarly to having an integer variable $\alpha$ bounded by $\alpha \leq 3$ instead of $\alpha \leq 3.4$.

### 3.3 Feasibility Checks on Data

In complex applications, checking data consistency is not easy, as infeasibilities show only up when certain constraints need to be fulfilled simultaneously. To avoid manually diagnosing infeasibility problems caused by data inconsistencies, checks should be automized. In process industry scheduling models, for instance, these tests can range from simple checks for inconsistent batch sizes, over solving auxiliary models to check state-task network and topology consistency, compatibility of production facilities, storage and demand, to more complicated procedures as in single-order analysis which solve the scheduling problem exactly for each order individually and provides lower bounds on underproduction.

Especially, unavoidable underproduction and delays should be rigorously computed using the goal programming techniques in Section 4.4 instead of resorting to penalty techniques to reduce underproduction and delays. As discussed by Kallrath and Maindl (2006, pp. 344), penalty terms containing weighting coefficients without any economic interpretation may lead to numerical problems, and the interpretation of the integrality gap becomes difficult.

## 4 Mathematical Algorithms

Here we cover generic, not problem-specific mathematical algorithms such as Branch&Bound, Branch&Cut, decomposition approaches, dynamic programming (DP), or goal programming (GP) for multi-criteria problems.

### 4.1 Branch&Bound and Branch&Cut

Most commercial solvers use Branch&Bound (B&B) to solve MIP or nonconvex NLP or MINLP problems. Branching usually operates on variables. For semi-continuous variables (SCVs) or special-ordered sets there exist special branching strategies. As not all solver exploit them, Kalvelagen (2003) implemented the whole branching method for semi-continuous variables in `GAMS` to solve a NLP problem which contained additional SCVs. Problem-specific branching on other structures, e.g., constraints, is, most likely, not contained in commercial solvers – another promising case for PMSAs.

Inequalities are added dynamically to models in, for instance, Branch&Cut (B&C) methods, outer approximation, or in GP discussed in Section 4.4; cf. Karuppiah and Grossmann (2008) for global optimization of nonconvex MINLPs with decomposable structures or Rebennack et al. (2011) for an in-depth tutorial on B&C on the example of the stable set problem, as well as Rebennack et al.

(2011) for advances in graph shrinking procedures within B&C for the stable set problem. Commercial MILP solvers support B&C. However, sometimes, special problem-specific cuts can be added statically, or dynamically. The dynamic case is more interesting and is illustrated by various examples in the `GAMS` model library, in which the tour-eliminating constraints are dynamically added to the core model of the traveling salesman problem (TSP). Although, this technique works only for small TSP problems, it can be useful to solve other problems.

### 4.2 Decomposition Approaches

Decomposition techniques decompose the problem into smaller problems which can be solved in sequence or in any combination. There are standardized techniques such as Dantzig-Wolfe Decomposition (a specific column generation techniques) for solving LP problems with special structure, or Benders' Decomposition [cf. Benders (1962) or Floudas (1995, Chap. 6)], but also structure-exploiting tailor-made decompositions.

**Benders' Decomposition (BD)** This decomposition was originally developed by Benders (1962) to decompose a MILP problem into a series of master problem (MP) and subproblems. The MPs are IP problems with one continuous variable and the subproblems are all LPs. The MP passes on integer solutions to the subproblem which then evaluate its optimality to the original MILP or which generates a feasibility or optimality cut passed back to the master problem. BD has also been applied to large LPs – particularly popular in the stochastic optimization community where BD is also called L-shaped method (Birge and Louveaux 2000) – and NLP problems, the so-called generalized BD. By applying the BD on the subproblems again, one obtains the so-called *Nested BD*.

**Column Enumeration (CE) and Column Generation (CG)** The term *column* usually refers to variables in LP parlance. In the context of CE and CG, it has wider meaning and stands for any kind of objects involved. In cutting stock problems a column might represent a cutting pattern, in network flow problems a feasible path through the network, and in vehicle routing problems a subset of orders assigned to a vehicle or a feasible tour derived by solving a shortest path problem (Desrochers et al. 1992).

The basic idea of CG is to decompose a problem into a master and a subproblem. Often, the decomposition has a natural interpretation. Problems which otherwise could be nonlinear can be completely solved by solving only linear problems. The critical issue is to generate master and subproblems which both can be solved in short time. The most famous example is by Gilmore and Gomory (1961, 1963; hereafter GG) for computing the minimal number of rolls to satisfy the requested demand for smaller sized rolls. The monolithic version of this cutting stock problem is a MINLP problem with a large number of integer variables.

In simple cases, it is possible to generate all columns explicitly, even within modeling languages. If not all columns can be generated, the columns are added dynamically to the problem; cf. Barnhart et al. (1998) or Lübbecke and Desrosiers (2005) for good reviews of CG.

CE can be seen as a special variant of CG applicable when a small number of columns is sufficient. This is, for instance, the case in real-world cutting stock problems when it is known that the optimal solution have only a small amount of trimloss. This, usually, eliminates most of the patterns. CE naturally leads to selecting columns by solving partitioning models; cf. Schrage (2006, Sect. 11.7) with several illustrative example problems of grouping, matching, covering, partitioning, and packing. A general framework for grouping a set of given objects into subsets has been developed by Rebennack et al. (2009) and applied to a metals industry cutting problem. Despite the limitations with respect to the number of columns, CE has some significant advantages: no pricing problem, easily applicable to MIP problems, and low implementation effort.

**Column Generation and Branch&Price** More generally, CG is used to solve well structured MILP problems involving a huge number, say several hundred thousand or millions, of variables, i.e., columns. Such problems lead to large LP problems, if the integrality constraints of the integer variables are relaxed. If the LP problem contains so many variables (columns) that it cannot be solved with a direct LP solver (revised simplex, interior point method) one starts solving this so-called *master problem* with a small subset of variables yielding the *restricted master problem*. After the restricted master problem has been solved, a pricing problem is solved to identify new variables. This step corresponds to the identification of a non-basic variable to be taken into the basis of the simplex algorithm and coined the term *column generation*. The restricted master problem is solved with the new number of variables. The method terminates when the pricing problems cannot identify any new variables. The most simple version of CG is found in the Dantzig-Wolfe decomposition (Dantzig and Wolfe 1960).

GG were the first who generalized the idea of dynamic CG to an IP problem: the cutting stock problem with many applications in the paper industry. In this case, the pricing-problem, i.e., the subproblem, is an IP problem itself - and one refers to this as a *column generation algorithm*. This problem is special as the relaxed master problem generates columns sufficient to get the optimal integer feasible solution of the overall problem. In general this is not so. If both, the subproblem and the master problem, involve integer variables, the CG part is embedded into a B&B method: this is called *Branch&Price* (B&P); cf. Barnhart et al. (1998) and Savelsbergh (2001). Thus, B&P is IP with CG. Note that during the branching process new columns are generated; therefore the name *branch-and-price*. B&P (often coupled with Branch&Cut) are tailor-made implementations exploiting a decomposition structure. While the implementation effort of B&P is considerable, it belongs to the most efficient techniques for solving MIP problems with CG. A list of successful applications in various fields has been given by Kallrath (2008).

### 4.3 Dynamic Programming

Dynamic programming (DP) computes the optimal solution as a sequence of optimization problems over time by defining value functions worked out backwards exploiting a recursive relationship, the Bellman equation. In the energy industry, *stochastic DP* (SDP) is popular and extends the Bellman recursion for DP problems to the stochastic case by taking the expectation. In order to approximate the expected cost or benefit functions, SDP relies on interpolation techniques. An interesting application is the water reservoir problem by Howitt et al. (2002) determining from year to year, the optimal water releases and the optimal carryovers. The North Platte Storage Project (NPSP) can be modeled as a single aggregated reservoir and a single release to the North Platte irrigators. The objective function is a net benefit function on the water releases.

### 4.4 Multi-Criteria Optimization

Pre-emptive goal programming (GP) is a recommended approach to multi-criteria optimization if there is a ranking between incommensurate objectives or goals ordered according to priorities; cf. Ignizio (1976), Romero (1991), or Schniederjans (1995). A multi-criteria MILP is converted to a sequence of MILP problems. The goal at priority level $i$ is considered to be infinitely more important than the goal at the next level, $i+1$, but they can be relaxed by an absolute or relative amount when optimizing for level $i+1$. In a scheduling problem we might have the ranking: minimize underproduction ($i = 1$), minimize delays ($i = 2$), and finally maximize contribution margin ($i = 3$). The list of goals is worked down according to this priority list and by converting higher-level objectives to additional inequalities when optimizing for goal at level $i+1$.

## 5 Primal Heuristics

Primal heuristics allow us to compute a good primal feasible point to be used as an approximation to the optimal solution, or as an initial point in improvement techniques. We distinguish structured primal heuristics and hybrid techniques. Structured primal heuristics follow certain computational steps and can be applied widely. Hybrid techniques are solution approaches which combine exact MIP algorithms and constructive heuristics or metaheuristics such as the improvement heuristics *simulated annealing*, *genetic algorithms*, *tabu search* and so on. Both approaches are usually problem-specific and exploit the problem structure for an appropriate decomposition and concept of environment. In their kernel they use a declarative model to be solved, for instance, by a MILP solver.

In its simplest form, local search can be used to improve the solution obtained by a constructive heuristic. The main idea is to solve repeatedly the subproblem on small number of binary variables reoptimizing, for instance, the production of some products. The binary variables for resolving could be chosen randomly, or by a metaheuristic such as tabu search. All binary variables related to them

are released the other ones are fixed to the previous best values. More general, the neighborhood of a solution may be defined in many other ways, e.g., for a binary vector, any other vector may be called a neighbor, if it is within a Hamming distance 3. In combination with MIP technique, such neighborhood leads to another polylithic method called *local branching* (Fischetti and Lodi 2003). In this context we also mention the RINS heuristics embedded in CPLEX or Xpress, as well as the *feasibility pump* proposed by Fischetti and Glover (2005), which can also be seen as a polylithic method.

If possible, in addition to the primal feasible point one should derive reasonable bounds. While the primal heuristic provides an upper bound for minimization problems, in favorite cases, the MIP part of the hybrid solver provides lower bounds. In other cases, lower bounds can be derived from auxiliary problems which are relaxations of the original problem, and which are easier to solve.

### 5.1 Structured Primal Heuristics

Structured primal heuristics enable us to compute primal feasible points, which give us in maximization (minimization) problems lower and upper bounds, but no dual information (upper or lower bounds, resp.), *i.e.*, a gap specifying the quality of the solution. In some cases they can be extended or combined with other methods enabling us to compute the gap as well.

**LP-Guided Dives, Relax-and-Fix** Primal feasible solutions can be computed by systematically fixing critical discrete variables. In this class of approaches we find, for instance, *dive-and-fix* (DF) and *relax-and-fix* (RF), also called *fix-and-relax*, or *sliding window technique*, e.g., by Van Dinter et al. (2011). In DF the LP relaxation of a MIP problem is solved followed by fixing a subset of fractional variables to suitable bounds. *Near-integer-fix* (NIF) is a variant of DF which fixes variables with fractional values to the nearest integer point.

While NIF and DF can easily lead to infeasibility, this is less an issue with RF, a marching scheme exploiting partitioning structures in products, jobs, time or geometry; cf. Kallrath (2012) using RF to solve small instances of the Eternity II puzzle problem (cf. Benoist and Burreau 2008 or Muñoz et al. 2009), a highly combinatorial, NP-complete edge-matching problem with very little structure but $256! \cdot 4^{256}$ possible combinations. RF requires, that we can partition the discrete variables into $R$ disjoint or weakly overlapping subsets $\mathcal{S}_r$, $r \in \mathcal{R} := \{1, \ldots, R\}$, of decreasing importance; in production planning, for instance, earlier time periods could be more important than later periods. Based on these partitions $\mathcal{S}_r$, $R$ MIP problems $\mathcal{P}_r$ are solved leading to a composed feasible point to the original problem $\mathcal{P}$. In problem, $\mathcal{P}_r$, $2 \leq r \leq R$, the values of $\delta \in \mathcal{S}_{r-1}$ are fixed at their optimal values from $\mathcal{P}_{r-1}$, and integrality is only required for $\delta \in \mathcal{S}_r$. As $\mathcal{P}_1$ is a relaxation of $\mathcal{P}$, for a minimization problem, the objective function value of $\mathcal{P}_1$ provides a lower bound of $\mathcal{P}$. Either $\mathcal{P}_r$ is infeasible for some $r \in \{1, \ldots, R\}$, or else $\delta_R$ and the associated continuous variables, $x_R$ establish a RF solution. Usually, it suffices to ensure feasibility to allow for

some overlap of $\mathcal{S}_{r-1}$ and $\mathcal{S}_r$. Additional free binary variables in partition $r-1$ connect adjacent partitions $r-1$ and $r$.

RF works well for pump-storage energy systems. Hourly discretization over a year and setup costs for pumps and thermal units lead to tens of thousands of binary variables. As later and earlier time periods are only weakly coupled, a few partitions comprising about 90 days, suffice to produce good feasible points solved to optimality in 2 minutes exploiting the `CPLEX` mipstart parameter.

**SOS-2 Based Linear Approximations to Nonlinear Problems** Beale and Forrest (1976) presented the idea of linear approximations to compute the global minimum of nonconvex nonlinear functions. They introduced special ordered sets of type 2 (SOS-2) and efficient branching schemes to exploit this structure. Since then, various contributions elaborated on the usage of SOS-2, among them Farias et al. (2000, 2008), optimizing a discontinuous separable piecewise linear function, Vielma et al. (2009) developing a unifying framework and extensions to mixed-integer models for nonseparable piecewise-linear optimization problems, or Misener and Floudas (2010) constructing explicit, piecewise-linear formulations of two- or three dimensional functions.

**Homotopy Sequences on Models** Kallrath (1999) solved two difficult MINLP problems, a water network flow problem (P1) and a reactor topology problem (P2). As both models were dominated by pooling problems, initial guesses for the fractional composition of the multi-component streams could be derived from a simplified LP for simplified NLP sub-models. A homotopy method is used in P1 by solving a sequence of sub-models formulated in `GAMS`, of increasing complexity (LP, NLP-1, NLP-2, MINLP) exploiting the results of the previous ones. In P2, scaling was very important. Solving an auxiliary problem with an artificial objective function minimizing the violation of critical nonlinear constraints provided excellent initial guesses.

Homotopy methods construct feasible points from a sequence of relaxed models, e.g., in a scheduling problem $\mathcal{P}$ with due times one might relax these due times obtaining the relaxed model $\mathcal{R}$. The optimal solution, or even any feasible point of $\mathcal{R}$ is a feasible point of $\mathcal{P}$ if the due times are modeled with appropriate unbounded slack variables.

### 5.2 Hybrid Techniques

Hybrid techniques combine exact MIP algorithms and constructive heuristics, or improvement methods (local search, or metaheuristics such as simulated annealing, genetic algorithms, tabu search and so on).

**Constructive Heuristics involving Exact Methods** In constructive heuristics we exploit the structure of the problem and compute a feasible point. Once we have a feasible point we can derive safe bounds on the optimum and assign

initial values to the critical discrete variable which could be exploited by the GAMS/CPLEX *mipstart* option. Feasible points can be generated by the structured primal heuristics and sequences of relaxed models discussed in Section 5.1, or by *unstructured heuristics* tailor-adjusted to the problem structure and difficult to transfer to other problems; cf. Kallrath (2008) for minimizing the number of patterns in a cutting stock problem. Developing unstructured heuristics is the art of computing with good feasible point in short times – everything is allowed.

**Metaheuristics involving Exact Methods** In another class of hybrid method called *Matheuristics* (cf. Maniezzo et al. 2010) LP, MILP, NLP, or MINLP solvers are combined with another algorithmic method, e.g., the combination of MIP and constraint logic programming (CLP) strategies developed by Harjunkoski et al. (2000) or Jain and Grossmann (2001) for solving scheduling and combinatorial optimization problems. Timpe (2002) solved a planning and scheduling problem with mixed MILP and constraint programming (CP). Maravelias & Grossmann (2004) proposed a hybrid/decomposition algorithm for the short term scheduling of batch plants, and Roe et al. (2005) presented a hybrid MILP/CLP algorithm for multipurpose batch process scheduling, in which MILP is used to solve an aggregated planning problem while CP is used to solve a sequencing problem. Other hybrid algorithms combine evolutionary and mathematical programming methods, cf. the heuristics by Till et al. (2005) for stochastic scheduling problems and by Borisovsky et al. (2006) for supply management problems.

### 5.3 Multi-Start Techniques

Monolithic models may exploit solver-specific parameters (TPs). PMSAs may contain additional model or algorithmic TPs. While it may not be obvious how to set such TPs, the results may strongly depend on their values. In such situation it pays out to start the whole application on several cores of multi-core CPUs. For solving scheduling problems in the process industry, we have implemented a multi-start techniques coded in GAMS with the master program *multi-start.gms* copying the whole application and all input data to separate subdirectories, generating the list of TPs, starting and communicating with the applications on the individuals cores, evaluating returned results, shutting-down processes and thus working down a list of parameter settings until a good feasible solution has been found. The current coding with the numbers of cores, active parallel runs and parameter sets to be evaluated as input control data works well on a 12 core machine; however, asynchronous GAMS calls could help to simplify it.

## 6 Outlook - Challenges for Algebraic Modeling Systems

Algebraic modeling systems (AMSs) benefit a) from the declarative approach which allows the user to implement monolithic models close to a formal model documentation in scientific publication. The second advantage, b) is that stable

interfaces to a broad class of solvers noticeably reduced the required knowledge about the numerical optimization algorithms.

PMSAs put more demands on AMSs in terms of coding, computational speed and modularization. Increased computing power has reduced the speed advantage higher programming languages (HPLs) such as C, or C++, or Fortran had over AML scripting code which is somewhat slower, but for many applications fast enough. The possible speed gain from HPLs might in many cases not offset the higher developments costs. A similar argument holds for the numerical optimization part: even highly non-linear, badly scaled and large models can often be solved in acceptable time by using default options of solvers. As such, the combination of AMLs, advances in the algorithms underlying the solvers and the increase of computing power have dramatically changed of what is possible. Even PMSAs with nonconvex and nonlinear subproblems to be solved to global optimality, are now possible; cf. Rebennack et al. (2009).

Tailor-made PMSAs with thousands or millions of solve statements to be executed put an extreme challenge on algebraic modeling languages. Hot-start techniques become essential and available; cf. Lohmann (2011) for planning hydro-thermal plants in the energy industry. PMSA code with ten thousands or hundred thousands of code lines comprises much more than the model declaration and the solve statement – the core modeling languages were developed for. The advantages of simple language features might turn into a disadvantage: With all symbols being global, name-space conflicts are very likely. Thus, it becomes very important for PMSAs that modeling languages support the concept of subroutines or functions with encapsulated, only locally visible objects.

What is the benefit of all these efforts? The set of solvable real-world problems is extended both in quality (structure) and size (number of variables and constraints). There is on the one hand, indeed more effort to setup and maintain PMSAs, but on the other hand, algebraic modeling systems become more and more suitable to support such approaches. We should not be surprised to see PMSAs used for really difficult problems, and those for which it pays out to take the effort to develop tailor-made solutions. The better and easier AMLs support PMSAs, the more these techniques will be used and the more will the edge of solvable real-world problems be moved to larger and more complex problems.

**Acknowledgement**: We thank Dr. Jan Jagla (GAMS GmbH, Braunschweig) for helpful suggestions, and the unknown referees for their constructive reports.

## References

1. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsberg, and P. H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.
2. E. Beale and J. Forrest. Global Optimization Using Special Ordered Sets. *Mathematical Programming*, 10:52–69, 1976.
3. J. F. Benders. Partitioning Procedures for Solving Mixed-variables Programming Problems. *Numerische Mathematik*, 4:238–252, 1962.

4. T. Benoist and E. Bourreau. Fast Gloabl Filtering for Eternity II. *Constraint Programming Letters*, 3:35–50, 2008.
5. J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Operations Research and Financial Engineering. Springer, New York, 2000.
6. P. Borisovsky, A. Dolgui, and A. Eremeev. Genetic Algorithms for Supply Management Problem with Lower-bounded Demands. In A. Dolgui, G. Morel, and C. Pereira, editors, *Information Control Problems in Manufacturing 2006: A Proceedings volume from the 12th IFAC International Symposium. Vol. 3., St Etienne, France, 17-19 May 2006*, pages 521–526, Dordrecht, North-Holland, 2006. Elsevier.
7. E. Danna, E. Rothberg, and C. Le Pape. Exploring Relation Induced Neighborhoods to improve MIP Solutions. *Mathematical Programming*, 102:71–90, 2005.
8. B. Dantzig and P. Wolfe. The decomposition algorithm for linear programming. *Operations Research*, 8:101–111, 1960.
9. I. R. de Farias Jr., E. L. Johnson, and G. L. Nemhauser. A Generalized Assignment Problem with Special Ordered Sets: a Polyhedral Approach. *Mathematical Programming Ser. A*, 89:187–203, 2000.
10. I. R. de Farias Jr., M. Zhao, and H. Zhao. A Special Ordered Set Approach for Optimizing a Discontinuous Separable Piecewise Linear Function. *Operations Research Letters*, 36:234–238, 2008.
11. M. Desrochers, J. Desrosiers, and M. M. Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with time Windows. *Operations Research*, 40(2):342–354, 1992, March-April.
12. J. V. Dinter, S. Rebennack, J. Kallrath, P. Denholm, and A. Newman. The Unit Commitment Model: A Tight Formulation for Benders' Decomposition with a Case Study. *Annals of Operations Research*, submitted, 2011.
13. M. Fischetti and F. Glover. The Feasibility Pump. *Mathematical Programming*, 104:91–104, 2005.
14. M. Fischetti and A. Lodi. Local Branching. *Mathematical Programming*, 98:23–47, 2003.
15. C. A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, Oxford, England, 1995.
16. C. A. Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.
17. C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45:3–38, 2009.
18. P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 9:849–859, 1961.
19. P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem, Part II. *Operations Research*, 11:863–888, 1963.
20. I. Harjunkoski, V. Jain, and I. E. Grossmann. Hybrid mixed-integer/constrained logic programming strategies for solving scheduling and combinatorial optimization problems. *Computers and Chemical Engineering*, 24:337–343, 2000.
21. R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.
22. R. Howitt, S. Msangi, A. Reynaud, and K. Knapp. Using polynomial approximations to solve stochastic dynamic programming problems: or a "betty crocker" approach to SDP. Technical report, Department of Agricultural and Resource Economics, University of California at Davis, Davis, CA, 2002.
23. J. P. Ignizio. *Goal Programming and Extensions*. Heath, Lexington, Massachusetts, USA, 1976.

24. V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *IFORMS Journal on Computing*, 13:258–276, 2001.
25. E. L. Johnson, M. M. Kostreva, and U. H. Suhl. Solving 0-1 Integer Programming Problems arising from Large Scale Planning Models. *Operations Research*, 33:803–819, 1985.
26. J. Kallrath. Mixed-Integer Nonlinear Programming Applications. In T. A. Ciriani, S. Gliozzi, E. L. Johnson, and R. Tadei, editors, *Operational Research in Industry*, pages 42–76. Macmillan, Houndmills, Basingstoke, UK, 1999.
27. J. Kallrath. Modeling Difficult Optimization Problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2284–2297. Springer Verlag, New York, 2008.
28. J. Kallrath. Combined Strategic Design *and* Operative Planning in the Process Industry. *Computers and Chemical Engineering*, 33:1983–1993, 2009.
29. J. Kallrath. *Solving Difficult Real World Optimization Problems: Polylithic Modeling and Solution Approaches With GAMS Examples*. Vieweg-Teubner-Springer, Wiesbaden, Germany, in preparation., 2012.
30. J. Kallrath and T. I. Maindl. *Real Optimization with SAP-APO*. Springer, Heidelberg, Germany, 2006.
31. E. Kalvelagen. Branch-and-Bound Methods for an MINLP Model with Semi-Continuous Variables, 2003, discontinued on http://www.gams.com.
32. R. Karuppiah and I. E. Grossmann. A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *Journal of Global Optimization*, 41:163–186, DOI: 10.1007/s10898–007–9203–8, 2008.
33. X. Lin, C. A. Floudas, and J. Kallrath. Global Solution Approaches for Nonconvex MINLP Problems in Product Portfolio Optimization. *Journal of Global Optimization*, 32:417–431, 2005.
34. T. Lohmann. Practical Stochastic Optimization using Algebraic Modeling Systems. Master's thesis, Technische Universität Braunschweig, 2011.
35. M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.
36. V. Maniezzo, T. Stützle, and S. Voss. *Hybridizing Metaheuristics and Mathematical Programming*. Annals of Information Systems. Springer, Redwoord City, CA, 2010.
37. C. T. Maravelias and I. E. Grossmann. A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants. *Computers and Chemical Engineering*, 28:1921–1949, 2004.
38. R. Misener and C. A. Floudas. Piecewise-Linear Approximations of Multidimensional Functions. *Journal of Optimization Theory and Applications*, 145:120–147, 2010.
39. J. Muñoz, G. Gutierrez, and A. Sanchis. Evolutionary techniques in a constraint satisfaction problem: Puzzle Eternity II. In *Proceedings 2009 IEEE Congress on Evolutionary Computation*, pages 2985–2991, May 2009.
40. Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer, New York, 2006.
41. S. Rebennack, J. Kallrath, and P. M. Pardalos. Column Enumeration based Decomposition Techniques for a Class of Non-convex MINLP Problems. *Journal of Global Optimization*, 43:277–297, 2009.
42. S. Rebennack, A. Nahapetyan, and P. M. Pardalos. Bilinear Modeling Solution Approach for Fixed Charged Network Flow Problems. *Optimization Letters*, 3:347–355, 2009.

43. S. Rebennack, M. Oswald, D. O. Theis, H. Seitz, G. Reinelt, and P. M. Pardalos. A Branch and Cut Solver for the Maximum Stable Set Problem. *Journal of Combinatorial Optimization*, pages to appear, DOI:10.1007/s10878–008–9175–8, 2011.
44. S. Rebennack, G. Reinelt, and P. M. Pardalos. A Tutorial on Branch&Cut Algorithms for the Maximum Stable Set Problem. *International Transactions in Operational Research*, pages to appear, DOI:10.1111/j.1475–3995.2011.00805.x, 2011.
45. B. Roe, L. G. Papageorgiou, and N. Shah. A Hybrid MILP/CLP Algorithm for Multipurpose Batch Process Scheduling. *Computers and Chemical Engineering*, 29:1277–1291, 2005.
46. C. Romero. *Handbook of Critical Issues in Goal Programming*. Pergamon Press, Oxford, 1991.
47. M. W. P. Savelsbergh. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. *ORSA Journal on Computing*, 6:445–454, 1994.
48. M. W. P. Savelsbergh. Branch-and-Price: Integer Programming with Column Generation. In C. A. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, pages 218–221. Kluwer Academic Publishers, Dordrecht, Holland, 2001.
49. M. J. Schniederjans. *Goal Programming: Methodology and Applications*. Kluwer Academic Publishers, Boston, MA, 1995.
50. L. Schrage. *Optimization Modeling with LINGO*. LINDO Systems, Inc., Chicago, IL, 2006.
51. J. Till, G. Sand, S. Engell, M. Emmerich, and L. Schönemann. A New Hybrid Algorithm for Solving Two-Stage Stochastic Problems by Combining Evolutionary and Mathematical Programming Methods. In L. Puigjaner and A. Espuña, editors, *Proc. European Symposium on Computer Aided Process Engineering (ESCAPE) - 15*, pages 187–192, Dordrecht, North-Holland, 2005. Elsevier.
52. C. Timpe. Solving planning & scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.
53. J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-Integer Models for Nonseparable Piecewise-Linear Optimization: Unifying Framework and Extensions. *Operations Research*, Articles in Advance:1–13, 2009.