Optimisation Methods in Sustainable Manufacturing

Sebastian Schenker, Ingmar Vierhaus, Ralf Borndörfer, Armin Fügenschuh and Martin Skutella

1 Introduction

Sustainable manufacturing is driven by the insight that the focus on the economic dimension in current businesses and lifestyles has to be broadened to cover all three pillars of sustainability: economic development, social development, and environmental protection. In this chapter, we present two state-of-the-art approaches of mathematical optimisation and how they can be used to solve problems in sustainable manufacturing.

The multi-criteria perspective considers areas of sustainability as independent functions that are to be optimised however with divergent objectives simultaneously. Accordingly, computed outcomes that cannot be improved upon (on at least one objective without getting worse at another) are considered to be superior to outcomes that can be improved upon. A decision maker will only be interested in the first set of outcomes in order to be able to form an educated opinion with respect to his/her sustainability goal.

The system dynamics perspective on the other hand focuses on the time-dependent (or dynamic) aspects of systems that are influenced by sustainable manufacturing practices. If, for instance, a production technology was identified

S. Schenker (✉) · I. Vierhaus · R. Borndörfer
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: schenker@zib.de

A. Fügenschuh
Helmut-Schmidt-University, Holstenhofweg 85, 22043 Hamburg, Germany

M. Skutella
TU Berlin, Str. des 17. Juni 136, 10623 Berlin, Germany

that cannot be improved in either of the sustainability dimensions, the question then arises as to how this technology can be used in an optimal way using only limited resources. How can the impact on society and economy be steered in the direction of allowing the technology to be as beneficial as possible?

2 Multi-criteria Optimisation

Mathematical optimisation and mathematical programming is concerned with finding good solutions from a set of available alternatives. The abstract nature of mathematical optimisation allows the user to model a wide range of different problems and different objectives using the same theoretical insights and practical tools. Problems in sustainability and sustainable manufacturing have in common that there is not only one objective to be considered but several conflicting ones. This is mathematically reflected by considering several objective functions simultaneously. The set of available alternatives and the structure of the considered objective functions can generally be modelled in different ways. The focus in the following section is put on the well-studied and fruitful field of linear optimisation involving linear objective functions and linear constraints allowing the user to model as well as to efficiently solve a wide range of quantitative problems.

2.1 Multi-criteria Problem Formulation

In a *general* multi-criteria linear optimisation problem, one is given a set of k cost vectors $c_1, \ldots, c_k \in \mathbb{R}^n$ and seeks to minimize all linear cost functions $c_i \cdot x = \sum_{j=1}^{n} c_{i_j} x_j$, for $i = 1, \ldots, k$, simultaneously over all n-dimensional vectors $x = (x_1, \ldots, x_n)$ subject to a set of linear inequality and integer constraints. In particular, let M be some finite index set and suppose that for every $i \in M$, we are given an n-dimensional vector a_i and a scalar b_i. Let N_1, N_2 and N_3 be subsets of $\{1, \ldots, n\}$ that indicate which variables x_j are constrained to be non-negative, binary or integer, respectively. We then consider the problem

$$\begin{aligned} & \min(c_1 \cdot x, \ldots, c_k \cdot x) && \\ \text{s.t. } & a_i \cdot x \leq b_i, && i \in M, \\ & x_j \geq 0, && j \in N_1, \\ & x_j \in \{0, 1\}, && j \in N_2, \\ & x_j \in \mathbb{Z}, && j \in N_3. \end{aligned} \tag{1}$$

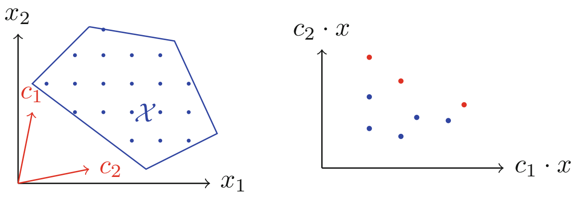

Fig. 1 Feasible space of a bi-criteria integer maximization problem and corresponding set in objective space with non-dominated points (*red*)

The variables $x_1, \ldots, x_n$ are called *decision variables* and a vector x satisfying all of the constraints is called a *feasible solution*. The set of all feasible solutions is called *feasible set* and will be denoted by $\mathcal{X}$. The image $y = (c_1 \cdot x, \ldots, c_k \cdot x)$ of a feasible solution x is called a *feasible point* and the set of all feasible points is called objective set and will be denoted by $\mathcal{Y}$. If N_1 coincides with $\{1, \ldots, n\}$ (implying $N_2 = N_3 = \emptyset$), then (1) is considered a *linear* programming problem. If $N_2 = \{1, \ldots, n\}$ or $N_3 = \{1, \ldots, n\}$, then we refer to (1) as a *binary* or *integer* programming problem, respectively. In case of $\emptyset \subsetneq N_1 \subsetneq \{1, \ldots, n\}$, (1) is considered a *mixed-integer* programming problem. The earliest investigations of multicriteria mathematical optimisation go back to the 1950s when the simplex method coined by Dantzig opened up a wide range of applications and prepared the ground for the huge success of linear programming (Dantzig 1963). If $k = 1$, then we refer to (1) as a single-objective problem and the notion of optimality is unambiguous. For a multi-criteria optimisation problem (with number of objectives $k \geq 2$) we cannot expect to find a solution that optimizes all objectives simultaneously leading to several possible notions of optimality in the multi-criteria case (Ehrgott 2005). A widely accepted (and in the following considered) one is the notion of *efficiency*. A solution $x^* \in \mathcal{X}$ is considered *efficient* if there is no other solution $x \in \mathcal{X}$ that achieves objective values at least as good with a strictly better value in at least one objective, i.e., there is no $x \in \mathcal{X}$ with $c_i \cdot x \leq c_i \cdot x^*$ for $i = 1, \ldots, n$ and $c_i \cdot x < c_i \cdot x^*$ for at least one $i \in \{1, \ldots, n\}$. The image of an efficient solution is called *non-dominated*. The challenge for a multi-criteria optimisation problem is then to compute all different non-dominated points (Figs. 1 and 2).

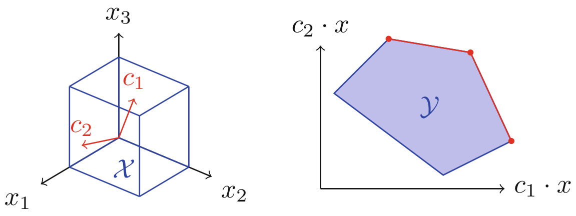

Fig. 2 Feasible space of a bi-criteria linear maximization problem and corresponding set in objective space with non-dominated points (*red*)

2.2 Manufacturing and Scheduling

Production problems, scheduling problems and similar decision problems are a fruitful domain for (mixed) integer programming. Binary variables might represent on-off decisions and linear or integer variables, respectively, might represent production quantities. In the following we will shortly present how multi-criteria integer programming could be used to model a scheduling problem that accounts for production costs, electricity consumption and worker satisfaction. Lets $[M] = \{1, \ldots, M\}$ be a finite set representing a set of different machines and let $[J] = \{1, \ldots, J\}$ be a finite set representing a set of jobs. We will consider a time horizon for the entire production process and let s and e be the start and end time of it. Introduce variables $x_{jmt} \in \{0, 1\}$ where $j \in [J]$, $m \in [M]$ and $t \in \{s, \ldots, e\}$. We set $x_{jmt} = 1$ if and only if starting time of job j on machine m is set to t. In order to model the constraint that every job needs to run on every machine before end time e, let $dur(m)$ the duration on machine m, i.e., the time that a job spends on machine m. Then,

$$\sum_{t=s}^{e-dur(m)} x_{jmt} = 1 \, \forall j \in [J] \wedge \forall m \in [M] \tag{2}$$

models the above fulfilment constraint. Furthermore, the constraint that job j is only allowed to run on machine $m+1$ if it is finished on machine m can be modelled via

$$\sum_{t=s}^{e} t \cdot x_{jmt} + dur(m) \leq \sum_{t=s}^{e} t \cdot x_{jm+1t} \forall j \in J \wedge \forall m \in [M-1] \tag{3}$$

Furthermore, it is very reasonable to assume that a new job can only be started on machine m if the previous job on machine m was finished. This constraint could be modelled via

$$\sum_{t=s}^{e} t \cdot x_{jmt} + dur(m) \leq \sum_{t=s}^{e} t \cdot x_{j+1mt} \forall j \in [J-1] \wedge \forall m \in [M] \tag{4}$$

2.3 Solving Multi-criteria Optimisation Problems

For the single-objective case there are several commercial solvers and software packages (CPLEX 2016; Xpress 2016; Gurobi 2016) and non-commercial ones (Achterberg 2009). One could have expected that the exponential growth in computing power and the even larger algorithmic speed-ups in mixed integer programming during the last decade (Bixby 2002) would automatically lead to multi-criteria extensions. But the situation is contrary: none of the available

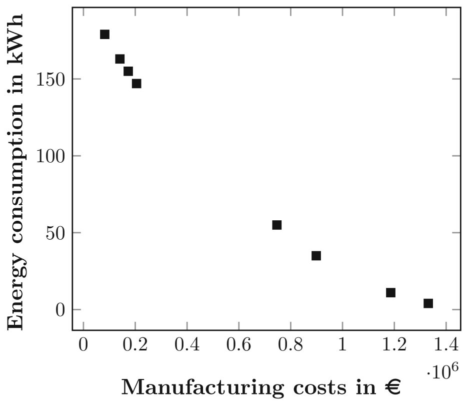

Fig. 3 Front of non-dominated points for a bi-criteria bicycle manufacturing problem

commercial solvers supports multi-criteria problems and there are only a few, recently developed non-commercial solvers available: BENSOLVE (Löhne and Weiing 2014) and inner (Csirmaz 2016) handle multi-criteria linear programming problems, SYMPHONY (Ladanyi et al. 2016) supports bi-criteria mixed integer problems and PolySCIP (Schenker et al. 2016) supports multi-criteria linear and integer problems.

PolySCIP reads problems of the above form (1) via its MOP file format which is based on the widely used MPS file format (MPS-Format 2016) and allows the user to model constraints like (2), (3), (4) easily via an algebraic modelling language (Koch 2004). It can handle an arbitrary number of objectives and thousands of variables and constraints (Fig. 3).

3 System Dynamics Optimisation

In this book, many technologies and approaches developed in the context of sustainable manufacturing are discussed. In this section, we will consider the global environment in which these technologies must be disseminated and implemented, in order to realise their positive potential.

The economy, the environment, and the society constitute complex entities and can be seen as finely balanced networks of mutual dependencies. Almost all components influence each other that have either supporting or weakening effects. Such dynamical systems can demonstrate counterintuitive behaviour. However, in order to bring about a change from the conventional production paradigm in the direction of a paradigm of sustainability, it is essential to appreciate the complex interdependencies of the systems involved.

We observe that the transition, i.e., the setup of many value creation modules and networks, constitutes a dynamic process over time that will span several years or decades. During this period, an array of interactions between the stakeholders need to be taken into account. Moreover, the transition does not take place by itself. It will only happen by means of deliberate influence on the system. A bundle of individual measures are necessary in this process.

To this end, the system dynamics (SD) approach provides the appropriate framework. It is an approach for the modelling and simulation of dynamical systems with a long history rooted in the understanding and teaching of dynamical systems in general, as well as in the field of sustainability.

After introducing system dynamics as a tool for simulation, we will formulate optimal control problems based on system dynamics models.

3.1 System Dynamics

In this section, we will introduce system dynamics as a modelling methodology as well as the most important modelling rules and characteristics of system dynamics models.

System dynamics was introduced by Jay Forrester in the 1950s as a method of describing and simulating time-dependent effects of complex influence networks with feedback loops (Forrester 1961). Such networks are characterized by non-linear, often surprising behaviour. In fact, a forecast of their future development, and thus their control, represents a difficult mathematical problem.

One of the strengths of the system dynamics approach lies in its visual representation of complex systems. This visual approach is essential in the system dynamics modelling process, and simplifies access for beginners and users who lack experience with systems of differential equations.

The main objects of system dynamics models are *stocks* and *flows*. The stocks contain the state information of the system. By convention, each stock has two flows, one flowing into the stock, and one flowing out of the stock. Figures 4 and 5 show visual representations of a stock and a flow respectively. As a third component, *auxiliary variables* are often introduced to structure a diagram. Lastly, the existence of functional dependencies between stocks, flows and variables is indicated by arrows. Figure 6 shows an example.

Using this visual representation, a systematic modelling process could be structured as follows:

- Definition of the modelling goal,
- Definition of the system limits,
- Definition of the system components,
- Definition of the direct relations between system components and the type of causal links (positive or negative),
- Design of an influence diagram to summarize components and their relations,

Fig. 4 Visual representation of a stock

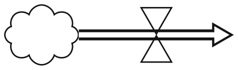

Fig. 5 Visual representation of a flow. The origin of the flow is outside of the limits of the system, as indicated by the *cloud symbol*. The *arrow* is decorated by an hourglass to indicate time dependency

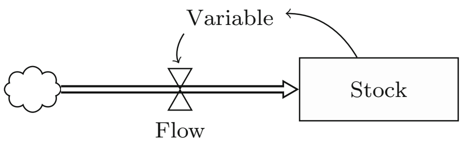

Fig. 6 Visual representation of a small model with one stock, one flow and one variable. The value of the flow depends on the value of the variable, which in turn depends on the value of the stock

- Creation of a system dynamics diagram with stocks for each of the system components as well as flows for each stock,
- Assignment of units and valid ranges to the values of stocks and flows,
- Definition of the functional relations between stocks and flows,
- Introduction of variables to simplify the relations if possible,
- Completion of the system dynamics diagram by adding variables and arrows for relations,

The result of this process is a complete system dynamics model. In the next section, we will discuss numerical methods for simulating a system dynamics model as it develops over time.

Although it is possible to find general solutions analytically for some models, this is generally neither possible nor required. A range of numerical simulation techniques exist that provide quickly accurate simulations. One class of such simulation techniques are the Runge-Kutta schemes (Runge 1895; Kutta 1901) which we will use in this chapter.

3.2 Optimal Control of System Dynamics Models

As we discussed in the previous sections, in its basic form, SD aims at describing and simulating influence networks. This is an important step in pursuit of understanding the mutual dependencies. In addition to obtaining a mere understanding however, what we would like to do is to intervene in the network, bring it to a desired stable state, or get as close as possible to that state.

In system dynamics, the points of the system which can be influenced by a conscious decision of an actor are modeled using the concept of *policies*.

Policies constitute a basic and important concept of system dynamics modelling. A policy is a function in some variables that describes the rates of flow in a system and hence the dynamic behaviour of the model (Richardson and Pugh 1981). Thus, a policy is a decision rule which specifies how a decision-maker processes available information from model variables (Sterman 2000). Questions regularly arise concerning whether a given policy can be improved, or even what a “good” policy “actually constitutes or entails. In this context, the need for efficient computational methods for policy analysis as well as policy improvement and design has been recognized in system dynamics, see, e.g., Yücel and Barlas (2011), Keloharju and Wolstenholme (1988), and is an active field of research.

When developing a simulation model, the modelling step of “policy formulation and evaluation” also compares the performance of two or more candidate policies (Sterman 2000). When two simulations with different policies lead to different system behaviors, one has to evaluate which of the two simulations is more suitable or “better” for a given model purpose. To answer this question, one needs to define an objective function so that the higher the value of the objective function for a given simulation, the more favorable or “better” the policy (Dangereld and Roberts 1996). Once an objective function is defined, several approaches to computer-aided policy improvement are at one’s disposal.

Direct parameter policy design starts with the definition of an analytic, parametrized, and usually nonlinear policy function (Keloharju and Wolstenholme 1989). The parameters of this function are set to starting values, and for each parameter, a range of valid values is defined. These parameters constitute then the free variables of the optimisation problem, i.e., the variables which can be varied freely in pursuit of an optimal solution. Consequently, the goal of the policy improvement is to find a set of parameter values within the given range that improves the value of the objective function. The solution space in this case is reduced by the *a priori* definition of the shape of the policy function. The solution found by the optimisation algorithm depends strongly on this definition and therefore on the expectations of the modeler. If a software package offers parameter optimisation capabilities, it is usually possible to attempt producing the solution of such direct parameter policy design problems.

Table function policy design is one possible way to generalizing direct parameter policy design, by defining a parametrized table function instead of an analytic function (Keloharju and Wolstenholme 1989). In this case, the modeler has to define the number of data points of the table function and two intervals that define the range of valid values of the data points on the *x*- and *y*-axis. This approach removes the modeler’s expectations of the shape of the policy from the optimisation process. However, the possible policies are reduced to the space of the piecewise linear functions with the selected number of points. If the data points are then required to have a pre-defined distance on the *y*-axis, the possible solutions are reduced further, but at the same time, the number of parameters and thus the number of free variables decreases. As in the previous case, the goal of the policy improvement is to find parameter values (i.e., data points of the table function), that improve the value of the objective function. A software package that supports table function policy design is found with the Powersim Studio plug-in SOPS (Moxnes and Krakenes 2005).

In both cases, the modeler has to define the functional dependencies of the policy function. This choice is closely related to the concept of bounded rationality (MoreCroft 1985; Simon 1984) models.

A policy function, i.e., a decision rule, is a model about what information cues an actor employs in order to make decisions in a given system. If this actor has only a limited view of the system, then the policy will only depend on the variables and information that are available to this particular actor (Sterman 2000). An improved policy will enable this actor to make better decisions based on the limited information available to him/her. Recent work has focused on improving policies for such actors, using, for instance, co-evolutionary analysis (Liu et al. 2012).

In this paper, we will consider a different kind of actor. Our actor has a global view of the model, i.e., he or she has information on all the state variables at all times within the simulation time horizon.

Modeling the policy of an actor with such a comprehensive level of awareness with the application of conventional approaches to policy analysis constitutes a difficult endeavor. One option would be to define a table function for each state, that depends only on that state. A mixed policy function that depends on all states, can then be defined as a sum of these functions (Keloharju and Wolstenholme 1989).

One conventional approach to System Dynamics optimisation is based on "optimisation by repeated simulation" (Liu et al. 2012). This has the advantage, that any model which can be simulated, can also be optimized, since there are no requirements on the properties of the model equations. However, approaches using repeated simulation suffer from the "curse of dimensionality" Bellman (2003) dynamic, where the significant dimension is that of the space of free variables. An additional free variable adds a dimension to the optimisation algorithm's search space. Solving optimisation problems with a large number of free variables therefore quickly becomes impractical. As a consequence, the degrees of freedom in a mixed policy function situation, are limited from a practical perspective, in the case of an optimisation of the policy by repeated simulation being attempted.

We present a different approach and in so doing, directly optimize the values of the policy function. This is equivalent to defining the policy as a time-dependent table function with one data point for each time step of the time horizon. In the context of physical systems, this kind of problem is known as an "optimal control problem" Betts (2011). With this approach no assumptions on the properties of the policy function are made *a priori*. It is only necessary to select the "free variables". In a conventional approach, these "free variables" would contain the values of the policy functions. For each of these variables, a range of valid values must be defined. It is then the task of the optimisation process, to find the optimal value for each free variable at each time.

The resulting optimisation problem based on a system dynamics model can be written as follows:

$$\begin{array}{ll} \max & c(x,y,z), \\ \text{s.t.} & \dot{x} = f(x,y,z), \\ & y = g(x,y,z), \\ & x(0) \in X_0 \\ & \\ \text{State variables:} & x = x(t) \in \mathbb{R}^n \\ \text{Algebraic variables:} & y = y(t) \in \mathbb{R}^m, \\ \text{Control variables:} & z = z(t) \in \mathbb{R}^s. \\ & \\ \text{Time horizon:} & t \in [t_i, t_f] \end{array}$$

In order to solve such a problem, we differentiate between two approaches:

3.2.1 Local Approach

In the local approach, the goal is to find a locally optimal solution. Local optimality means, that in a small neighborhood around the given solution, there is no solution with a better objective value. For this approach, standard methods exist for dynamical systems, which reliably deliver local solutions for small and moderately sized problems. The task at hand is to reformulate and adapt a system dynamics model, so that these methods can be used. Work on the local optimisation of system dynamics models can be found for instance in Vierhaus et al. (2014). In this chapter, we will focus only on the global approach.

3.2.2 Global Approach

In the global approach, the goal is to find a solution, and in addition to prove its global optimality. This means that no feasible solutions of the problem with a better objective function value exist. Hence, the global solution approach has two steps: Find an optimal solution and prove that no better solution exists.

Both of these approaches can prove successful using techniques from mathematical optimisation.

In the next section, we will show how modern optimisation techniques can be used in the global approach to system dynamics optimisation. The basis is the formulation of an optimisation problem, based on the control problem introduced in Sect. 3.2. As mentioned before, the simulation of a system dynamics model using numerical methods is well-established. This simulation is based on a time-discretisation of the model, which we will also use for our optimisation problems.

In order to discretise the model, we introduce a fixed time step of length Δt. We then consider the equations of (Sect. 3.2) no longer at any $t \in [0, T]$, but only at n_t points in time defined by $t = j \cdot \Delta t, \quad j \in \{0, 1, \ldots, n_t - 1\}$. The derivatives appearing in (Sect. 3.2) need to be replaced by an appropriate discretisation scheme, for example a Runge-Kutta scheme. The resulting system can then be written as follows:

$$\max c(x_0, \ldots, x_{n_t-1}, y_0, \ldots, y_{n_t-1}, z_0, \ldots, z_{n_t-1}), \tag{5a}$$

$$\text{s.t.}\, x_{j+1} = f(x_j, y_j, z_j), \quad j \in 0, 1, \ldots, n_t - 2 \tag{5b}$$

$$y_j = g(x_j, y_j, z_j), \quad j \in 0, 1, \ldots, n_t - 1 \tag{5c}$$

$$x_0 \in X_0 \tag{5d}$$

$$\text{State Variables: } x_j \in \mathbb{R}^n \tag{5f}$$

$$\text{Algebraic Variables: } y_j \in \mathbb{R}^m, \tag{5g}$$

$$\text{Control Variables: } z_j \in \mathbb{R}^s. \tag{5h}$$

This system now has the standard form of an optimisation problem, similar to the one introduced in (1). In contrast to (1), we now only have a single objective function. On the other hand, we have nonlinear equality constraints in place of linear inequality constraints.

3.3 MINLP Approach

After the discretization of the system dynamics optimisation problem, it is possible to attempt to solve it with existing solvers. Since we are interested in global solutions, the algorithm used should be able to provide a certificate of global optimality. One group of solvers that can provide this certificate are the branch-and-cut solvers that were introduced in Sect. 2.3 This approach has been successfully applied in the solution of Mixed Integer Linear Programs as well as MINLPs from a range of applications [for example, see Defterli et al. (2011), Borndörfer et al. (2013), Humpola and Fügenschuh (2013)]. Solving a control problem derived from a discretised dynamical system with a standard branch-and-cut solver is, however, in many cases unsuccessful, since the solver does not take into account the special structure of the MINLP that arises from the discretization, and from the handling of non-smooth functions via integer variables. Without considering this structure, even finding a single feasible solution can exceed a reasonable time budget of several hours or even days.

In the remainder of this section, we will present the concept of a tailored solver for system dynamics optimisation problems. Like PolySCIP, this concept has been implemented in the framework of the modern MINLP solver SCIP and results can be found in Fügenschuh and Vierhaus (2013a, b), Vierhaus et al. (2014),

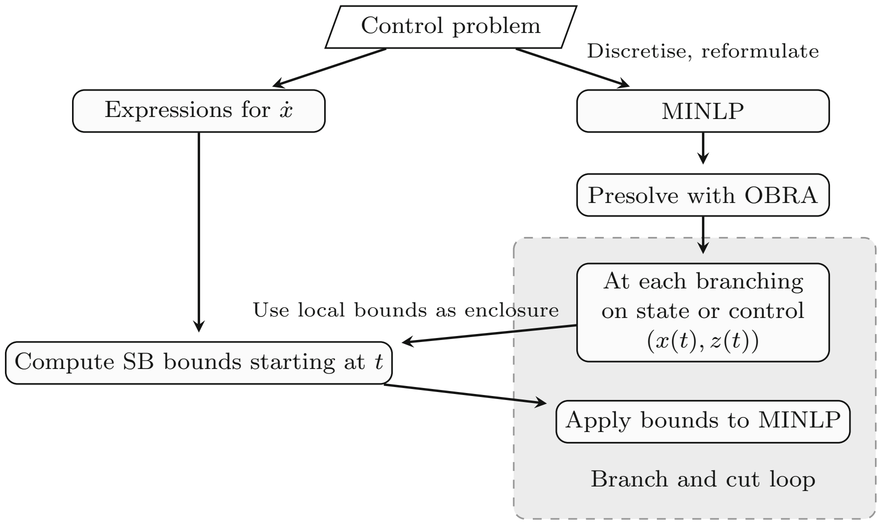
Control problem
Discretise, reformulate
Expressions for $\dot{x}$
MINLP
Presolve with OBRA
At each branching on state or control $(x(t), z(t))$
Use local bounds as enclosure
Compute SB bounds starting at t
Apply bounds to MINLP
Branch and cut loop

Fig. 7 Concept of a global solver for system dynamics optimization problems

Fügenschuh et al. (2013). A diagram describing the improved solution process is shown in Fig. 6.

3.3.1 Transcription

The first step is the reading and transcription of the system dynamics model and the optimisation parameters. Once the model and the optimisation parameters have been read, the optimisation model is processed in two ways. An equivalent MINLP is, then set up. This includes the time discretisation. At the same time, expressions for the function $\dot{x}(t)$ are derived from the model (Fig. 7).

3.3.2 Optimisation Based Reachability Analysis

To improve on the dual side of the algorithm, an Optimisation Based Reachability Analysis (OBRT) is performed for every problem. This analysis computes bounds for the possible states of the system using the dynamic behaviour and the initial values x_0 as input.

3.3.3 Primal Heuristic

In the interest of producing quickly feasible solutions, we implemented a simple heuristic that reduces the control problem to a simulation problem by fixing the control variables to their lower (or in a second run upper) bounds. If there are no path constraints, this process will always yield a feasible solution.

3.3.4 Bound Propagation Based on Differential Inequalities

To improve the bounds within the branch-and-cut process, we compute differential inequalities as outlined in Scott and Barton (2013). This involves the solution of an auxiliary simulation problem using the expressions for $\dot{x}$ derived in the reading of the problem.

3.3.5 System Dynamics SCIP

The concepts mentioned above have been implemented as the solver System Dynamics SCIP (SD-SCIP). Like polyscip, SD-SCIP is an extension of the modern MINLP solver SCIP and is publicly available (Füegenschuh and Vierhaus 2013a, b).

4 Conclusion

This chapter introduced the framework of multi-criteria optimization and system dynamics optimisation together with different modelling techniques. It showed that mathematical optimisation is a useful tool for modelling a wide variety of problems from the sustainability context. The two solvers presented PolySCIP (Schenker et al. 2016) and SD-SCIP (Fuegenschuh and Vierhaus 2013a, b) were specifically developed with applications from sustainability in mind. They can be used as decision support instruments for a wide range of problems, from scheduling, manufacturing and production to planning subsidies and taxes and exploring dynamical pathways into the future. Both tools are publicly available and present an opportunity for the sustainability community to benefit from recent advances in mathematical optimisation.

References

Achterberg, T. 2009. Scip: Solving constraint integer programs. *Mathematical Programming Computation* 1(1): 1–41. http://mpc.zib.de/index.php/MPC/article/view/4.

Bellman, R. 2003. *Dynamic programming*. Dover Books on Computer Science Series. Dover Publications.

Betts, J.T. 2010. *Practical methods for optimal control using nonlinear programming. Advances in design and control*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Bixby, R.E. 2002. Solving real-world linear programs: A decade and more of progress. *Operations Research* 50: 3–15.

Borndörfer, R., A. Fügenschuh, T. Klug, T. Schang, T. Schlechte, and H. Schülldorf. 2013. The freight train routing problem. Technical report, ZIB Technical Report ZR-13-36.

CPLEX. 2016. IBM ILOG CPLEX Version 12.6.3. www-03.ibm.com/software/products/en/ibmilogcpleoptistud.

Csirmaz, L. 2016. Inner. https://github.com/csirmaz/inner.

Dangerfield, B., and C. Roberts. 1996. An overview of strategy and tactics in system dynamics optimization. *Journal of the Operational Research Society* 47: 405–423.

Dantzig, G. 1963. *Linear programming and extensions*. Rand Corporation Research Study. Princeton University Press.

Defterli, O., A. Fügenschuh, and G.-W. Weber. 2011. Modern tools for the time-discrete dynamics and optimization of gene-environment networks. *Communications in Nonlinear Science and Numerical Simulation* 16(12): 4768–4779.

Ehrgott, M. 2005. *Multicriteria optimization*, 2nd ed. Berlin: Springer.

Fügenschuh, A., and I. Vierhaus. 2013. A global approach to the optimal control of system dynamics models.

Forrester, J.W. 1961. *Industrial dynamics*. Waltham, MA: Pegasus Communications.

Fügenschuh, A., S.N. Grösser, and I. Vierhaus. 2013. A global approach to the control of an industry structure system dynamics model. Technical Report 13-67, ZIB, Takustr.7, 14195 Berlin.

Fügenschuh, A., and I. Vierhaus. 2013. A global approach to the optimal control of system dynamics models. Technical report, ZIB Technical Report ZR-13-28.

Gurobi. 2016. GUROBI Optimization Version 6.5. www.gurobi.com.

Humpola, J., and A. Fügenschuh. 2013. A unified view on relaxations for a nonlinear network flow problem. Technical report, ZIB Technical Report ZR-13-31.

Keloharju, R., and E. Wolstenholme. 1988. The basic concepts of system dynamics optimization. *Systemic Practice and Action Research* 1(1): 65–86.

Keloharju, R., and E. Wolstenholme. 1989. A case study in system dynamics optimization. *Journal of the Operational Research Society* 40(3): 221–230.

Koch, T. 2004. *Rapid mathematical prototyping*. Ph.D. thesis, Technische Universität, Berlin.

Kutta, W. 1901. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. B.G Teubner.

Ladanyi, L., T. Ralphs, M. Guzelsoy, and A. Mahajan. 2016. SYMPHONY 5.6.14. https://projects.coin-or.org/SYMPHONY.

Liu, H., E. Howley, and J. Duggan. 2012. Co-evolutionary analysis: A policy exploration method for system dynamics models. *System Dynamics Review* 28(4): 361–369.

Löhne, A., and B. Weißing. 2014. Bensolve 2. http://www.bensolve.org.

Morecroft, J.D.W. 1985. Rationality in the analysis of behavioral simulation models. *Management Science* 31(7): 900–916.

Moxnes, E., and A. Krakenes. 2005. SOPS—a tool to find optimal policies in stochastic dynamic systems. In *Proceedings of the 23rd International Conference of the System Dynamics Society*, ed. by J.D. Sterman, N.P. Repenning, R.S. Langer, J.I. Rowe, and J.M. Yanni.

MPS Format. 2016. Mps format—Wikipedia, the free encyclopedia. Accessed 28 July 2016.

Richardson, G.P., and A.L. Pugh. 1981. *Introduction to system dynamics modeling with dynamo*. Cambridge, MA: MIT Press.

Runge, C. 1895. Ueber die numerische Auflösung von Differentialgleichungen. *Mathematische Annalen* 46: 167–78.

Schenker, S., R. Borndörfer, M. Skutella, and T. Strunk. 2016. PolySCIP. In *Mathematical Software –ICMS 2016, 5th International Congress, Proceedings*, ed. by G.-M. Greuel, T. Koch, P. Paule, and A. Sommese, vol 9725 of *LNCS*, Berlin, Germany. Springer. http://polyscip.zib.de.

Scott, J.K., and P.I. Barton. 2013. Bounds on the reachable sets of nonlinear control systems. *Automatica* 49(1): 93–100.

Simon, H.A. et al. (1984). *Models of bounded rationality*, volume 1: economic analysis and public policy. MIT Press Books 1.

Sterman, J.D. 2000. *Business dynamics—systems thinking and modeling for a complex world*. Boston: Irwing McGraw-Hill.

Vierhaus, I., A. Fügenschuh, R.L. Gottwald, and S. Groesser. 2014. Modern nonlinear optimization techniques for an optimal control of system dynamics models. In *Proceedings of The 32nd International System Dynamics Conference*.

Xpress. 2016. FICO Xpress Optimization Suite Version 7.9. www.fico.com/en/products/fico-xpress-optimization-suite.

Yücel, G., and Y. Barlas. 2011. Automated parameter specification in dynamic feedback models based on behavior pattern features. *System Dynamics Review* 27(2): 195–215.