GCSM
11th Global Conference on Sustainable Manufacturing
2013
23rd - 25th September
Berlin - Germany

17.2 System dynamic optimization in the sustainability assessment of a world-model

A. Fügenschuh[1], I. Vierhaus[2]

[1] Helmut Schmidt University / University of the Federal Armed Forces Hamburg, Germany

[2] Department Optimization, Zuse Institute Berlin, Germany

Abstract:
The System Dynamics (SD) methodology is a framework for modeling and simulating the dynamic behavior of socioeconomic systems. Characteristic for the description of such systems is the occurrence of feedback loops together with stocks and flows. The equations that describe the system are usually nonlinear. Therefore seemingly simple systems can show a nonintuitive, nonpredictable behavior over time. Controlling a dynamical system means to define a desired final state in which the system should be, and to specify potential interventions from outside that should keep the system on the right track. The central question is how to compute a globally optimal control? We propose a branch-and-bound approach that is based on a bound propagation method, primal heuristics, and spatial branching. We apply our new SD-control method to a model that describes the evolution of a social-economic system over time. We examine the problem of steering this system on a sustainable consumption path.

Keywords: System Dynamics; Mixed-Integer Nonlinear Optimization

1 INTRODUCTION

In sustainability science, the object of examination and the manipulation are often complex, large and dynamical systems [1]. These systems can be found on the macroscopic socioeconomic scale [2] as well as on the microscopic level in individual manufacturing processes (see [3] and the references therein). The reaction of these systems to external interventions can be counter intuitive, so that even when a quantitative goal has been identified and resources have been dedicated to achieving this goal, its implementation is non-trivial [4].

For this reason, the study of complex dynamical systems is an essential part of sustainability research. A traditional and successful approach to modeling dynamical systems is the System Dynamics (SD) methodology. SD was the basis for the report “Limits to Growth” [5], which introduced one of the first concepts of sustainability. For an introduction to SD and its applications we refer to Sterman [6].

SD models describe the behavior of a system that consists of several interrelated stocks, flows and feedback loops. The relation between those is usually determined by ordinary differential equations, nonlinear functional relations, logical relations (such as if-then-else), or tabular data. Even if each of these relations is individually well understood, the interplay of several of these relations can show a surprising, unexpected behavior in a simulation over time.

A System Dynamics model together with control functions and a real-valued objective function is called a System Dynamics Optimization (SDO) problem in the sequel. The need for an integration of optimization methods into the SD framework has been recognized already in the past. In previous approaches, different methods are used, such as nonlinear local optimization (for example, gradient search methods) or heuristics (such as genetic algorithms), which are essentially based on an “optimization through repeated simulations” approach, see [7, 8]. All these approaches have in common that they at best only provide feasible or local optimal solutions. Moreover, as pointed out by Burns and Janamanchi [9], nonlinear optimization methods rely on the availability of derivative information from sufficiently smooth functions. This restriction is in conflict with certain SD modeling elements, such as if-then-else clauses. In principle, all these methods cannot give a proof of global optimality or any other quality certificate of the computed solutions. The ultimate goal of our research is to fill this gap by developing a computational method that is able to handle all kinds of SDO models and yields feasible solutions as well as a certificate of optimality.

In the following we describe our approach to solve mixed-integer dynamic optimization problems that originate from SDO models. As a foundation we use the linear and nonlinear programming based mixed-integer

G. Seliger (Ed.), Proceedings of the 11th Global Conference on Sustainable Manufacturing - Innovative Solutions
ISBN 978-3-7983-2609-5

nonlinear solver SCIP [10]. This framework already handles the input of an instance, the set up of the model's variables and constraints, and the overall branch-and-cut solution process. Initially, we tried to solve SDO model instances using SCIP as a black-box solver off the shelf. Although SCIP is a highly capable general-purpose solver, it became clear that the special constraint structure of the SDO is for the most part intractable for the solver. However, if this structure is known and exploited to a high degree, the solution process can be fundamentally improved. In this paper, we will address the two aspects of the branch-and-bound solution approach, with the highest potential of improvement.

2 THE MINI-WORLD MODEL

We demonstrate our proposed methods using the *Mini-world* model introduced in Bossel [11]. It is a simplification of the much more sophisticated World3 model of Meadows et al. [5]. World3 uses 18 stocks, 60 parameters, 52 tabular functions, and around 200 equations to model the system's relations. Bossel's Miniworld is an aggregated version, that comes with just 3 stocks: the world population, the production (industrial, commercial, agricultural), which equals the consumption of goods, and the environmental pollution. Consequently, the number of parameters, tables, and equation relations are much lower. Interestingly, the model shows a qualitatively similar behavior compared to the much more evolved World3 model.

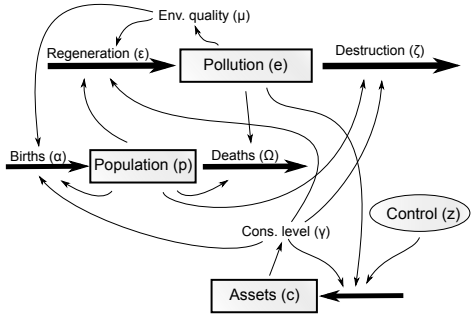

Figure 1: System Dynamics diagram for the mini-world control problem.

As discussed in Section 1, we will extend this model to an SDO, by introducing an objective function and a control mechanism. Our goal is to maximize the economic growth level, in order to provide a high standard of living to the world's population. However, if this standard would be too high, then a fast growing population would quickly consume its natural resources, and thus the population would collapse soon after. To prevent such behavior we introduce a population level ξ as a lower limit. The question we want to address is, how much sustainable growth can bear this mini-world at most for a population being at least ξ. The SDO diagram is shown in Figure 1. The corresponding mathematical model has four time-dependent functions: $p(t)$ for the population, $e(t)$ for the environmental pollution, $c(t)$ for the consumption, and $z(t)$ for the control function (the growth level).

2.1 MINLP Formulation

We apply an Euler forward discretization to the differential terms, with (Δt) being the size of a step in the time discretization. The continuous variables $p_i, e_i, c_i, z_i \in \mathbb{R}$ for $i = 0, 1, \ldots, T$ approximate the functions of the respective same name. Furthermore, we introduce the following real-valued continuous variables, each for $i = 0, 1, \ldots, T$: α_i are the number of births and Ω_i are the number of deaths at time step i. γ_i is the consumption level. μ_i describes the environmental quality. The environmental conditions change over time, and for this we introduce ε_i for the environmental recovery and ζ_i for the environmental destruction. The discretized SDO problem then reads as follows:

$$\max \sum_{i=0}^{T} z_i \tag{1a}$$

$$s.t.\ \frac{p_{i+1} - p_i}{\Delta t} = \alpha_i - \Omega_i,\ \ i = 0, 1, \ldots, T, \tag{1b}$$

$$\alpha_i = 0.03 \cdot p_i \cdot \mu_i \cdot \gamma_i,\ \ i = 0, 1, \ldots, T, \tag{1c}$$

$$\Omega_i = 0.01 \cdot p_i \cdot e_i,\ \ i = 0, 1, \ldots, T, \tag{1d}$$

$$\frac{e_{i+1} - e_i}{\Delta t} = \varepsilon_i - \zeta_i,\ \ i = 0, 1, \ldots, T, \tag{1e}$$

$$\zeta_i = 0.02 \cdot p_i \cdot \gamma_i,\ \ i = 0, 1, \ldots, T, \tag{1f}$$

$$e_i = e_i^+ - e_i^- + 1.0,\ \ i = 0, 1, \ldots, T, \tag{1g}$$

$$e_i^+ \leq 20.0 \cdot x_i,\ \ i = 0, 1, \ldots, T, \tag{1h}$$

$$e_i^- \leq 1.0 - x_i,\ \ i = 0, 1, \ldots, T, \tag{1i}$$

$$\varepsilon_i = 0.1 \cdot (1.0 - e_i^-),\ \ i = 0, 1, \ldots, T, \tag{1j}$$

$$1.0 = e_i \cdot \mu_i,\ \ i = 0, 1, \ldots, T, \tag{1k}$$

$$\frac{c_{i+1} - c_i}{\Delta t} = 0.05 \cdot \gamma_i \cdot e_i \cdot \left(1 - \left(\frac{\gamma_i \cdot e_i}{z_i}\right)\right),\ \ i = 0, 1, \ldots, T, \tag{1l}$$

$$c_i = \gamma_i,\ \ i = 0, 1, \ldots, T, \tag{1m}$$

$$p_i \geq \xi,\ \ i = t, t+1, \ldots, T, \tag{1n}$$

$$p_0 = \bar{p}, \tag{1o}$$

$$e_0 = \bar{e}, \tag{1p}$$

$$c_0 = \bar{c}, \tag{1q}$$

$$p_i, e_i, e_i^+, e_i^-, c_i, z_i, \in \mathbb{R}_+,\ \ i = 0, 1, \ldots, T, \tag{1r}$$

$$\alpha_i, \Omega_i, \gamma_i, \mu_i, \theta_i, \varepsilon_i, \zeta_i \in \mathbb{R}_+,\ \ i = 0, 1, \ldots, T, \tag{1s}$$

$$x_i \in \{0, 1\},\ \ i = 0, 1, \ldots, T. \tag{1t}$$

The change in the population p from time step i to $i+1$ is a result of the births α_i minus the deaths Ω_i in time step i (1b). The number of births α_i in time step i is proportional to size of the population p_i, the environmental quality μ_i, and the consumption level γ_i, where 0.03 is

the proportionality factor (birth rate) (1c). The number of deaths Ω_i in time step i is proportional to the population p_i and the environmental pollution e_i, with a proportionality factor (death rate) of 0.01 (1d).

The change in the environmental pollution e from time step i to $i+1$ is a result of the negative environmental destruction ζ_i and the positive environmental recovery ε_i in time step i (1e). The environmental destruction ζ_i in time step i is proportional to the population size p_i and the consumption level γ_i (1f). Here 0.02 is the proportionality factor (environmental destruction rate). The environment is able to recover over time. If the environmental quality μ_i is above a certain threshold value (here 1.0), then the recovery ε_i in time step i is proportional to the environmental pollution e_i, with 0.1 being the proportionality factor (recovery rate). However, if the environmental quality μ_i is below the threshold value, then the recovery rate is no larger than 0.1 (1j). This is equivalent to a minimum function $0.1\min(1,\mu_i)$, which is implemented with the auxiliary variables e_i^+, e_i^- and x_i and the additional constraints (1g), (1h) and (1i).

The change in the production and consumption c from time step i to $i+1$ is the result of a logistic growth function of Verhulst type (1l), which depends on both the consumption level γ_i and the environmental pollution e_i. The control variable z_i plays the role of the system's capacity (this is a constant in the original Verhulst equation). The constant value of 0.05 is a growth rate for the consumption. The consumption level γ_i equals the production c_i (1m).

The population p_i must not fall short of the given level ξ_i in each time step i (1n). Initially, in time step $i=0$, the size of the population (1o), the environmental pollution (1p), and the consumption resp. production (1q) are given.

The system of equations and inequalities (1) completely defines the model. Setting $z_i = 0$ for all times i and choosing initial values p_0, e_0 and c_0 reproduces the Miniworld simulation or initial value problem (IVP).

2.2 Constraint Classification

We distinguish five types of constraints, that are typical for any SDO:

The discretized differential equations (1l), (1b), (1e) define the state variables p_i, e_i, c_i at a given time, in terms of the previous time.

The *explicit algebraic equations* (1c), (1d), (1f), (1j), (1k), (1k), (1m), define the algebraic variables on the left hand side. These equations depend on coinciding states and other coinciding algebraic variables.

The equations (1g), (1h) and (1i) define a *system of algebraic equations and inequalities* which implicitly but uniquely defines the algebraic variables e_i^+ and e_i^-.

Finally, equation (1n) defines a *linear state constraint*, and equations (1p),(1p) and (1q) defines *initial values* as mentioned above.

3 IMPROVING THE SOLUTION OF SDOS

In this section we will describe the proposed presolving bound propagation method and the proposed primal heuristic.

3.1 Primal Heuristics

Linear programming based branch-and-bound methods for solving mixed-integer linear and nonlinear problems are on the one hand today's method of choice for verifying the global optimality of a solution to a given instance of some model. On the other hand, they are also known for their actual weakness in creating such solutions. A pure branch-and-bound method can only come up with a feasible solution, if the linear programming relaxation of the current node is feasible for the non-relaxed model. This, however, is a rare exception. Thus it is necessary to provide feasible solutions from other sources. Having a good feasible solution early in the solution process allows to prune the search tree. Ideally, large portions of the otherwise exponential sized search tree do not have to be created and examined.

In modern MINLP solvers a large variety of different search strategies is used to construct feasible solutions, see [12–14] and the survey [15], for instance. These methods are designed and applied to general purpose MINLP problems. As reported by the respective authors, they are good in identifying solutions on a wide range of different problems. Some of them are readily available within SCIP. However, these have problems in finding solutions for dynamic MINLPs. This motivated our approach, which is presented in the following.

As described in Section 2, setting all controls in an SDO to zero, results in an IVP. If no path constraints like (1n) exist, then every solution of the IVP is a feasible solution of the SDO. From this, a fast primal heuristic for the model can be constructed. We set the control to a constant value for all times $z_i = z_c$ and then iterate the following steps for each discretized time i except $i = T$. We start at $i = 0$, where the states are given by the initial values:

1. Calculate the values of γ_i, μ_i, from the values of the state variable. Since every variable on the right hand side is fixed, we can treat the corresponding constraints as an equation with a unique solution.
2. Calculate the values of $\alpha_i, \Omega_i, \epsilon_i, \xi_i$, from the values of the states and the already calculated algebraic variables, by treating the corresponding constraints as equations as before.
3. Set $e_i^- = -\min(1,\mu_i), e_i^+ = \mu_i - e_i^-$ and x_i to 1 if $\mu_i > 1$, otherwise to 0.
4. Calculate the values of the state variables in $i+1$, using the values computed in steps 1, 2, 3.

This procedure can be easily generalized to general SDOs.

In one run of our current implementation, we attempt to construct two solutions fo the problem by setting $z_i = \bar{z}$ in the first, and $z_i = \underline{z}$ in the second run.

As mentioned above, this method may not yield a feasible solution if there are state constraints. To react to these cases, we add a check at the end of the propagation at each time if any state constraints are violated by the states calculated in the current step. If a violation is detected, we flip the control from $\bar{z}$ to $\underline{z}$ in the first run, and from $\underline{z}$ to $\bar{z}$ in the second run. It is also possible to backtrack a given number of timesteps, and apply the flip at an earlier point.

3.2 Presolving Bound Propagation

From the modelers perspective, it appears unnatural to give bounds to the states of the system. The definition of intial states or bounds on the initial states and bounds on the control already defines a finite state space, that should not be reduced by giving additional bounds. However, finding the exact bounds defining that state space is usually not possible in finite time. On the other hand, tight bounds are essential in speeding up the branch-and-bound process for two reasons. First, we apply linear programming to solve the subproblems in the branch-and-bound tree. If the variable bounds are weak, then the solution of the linear programs will also yield weak (upper) bounds for the solution of the MINLP. Second, when branching on a variable, it needs more branching decisions to fix a variable when the bounds are weak. Both issues are related to the solution speed and the memory requirement during the solution process. Hence in general, one is interested in obtaining good (narrow) bounds on all variables, assuming that the time spend for computing them is marginal compared to the gain in solving the overall problem.

A crucial step in solving mixed-integer linear or nonlinear problems is the preprocessing phase. In this initial step of the solution process, one tries to extract as much information as possible from the problem at hand. These information are implicitly contained in the model formulation, but hidden for the solver. The goal is to make them visible, so that the solver can use them to shorten the solution process. Preprocessing or presolving refers not just to a single method, but a whole bunch of various techniques, where some are more general, and others are more specific for certain problem structures. For more details and general surveys to MILP and MINLP solution techniques including preprocessing, we refer to [15–19].

In this section, we consider bound propagation in presolving. The usual approach for MINLPs is to consider each constraint individually, and to attempt the deduction of tighter bounds on each variable of the constraint from the bounds of the other variables [20]. For our model, this approach yields bounds that quickly diverge after the first few timesteps. This phenomenon of "exploding bounds" is characteristic for SDO problems, and could be avoided by consiering the full problem and solving the maximum and minimum problems for each state variables. However, the computational effort would be comparable to the original control problem, rendering this approach useless.

We propose a compromise between these two approaches. Instead of considering a single constraint, we formulate subproblems, by selecting constraints at a given time. We introduce the *lookback parameter* $h \geq 0$, which defines how far the problem extends in time. We define the subproblem at time i with lookback parameter h, as the subproblem $s_{t,h}$ consisting of the constraints (1b) through (1n), where $i \in \{t-h, t-h+1, \ldots, t\}$.

Our proposed bound propagation method iterates the following steps for each discretized time i except $i = T$, starting at $i = 0$:

1. Formulate the subproblem $s_{i,h}$.
2. For each algebraic variable v_i at time i:
   a) Solve the maximization problem $\max v_i$ subject to the constraints of $s_{i,h}$ to optimality and set the upper bound of the variable $\bar{v}_i$ to the solution value.
   b) Solve the minimization problem $\min v_i$ subject to the constraints of $s_{i,h}$ to optimality and set the lower bound of the variable $\bar{v}_i$ to the solution value.
3. For each differential variable w_{i+1} at time $i+1$:
   a) Solve the maximization problem $\max w_{i+1}$ subject to the constraints of $s_{i,h}$ to optimality and set the upper bound of the variable $\bar{w}_{i+1}$ to the solution value.
   b) Solve the minimization problem $\min w_{i+1}$ subject to the constraints of $s_{i,h}$ to optimality and set the lower bound of the variable $\bar{w}_{i+1}$ to the solution value.

The subproblems are solved with a time limit. If the time limit is reached before the problem is solved to optimality, the considered bound is set to the best dual bound.

4 COMPUTATIONAL RESULTS

For our computational experiments, we introduce a piecewise constant control with two states z_0 and z_1, where z_0 is valid for the first $T/2$ many time steps, an z_1 is valid for the second half of $T/2$ time steps. Then the objective function is a two-dimensional function depending on the values z_0, z_1 for the two constant controls. Figure 4 shows a contour plot of the objective function values for the two controls varying independently in the interval $[1, 10]$. Feasible solutions are in the lower-left corner. The black curve marks the borderline between the feasible and the infeasible region. Our goal is to keep the mini-world population above $\xi \geq 4$ (billion inhabitants). Any solution having less inhabitants is infeasible. Solutions in the infeasible region all fall short of

the population lower limit of constraint (1n). A possibly optimal solution occurs for $z_0 = 1.79$ and $z_1 = 2.16$. This solution yields the highest total consumption level, while keeping the population size above the specified limit.

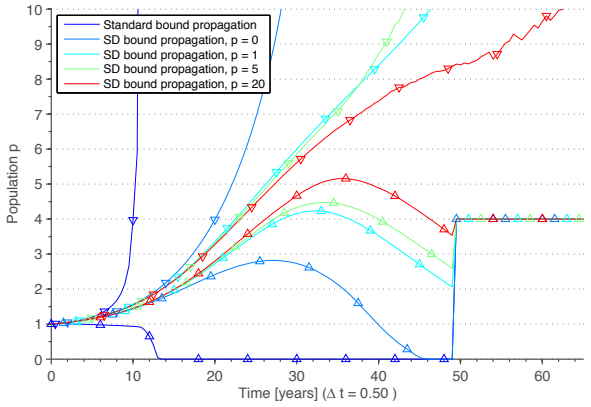

Figure 2: Presolved bounds for different look-back levels.

In Figure 3 we show three different solutions. The first for $z_0 = z_1 = 1$ is feasible but not optimal. The second for $z_0 = z_1 = 10$ is infeasible. (These two reference plots can also be found in Bossel [21].) Our potential optimal solution for $z_0 = 1.79$ and $z_1 = 2.16$ is shown as third plot.

Note that the plot in Figure 4 and the identification of an optimal solution is based on a number of simulation runs. Even if we use a fine grid, we cannot be absolutely sure that there is not a better solution that is hidden between two neighboring grid points. Our goal is to demonstrate that by using our bounding techniques we can cut off large portions of the search space. It is thus guaranteed that no better solution can be found in such cut off part.

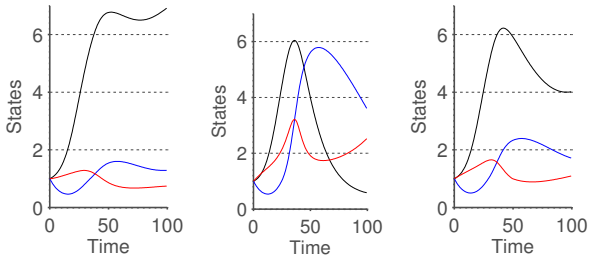

Figure 3: Black lines: p, blue: e, red: c. a): Feasible but suboptimal solution with control $z_0 = z_1 = 1$. b) Infeasible solution with control $z_0 = z_1 = 10$. c) Optimal solution with $z_0 = 1.79$ and $z_1 = 2.16$.

We apply our bounds strengthening presolve routine. The results for different look-back levels are shown in Figure 2 and Table 1. The SCIP-level (first line in Table 1) is special: The solver was very fast, but due to its numerical instability "proved" that the problem is infeasible (i.e., does not have a feasible solution), which is of course not true.

	presolve		branch-and-bound	
level	total time [h]	dual	dual	gap
SCIP	0.002	–	–	–
0	1.33	$6 \cdot 10^9$	625	$\gg 100\%$
1	4.42	640	638	71.1%
2	2.702	581	579	68.1%
5	4.53	508	506	63.5%
10	9.01	479	476	61.3%

Table 1: Computational results for different presolve look-back levels. Primal bound during branch-and-bound was 184.655 for all instances. Running time of the branch-and-bound runs was one hour.

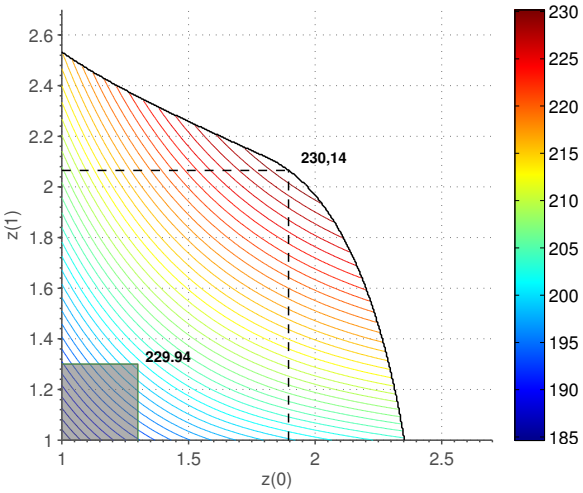

Figure 4: Objective function values for piecewise constant controls varying in $[1, 10] \times [1, 10]$. The (unique) optimal solution is in $(1.79, 2.16)$. The part of the control space with $z_1 \leq 1.3$ and $z_2 \leq 1.3$ marked by the gray box can be cut off. No better feasible solutions can be found in here.

A part of the search space that does not require further atention due to the bounding argument is shown in Figure 4. Here a look-back presolve level of 5 time steps was applied. Our optimal solution for $z_0 = 1.79$ and $z_1 = 2.16$ has an objective function value of 230.14. Any solution in the box defined by the green lines has an upper bound on the objective function value of less-or-equal 229.94, hence no better solution can be found in this part of the control space.

At the present stage of our implementation, we are, however, not able to prove that our potential optimal solution is indeed a global optimal one. The solver SCIP terminates, even in the highest presolve level, with a huge gap of 61.26%. Our ongoing research is devoted to close this gap to 0% within the given time limit, and to solve the problem for a control function with more than two constant segments.

5 SUMMARY AND CONCLUSIONS

We presented a System Dynamics Optimization (SDO) as an extension of classical System Dynamics Simulation (SD) model, by introducing a control mechanism and an objective function into the system. To solve such problems, we suggest mixed-integer nonlinear programming (MINLP) methods. We tested our methods on the mini-world SDO problem, and presented the results for several parameters.

We believe that our methods are generally applicable to all different kinds of SDO models. At the present stage of our work, each model needs to be individually analyzed and treated. We are working towards a black-box solver that a model will be able to use for his or her models in the same way SD simulation tools are used today. We think that having not only good-looking solutions, but additionally a certificate of optimality, is a further asset in practical applications.

ACKNOWLEDGEMENT

We gratefully acknowledge the funding of this work by the German Research Association (Deutsche Forschungsgemeinschaft, DFG), Collaborative Research Center CRC1026 (Sonderforschungsbereich SFB1026). The first coauthor conducted parts of this research under a Konrad-Zuse-Fellowship.

References

[1] Todorov, V., Marinova, D., 2011. Modelling sustainability. Math. Comput. Simul., 81(7):1397–1408.

[2] Scoones, I., 2007. Dynamic systems and the challenge of sustainability. STEPS working paper. STEPS Centre.

[3] Kibira, D., Jain, S., McLean, C. R., 2009. A System Dynamics Modeling Framework for Sustainable Manufacturing. In Proceedings of the 27th Annual System Dynamics Society Conference.

[4] Ghaffarzadegan, N., Lyneis, J., Richardson, G. P., 2011. How small system dynamics models can help the public policy process. Syst. Dyn. Rev., 27(1):22–44.

[5] Meadows, D., Meadows, D., Randers, J., Behrens III, W., 1972. The Limits to Growth. Universe Books, New York, NY.

[6] Sterman, J. D., 2000. Business Dynamics – Systems Thinking and Modeling for a Complex World. Irwing McGraw-Hill, Boston.

[7] Dangerfield, B., Roberts, C., 1996. An Overview of Strategy and Tactics in System Dynamics Optimization. Journal of the Operational Research Society, 47:405–423.

[8] Fu, M., 1994. Optimization via simulation: A review. Annals of Operations Research, 53(1):199–247.

[9] Burns, J. R., Janamanchi, B., 2007. Optimal Control and Optimization of System Dynamics Models: Some Experiences and Recommendations. In Proceedings of the 2007 Meeting of the Southwest Region Decision Sciences Institute.

[10] Achterberg, T., 2009. SCIP: Solving constraint integer programs. Mathematical Programming Computation, 1(1):1–41.

[11] Bossel, H., 2007. Systems and Models: Complexity, Dynamics, Evolution, Sustainability. Books on Demand GmbH, Norderstedt.

[12] Nannicini, G., Belotti, P., 2012. Rounding-based heuristics for nonconvex MINLPs. Mathematical Programming Computation, 4:1–31.

[13] Bonami, P., Cornuéjols, G., Lodi, A., Margot, F., 2009. A feasibility pump for mixed integer nonlinear programs. Mathematical Programming, 119:331–352.

[14] Berthold, T., Gleixner, A., 2013. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. Mathematical Programming.

[15] Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A., 2012. Mixed-Integer Nonlinear Optimization. Technical Report ANL/MCS-P3060-1112, Argonne National Laboratory, Argonne, Illinois.

[16] Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A., 2009. Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software, 24(4-5):597–634.

[17] Burer, S., Letchford, A. N., 2012. Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science, 17(2):97–106.

[18] Fügenschuh, A., Martin, A., 2005. Computational Integer Programming and Cutting Planes. In R. W. K. Aardal, G. Nemhauser, editor, Handbook on Discrete Optimization, Series Handbooks in Operations Research and Management Science, volume 12. Elsevier, 69 – 122.

[19] Savelsbergh, M., 1994. Preprocessing and Probing Techniques for Mixed Integer Programming Problems. INFORMS Journal on Computing, 6:445–454.

[20] Berthold, T., Heinz, S., Vigerske, S., 2012. Extending a CIP Framework to Solve MIQCPs. In J. Lee, S. Leyffer, editors, Mixed Integer Nonlinear Programming, volume 154, part 6 of *The IMA Volumes in Mathematics and its Applications*. Springer Verlag, Berlin, 427–444.

[21] Bossel, H., 2007. System Zoo 3 Simulation Models: Economy, Society, Development. Books on Demand GmbH, Norderstedt.