Simulation Modelling Practice and Theory 46 (2014) 164–175

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Derivative-based hybrid heuristics for continuous-time simulation optimization

Atiyah Elsheikh

Austrian Institute of Technology, Vienna, Austria

ARTICLE INFO

Article history:
Received 31 January 2013
Received in revised form 11 November 2013
Accepted 25 November 2013
Available online 28 December 2013

Keywords:
Hybrid heuristics
Derivative-based optimization methods
Differential algebraic equations
Modeling languages
Modelica

ABSTRACT

The topic of simulation–optimization has not been fundamentally tackled by many continuous-time modeling and simulation tools, yet. Common simulation-based optimization problems are usually coupled with standard optimization algorithms like any other simulation-free nonlinear optimization problems. While such couplings are usually based on many state-of-the-art software engineering concepts with a high-level user interface for flexible incorporation of simulation and optimization, the design of specialized optimization strategies targeting simulation-based objective functions is lacked within many simulation–optimization tools. In this work, new redefinition of Non Linear Programming (NLP) problems in the context of continuous-time simulation optimization is presented. Then, the modified optimization problems are efficiently tackled using derivative-based hybrid heuristics. In order to specify, illustrate and implement such heuristics, a new terminology is proposed. According to the proposed terminology, derivative-based hybrid strategies are implemented by hybridizing naive multistart derivative-based optimization methods with population-based metaheuristics. It is shown that the adoption of derivative-based optimization methods within hybrid optimization strategies significantly improves the solution quality of continuous-time simulation optimization problems.

1. Introduction

1.1. Simulation–optimization of object-oriented physical models

Modern equation-based universal modeling languages like Modelica [15] and gPROMS [23] have become widely used in many everyday life continuous-time simulation activities. They are best suited for modern industrial applications requiring the interactions of many physical domains, an essential requirement supported by such languages. Their descriptive power capabilities enable flexible physical modeling with equation-based syntax, hierarchical model design and components reuse. The underlying acausal modeling concepts on which they rely allow for rapid prototyping of complex systems based on differential algebraic equations (DAEs) [14]. Due to avoidance potentials of input–output relations among system components, such systems can be easily assembled in a way similar to the conceptual reality. Additionally, typical simulation environments for such languages are capable of transforming high-level model specification into efficient simulation code. Symbolic model simplifications, DAE Index reduction, analytical computation of Jacobians and linking with advanced DAE solvers are standardly implemented within these environments. In this way, the modeler needs only to perform the physical part of the modeling without focusing on tiny mathematical details.

E-mail address: Atiyah.Elsheikh@ait.ac.at

1569-190X/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2013.11.011

Optimization provides an essential tools package for simulation models[1] with many useful applications like model calibration, parameter identification, optimal control and optimal design among many others. Always more simulation tools provide optimization capabilities some of which are provided by default. For example, simulations of SimScape [1] can be directly accessed from the Matlab workspace and get linked with existing optimization toolboxes. Similarly, Wolfram SystemModeler [2] can be directly linked with the Mathematica environment. Another approach is to provide additional complementary tool for Optimization. For instance, the OpenModelica simulation environment [21] provides also an additional tool OMOptim [45] for optimization with genetic algorithms. A significantly distinguished approach is highlighted by JModelica [4] which enhances Modelica with additional language constructs, Optimica [3], for flexible prototyping simulation optimization problems. The JModelica python-based scripting environment utilizes fundamental derivative-based optimization algorithms implemented with IPOPT [47] as well as multiple-shooting algorithms [39] and genetic algorithms for solving the prototyped optimization problems. Further interesting simulation–optimization tools that provides low-level and high-level developer-oriented capabilities for enclosing arbitrary simulation tools and performing rapid prototyping and design of optimization algorithms are HeuristicLab [26], GenOpt [49] and ParaddisEO [8].

1.2. Overcoming the gaps in current state of simulation–optimization

Despite of these considerable efforts provided by optimization environments for simulation tools, the progress in continuous-time simulation optimization is not yet proportional to the success achieved by simulation tools for physical modeling of dynamical systems. This could be raised to the following factors:

1. The main focus of many simulation tools remains on maintaining and developing the always growing underlying complex language specification while optimization is provided as a secondary luxury service.
2. Linking large-scale simulations with standard optimization algorithms does not usually lead to satisfactory results.
3. The serious enormous difficulties behind realistic large-scale nonlinear simulation optimization problems.
4. The lack of considering the simulation aspect in optimization:

(a) Simulation-based objective functions are considered as just normal objective functions evaluated by simulations.
(b) No efforts are done for developing new optimization strategies especially designed and best suited for simulations instead of just employing standard optimization algorithms.

In this work, many gaps between the current-state of “optimization of simulations” and the desired state of the art of “simulation–optimization” [22,6] are bridged. This is mainly realized according to the following steps:

1. Establishing a redefinition of NLP in the context of large-scale nonlinear dynamics taking the simulation background into account.
2. Considering the equivalent optimization problem of finding optimal start values with which derivative-based strategies converge to local optima of high-quality.
3. Developing a fundamentally new operator-based terminology allowing for straightforward specification and implementation of derivative-based hybrid strategies for the redefined optimization problem.

The proposed heuristics extend hybridization mechanisms presented in [43] in the following aspects:

1. The same techniques exclusively applied to combinatorial optimization problems with discrete search spaces are examined with NLP of non-convex continuous search space.
2. Non-metaheuristics, i.e. derivative-based methods, are considered for hybridization.

The effectiveness of these strategies are examined on a model calibration test problem of a considerable size. It is shown that employing derivative-based hybrid strategies significantly improves the quality of the results in comparison with employing metaheuristics and naive derivative-based multistart strategies alone.

The rest of the paper is structured as follows: Section 2 presents well-known and common problems of continuous-time optimization problems. Section 3 introduces the optimization problem of identifying semi-optimal start values for derivative-based methods. Accordingly, Section 4 provides specialized terminology for developing the hybrid heuristics targeting the redefined optimization problem. These heuristics are to be evaluated along a benchmark in Section 5. Finally, conclusions and ongoing works are addressed in Section 6.

[1] From now on, simulation means continuous-time based simulations.