

Marcus Baum · Gunther Brenner
Jens Grabowski · Thomas Hanschke
Stefan Hartmann · Anita Schöbel (Eds.)

Communications in Computer and Information Science

889

Simulation Science

First International Workshop, SimScience 2017
Göttingen, Germany, April 27–28, 2017
Revised Selected Papers

 Springer

Clausthal-Göttingen International Workshop on
Simulation Science
27 - 28 April 2017, Göttingen, Germany

Communications in Computer and Information Science

889

Commenced Publication in 2007

Founding and Former Series Editors:

Phoebe Chen, Alfredo Cuzzocrea, Xiaoyong Du, Orhun Kara, Ting Liu,
Dominik Ślęzak, and Xiaokang Yang

Editorial Board

Simone Diniz Junqueira Barbosa

*Pontifical Catholic University of Rio de Janeiro (PUC-Rio),
Rio de Janeiro, Brazil*

Joaquim Filipe

Polytechnic Institute of Setúbal, Setúbal, Portugal

Igor Kotenko

*St. Petersburg Institute for Informatics and Automation of the Russian
Academy of Sciences, St. Petersburg, Russia*

Krishna M. Sivalingam

Indian Institute of Technology Madras, Chennai, India

Takashi Washio

Osaka University, Osaka, Japan

Junsong Yuan

University at Buffalo, The State University of New York, Buffalo, USA

Lizhu Zhou

Tsinghua University, Beijing, China

More information about this series at <http://www.springer.com/series/7899>

Marcus Baum · Gunther Brenner
Jens Grabowski · Thomas Hanschke
Stefan Hartmann · Anita Schöbel (Eds.)

Simulation Science

First International Workshop, SimScience 2017
Göttingen, Germany, April 27–28, 2017
Revised Selected Papers

Editors

Marcus Baum
Institute of Computer Science
University of Göttingen
Göttingen, Lower Saxony
Germany

Gunther Brenner
Institute of Applied Mechanics
TU Clausthal
Clausthal-Zellerfeld, Lower Saxony
Germany

Jens Grabowski
Institute of Computer Science
University of Göttingen
Göttingen, Lower Saxony
Germany

Thomas Hanschke
Institute of Applied Stochastics
and Operations Research
TU Clausthal
Clausthal-Zellerfeld, Lower Saxony
Germany

Stefan Hartmann
Institute of Applied Mechanics
TU Clausthal
Clausthal-Zellerfeld, Lower Saxony
Germany

Anita Schöbel
Institute for Numerical and Applied
Mathematics
University of Göttingen
Göttingen, Lower Saxony
Germany

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-319-96270-2 ISBN 978-3-319-96271-9 (eBook)
<https://doi.org/10.1007/978-3-319-96271-9>

Library of Congress Control Number: 2018950461

© Springer Nature Switzerland AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Welcome to the proceedings of the first Clausthal–Göttingen International Workshop on Simulation Science, which took place in Göttingen, Germany, during April 27–28, 2017.

Owing to the rapid development of information and communication technology, the understanding of phenomena in areas such as natural sciences and engineering increasingly relies on computer simulations. Traditionally, simulation-based analysis and engineering techniques are a research focus of both TU Clausthal and the University of Göttingen, which is also reflected in their interdisciplinary joint research center “Simulation Science Center Clausthal–Göttingen.” In this context, the first Clausthal–Göttingen International Workshop on Simulation Science brought together researchers and practitioners in order to report on the latest advances in simulation science. In particular, the workshop concentrated on (a) simulation and optimization in networks, (b) simulation of materials, and (c) distributed simulations.

The Convention Centre by the Observatory in Göttingen served as the workshop venue. It is an outbuilding of the Historical Observatory where the famous scholar Carl Friedrich Gauss used to work and live. The welcome address of the workshop was given by Prof. Norbert Lossau (Vice-President of the University of Göttingen) and Prof. Thomas Hanschke (President of the TU Clausthal). Recent results and an outlook to future developments in simulation science were discussed in three plenary talks given by Achim Streit (Karlsruhe Institute of Technology), Samuel Forest (MINES Paristech), and Kai Nagel (TU Berlin). The social program included a guided city tour through Göttingen’s historical old town and a workshop dinner that took place in the basement of the city hall – the “Ratskeller” of Göttingen.

In total out of 40 submitted extended abstracts 39 were accepted for presentation at the workshop. After the workshop, 16 full-length papers of a subset of submissions have been accepted in a second review round for the post-proceedings.

We are very grateful to everyone who supported the workshop. In particular, we would like to thank the Technical Program Committee, the local arrangements co-chairs, Annette Kadziora and Fabian Sigges, and the finance chair, Alexander Herzog. The registration process was organized by VDE conference services and we highly appreciate the co-sponsoring by the Gesellschaft für Operations Research e.V. (GOR) and the Arbeitsgemeinschaft Simulation (ASIM).

After the success of this workshop, we look forward to the second Clausthal–Göttingen International Workshop on Simulation Science, which will take place in May 2019 in Clausthal, Germany.

April 2018

Marcus Baum
Gunther Brenner
Jens Grabowski
Thomas Hanschke
Stefan Hartmann
Anita Schöbel

Organization

Workshop General Chair

Marcus Baum University of Göttingen, Germany

Workshop Co-chairs

Gunther Brenner TU Clausthal, Germany
Jens Grabowski University of Göttingen, Germany
Thomas Hanschke TU Clausthal, Germany
Stefan Hartmann TU Clausthal, Germany
Anita Schöbel University of Göttingen, Germany

Technical Program Committee

Valentina Cacchiani University of Bologna, Italy
Stefan Diebels Saarland University, Germany
Jürgen Dix TU Clausthal, Germany
Umut Durak DLR, Institute of Flight Systems, Germany
Felix Fritzen University of Stuttgart, Germany
Igor Gilitschenski ETH Zürich, Switzerland
Marc Goerigk Lancaster University, UK
Marco Huber USU AG, Karlsruhe, Germany
Tobias Kretz PTV Group, Karlsruhe, Germany
Allan Larsen Technical University of Denmark, Denmark
Ming Li Nanjing University, China
Laura De Lorenzis TU Braunschweig, Germany
Kai Nagel TU Berlin, Germany
Helmut Neukirchen University of Iceland, Iceland
Bernhard Neumair Karlsruhe Institute of Technology, Germany
Natalia Rezanova Danish State Railways, Denmark
Ulrich Rieder Ulm University, Germany
Rüdiger Schwarze TU Freiberg, Germany
Marie Schmidt Erasmus University Rotterdam, The Netherlands
Thomas Spengler TU Braunschweig, Germany
Ulrich Tallarek Philipps-Universität Marburg, Germany
Pieter
 Vansteenwegen KU Leuven, Belgium
Sigrid Wenzel University of Kassel, Germany
Peter Wriggers University of Hanover, Germany
Ramin Yahyapour GWDG, Germany
Martin Zsifkovits Bundeswehr University Munich, Germany

Finance Chair

Alexander Herzog TU Clausthal, Germany

Local Arrangements Co-chairs

Annette Kadziora University of Göttingen, Germany

Fabian Siggas University of Göttingen, Germany

Organized by



Co-sponsored by

Gesellschaft für Operations Research e.V.



In Cooperation with

ASIM - Arbeitsgemeinschaft Simulation



Information Technology Society in the VDE



Contents

Simulation and Optimization in Networks

| | |
|--|----|
| Passenger-Induced Delay Propagation: Agent-Based Simulation of Passengers in Rail Networks | 3 |
| <i>Sebastian Albert, Philipp Kraus, Jörg P. Müller, and Anita Schöbel</i> | |
| Impacts of Vehicle Sharing with Driverless Cars on Urban Transport | 24 |
| <i>Markus Friedrich, Maximilian Hartl, and Christoph Magg</i> | |
| Combining Simulation and Optimization for Extended Double Row Facility Layout Problems in Factory Planning | 39 |
| <i>Uwe Bracht, Mirko Dahlbeck, Anja Fischer, and Thomas Krüger</i> | |
| Interactive Multiobjective Robust Optimization with NIMBUS | 60 |
| <i>Yue Zhou-Kangas, Kaisa Miettinen, and Karthik Sindhya</i> | |
| Heuristics and Simulation for Water Tank Optimization | 77 |
| <i>Corinna Hallmann, Sascha Burmeister, Michaela Wissing, and Leena Suhl</i> | |

Simulation of Materials

| | |
|--|-----|
| Accelerated Simulation of Sphere Packings Using Parallel Hardware | 97 |
| <i>Zhixing Yang, Feng Gu, Thorsten Grosch, and Michael Kolonko</i> | |
| MC/MD Coupling for Scale Bridging Simulations of Solute Segregation in Solids: An Application Study | 112 |
| <i>Hariprasath Ganesan, Christoph Begau, and Godehard Sutmann</i> | |
| 3D Microstructure Modeling and Simulation of Materials in Lithium-ion Battery Cells | 128 |
| <i>Julian Feinauer, Daniel Westhoff, Klaus Kuchler, and Volker Schmidt</i> | |
| On Microstructure-Property Relationships Derived by Virtual Materials Testing with an Emphasis on Effective Conductivity | 145 |
| <i>Matthias Neumann, Orkun Furat, Dzmitry Hlushkou, Ulrich Tallarek, Lorenz Holzer, and Volker Schmidt</i> | |

Distributed Simulations

Simulating Software Refactorings Based on Graph Transformations 161
*Daniel Honsel, Niklas Fiekas, Verena Herbold, Marlon Welter,
Tobias Ahlbrecht, Stephan Waack, Jürgen Dix, and Jens Grabowski*

Transparent Model-Driven Provisioning of Computing Resources
for Numerically Intensive Simulations 176
*Fabian Korte, Alexander Bufe, Christian Köhler, Gunther Brenner,
Jens Grabowski, and Philipp Wieder*

Extending the CMMI Engineering Process Areas for Simulation
Systems Engineering 193
*Somaye Mahmoodi, Umut Durak, Torsten Gerlach, Sven Hartmann,
and Andrea D’Ambrogio*

Learning State Mappings in Multi-Level-Simulation 208
Stefan Wittek and Andreas Rausch

Unifying Radio-in-the-Loop Channel Emulation and Network Protocol
Simulation to Improve Wireless Sensor Network Evaluation 219
Sebastian Böhm and Michael Kirsche

Assessing Simulated Software Graphs Using Conditional Random Fields 239
*Marlon Welter, Daniel Honsel, Verena Herbold, Andre Staedtler,
Jens Grabowski, and Stephan Waack*

Elephant Against Goliath: Performance of Big Data Versus
High-Performance Computing DBSCAN Clustering Implementations 251
Helmut Neukirchen

Author Index 273

Simulation and Optimization in Networks



Passenger-Induced Delay Propagation: Agent-Based Simulation of Passengers in Rail Networks

Sebastian Albert¹, Philipp Kraus², Jörg P. Müller², and Anita Schöbel¹ (✉)

¹ Georg-August-Universität Göttingen, Göttingen, Germany
{albert,schoebel}@math.uni-goettingen.de

² Technische Universität Clausthal, Clausthal-Zellerfeld, Germany
{philipp.kraus,joerg.mueller}@tu-clausthal.de

Abstract. Current work on delay management in railway networks has – to the best of our knowledge – largely ignored the impact of passengers’ behavior on train delays. This paper describes ongoing work aiming to explore this topic. We propose a hybrid agent-based architecture combining a macroscopic railway network simulation with a microscopic simulation of passengers in stations based on the *LightJason* agent platform. Using an initial instantiation of the architecture, we model a simple platform changing scenario and explore how departure delays of trains are influenced by delays of incoming trains, and by numbers and heterogeneity of passengers. Our results support the hypothesis that passengers’ behavior in fact has a significant effect on delays of departing trains, i.e., that passengers’ behavior in stations must not be neglected. We recommend to include these effects in up-to-date models of delay management.

1 Introduction

Delays are a fact in most railway systems. Triggered by one or several source events (a track is closed, a signal fails, a train departs late because a large group is boarding) they may spread through large parts of the railway network. Many mechanisms of such a delay propagation are well understood: A train which departs with some delay also arrives with some (maybe smaller) delay; but delays can also propagate from one train to another if a punctual train waits for a delayed feeder train (*wait-depart decision*), or if a punctual train has to slow down because its track is occupied by a delayed train ahead of it (*priority decision*). It is also known that delays may propagate due to vehicle and drivers’ schedules. In order to keep the delays small, *delay management decisions*

Partially supported by Simulation Science Center Clausthal/Göttingen (SWZ), project ASIMOV.

are made at railway traffic control centers. Optimizing these from a passenger-oriented point of view is an ongoing topic of research, see [9] for a recent survey on delays in railway systems and delay management.

However, work about delay propagation and delay management ignores the following two important issues. First, it is mostly neglected that the route a passenger¹ would take depends on the actual delays and on the delay management decisions. In many cases, waiting for the next train of the same service after missing a connection is inefficient for passengers, because a different combination of train services may result in earlier arrival at their destinations. Only very few approaches take this into account [8, 22]. Note that the delay of a train may even result in new opportunities for connections that do not exist in regular, undisturbed operations.

Another neglected aspect is related to the behavior of passengers at the stations: What do they do if a transfer is likely to be missed? People running from one platform to another in a hurry can interfere with others, heavy luggage may slow down passengers and increase the time they need for changing trains, and crowds in the station also slow down traffic. Particular patterns of passenger flow can even cause additional train delays when, for instance, a steady trickle of people entering a train prevents the doors from closing.

The following scenario illustrates this effect: Suppose a large number of passengers alight from an incoming train A in a station. There are only a few minutes for changing to the platform of a connecting train, B. In such a situation, it happens often enough that one (fast) passenger reaches train B on time, and before the doors can close, the next passenger arrives, then the passenger after, and so on. This might lead to a delay of train B, even if B was punctual so far. To the best of our knowledge, effects as this one have not been considered in delay management yet.

In this paper we simulate not only trains in railway networks and delays propagating between them due to priority and wait-depart decisions, but we also simulate the passengers and which effects their behavior has on delays. This includes their route choices in the railway network as well as their movements through the stations. Hence, not only do we study the influence of delays on passengers but also the influence of passengers on delays. The resulting simulation model can be used to predict delays more realistically in every specific situation. This is useful for several reasons. First, being able to predict delays more precisely helps when informing passengers about the options they have. Second, our simulation can be used to evaluate particular delay management decisions (e.g.: train A should wait for train B today) or even more general delay management strategies and hence help to reduce follow-up delays in railway systems.

Closest to the topic we study is the simulation of crowd congestion at interchange stations which has been studied for a station in Toronto in [25]. In [26] these effects are included in a crowd dynamics and transit network simulation

¹ For reasons of simplicity, throughout this paper we will uniformly use the term *passenger* to refer to passengers on a train, but also to travelers at a railway station (including pedestrian through-traffic).

platform which switches between different simulators. Delay propagation in the context of railway networks has been studied in [15, 16] and simulated in [19]. It has also been used in delay management, see [9] and references therein. In these papers, wait-depart decisions to maintain connections for transferring passengers are considered for delay propagation, but the route choice of passengers and their behavior in stations is neglected. In [17], Li and Zhu propose a model considering passenger choice behaviors that take train delays into account. They study in simulation how this can be factored into a passenger flow distribution calculation. A discrete event simulation mechanism is used to evaluate their model. They consider train delays and platform delays as sources for delays. However, they do not provide a detailed microscopic model of passengers at stations, but use a mesoscopic probabilistic approach. In [27], Wales and Marinov report the results of a case study of a real metropolitan rail network, analyzing the impact, frequency and scope of delays, and attempting to mitigate them. They employ a mesoscopic discrete event-based railway network simulation, based on empirical data while detailed modeling of travelers and platform delays are not considered.

There are many papers and tools for a microscopic simulation of physical train movements along the tracks in a railway network; we refer to [20] for an overview. However, passengers are neglected in these simulations. The work presented by Zhang et al. in [29] does describe a rich cellular automata-based alighting and boarding micro-simulation model for passengers in Beijing metro stations and the effects of different group sizes on the alighting and boarding performances. However, they do not study the integrated modeling of railway station and railway network, and the cellular automata approach makes it difficult to model more advanced cognitive behavior such as planning.

Thus, to the best of our knowledge, there is no work that aims at studying the impact of passengers' behavior on train delays by considering both the railway networks and railway stations in microscopic models. Therefore, we propose an agent-based modeling and simulation approach using the *LightJason* framework (see Sect. 3 for related work and details), since it provides a flexible, scalable architecture and can link microscopic and macroscopic elements.

The remainder of this paper is structured as follows. In Sect. 2 we describe the simulation model combining a macroscopic railway network model with a microscopic model of the passengers at the stations. These two worlds get connected when passengers board or alight: train doors and platforms serve as interfaces. Section 3 is devoted to our realization of the agent-based simulation, and Sect. 4 provides first simulation results showing that passengers' behavior in fact has a significant effect on delays of the departing trains. The paper is ended with some conclusions and a plan for further research.

2 Simulation Model

2.1 Overview

Delay management problems have been traditionally modeled from the railway network perspective using macroscopic networks, and formulated and solved as

integer programs (IPs). Considering pedestrians in a more realistic fashion both in the train network and in the stations imposes additional requirements on modeling. E.g., it is necessary to link a graph-based model for the network with a grid-based model for the station. Also, while a macroscopic flow simulation can be applied for the network, microscopic pedestrian flow models are required to capture realistic fine grained movement patterns in stations according to the pedestrians’ information states, preferences, and plans. Essentially, our model needs to support the “network world” and the “station world” as well as passengers moving across these worlds when entering or leaving trains.

There are different architectural approaches for combining a macroscopic railway network simulation with a microscopic station simulation model. Firstly, a co-simulation approach could be considered (e.g. [5]), linking two separately run simulations. Secondly, an existing simulation system could be extended by new models capturing e.g. the station/pedestrian part, or the network part. Third, a new simulation framework could be created based on a unified model. For scalability and to reduce the integration and maintenance effort, we chose the third alternative, aiming at a unifying agent-based model based on the agent-based platform *LightJason* (see Sect. 3). An important design choice is that trains, passengers, and dispatcher(s) are modeled as Belief-Desire-Intention (BDI) agents. Trains drive, and open or close doors; passengers travel from start to destination, move through stations and board/alight trains. The dispatcher decides which trains wait. This enables a uniform view on all active simulation entities while maintaining different levels of detail.

Section 2.2 describes in more detail how the railway network is modeled. Our approach for describing railway stations and passenger behavior within railway stations is presented in Sect. 2.3. Section 2.4 explains how our initial model handles the transition between the railway network and station “worlds”.

2.2 Railway Network Submodel

The macroscopic simulation of trains is based on the so-called *event-activity network* (EAN) $\mathcal{N} = (\mathcal{E}, \mathcal{A})$. The vertices of the network are *arrival* and *departure events* \mathcal{E}_{arr} and \mathcal{E}_{dep} where both consist of a train and a station. The events are linked by the following *activities*:

- A *driving activity* $a \in \mathcal{A}_{drive}$ links a departure event at a station with an arrival event of the same train at its next station. It represents a train driving between the two consecutive stations.
- A *waiting activity* $a \in \mathcal{A}_{wait}$ links the arrival event of a train with its departure event at the same station and corresponds to the time period in which a train is waiting in a station to let passengers alight and board.
- A *changing activity* $a \in \mathcal{A}_{change}$ links an arrival event of a train at a station with the departure event of another train at the same station. It corresponds to the transfer of passengers from one train to another by foot within a station.
- Finally, a *headway activity* $a \in \mathcal{A}_{head}$ models the limited capacity of the track system. This can either be two trains driving on the same track into the same

direction or two trains driving into opposite directions on a single-way track. The duration $L_{(i,j)}$ of a headway activity (i, j) means that the departure j must take place at least $L_{(i,j)}$ minutes after the departure i (if j actually takes place after i). We refer to [24] for details. Headway activities are used to prevent that no two trains occupy the same platform at the same time. Note that not all conflicts on the tracks can be prevented by using headway constraints in a macroscopic model.

There exist simulations (see [20] for an overview) which are able to route trains on the track system respecting all signals, speed limits and other safety measures, including interlocking effects of multiple trains' routes. However, since the focus in this simulation is to analyze the influence of the behavior of the passengers we neglected the details of the physical railway network in this first version and used the macroscopic event-activity network for simulating the railway world. A small example of an EAN with three trains is depicted in Fig. 1.

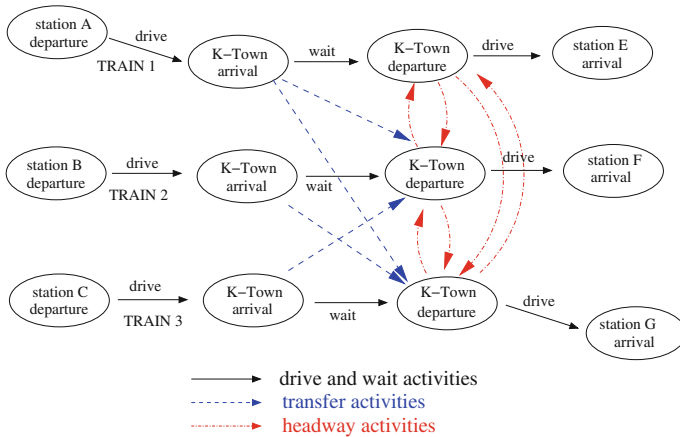


Fig. 1. A small event-activity network consisting of three trains which meet at the station K-town.

Event-activity networks are suitable for modeling trains, passengers, and delays: The trains are visible directly; passengers can be routed through an EAN (where they may not use headway activities) allowing to change between trains along the changing activities; and delays can be propagated through the network along driving, waiting, changing and along headway activities if the buffer times of each of these activities is known.

In our simulation model, every train in the EAN is represented as one simulation entity, or agent. The *train agents* technically know their timetable as a list of stations together with their arrival and departure times. Train agents are aware of the current time and must not depart before the respective published departure time. Their arrival at the next station follows after the amount of time

determined by the distance and their running speed, plus any additional delay that can be modeled between the two stops. A detailed simulation on the tracks is currently neglected since our focus is to study passengers at stations. These are also modeled as agents. In the macroscopic railway network they move along the network in their trains. In order to reach their destinations, each *passenger agent* knows their own itinerary as a list of train rides, each denoting the train number, the station, and the departure time where they have to board or alight. Some of them are able to adapt their itineraries when their planned journey is affected by a delay. These passenger agents represent informed passengers that use their smart-phones to optimize their journeys. Other passengers will stick to their itineraries, or follow the guidance given by the staff or an information board. All passengers can update their itineraries when they miss a transfer.

2.3 Railway Station Submodel

Following a microscopic agent-based approach, this submodel describes the physical environment of a railway station, the travel demand, and the behavior (flow) of passengers in the station. Figure 2 shows the simple example scenario used throughout this paper. It features two opposing entrances and platforms, an info sign containing information about track plans and departure times for trains, as well as points of interests such as a restroom or a store. From the central hall, there is an entrance/exit to/from each platform. The two entrances and exits of the station hall can be used to simulate different levels of pedestrian through-traffic and their impact on delays. To model travel demand, we use origin-destination (O/D) matrices. Based on the O/D information, passengers are generated with an itinerary and released at the entrances of the initial station of their journey. Note that for the scope of this paper, the itinerary is a given input, and we do not consider rerouting. In the future, we will investigate the case that passengers first need to obtain it by actually moving near the info sign, or that they may use mobile devices.

Based on its itinerary and departure information, a passenger decides where to go (e.g., to catch a train or to leave the station). It then plans a route and moves towards its destination. While moving, the passenger can decide to opportunistically interrupt or modify the planned route towards points of interest, e.g., to eat something (represented as dynamic internal drive to increase its energy level) or to get a newspaper (represented by a level of preference or interest). Furthermore, its trajectory is influenced by the asynchronous movement of other passengers which can lead to collisions.

We model the flow behavior of passengers using a cellular automaton approach [6] based on a grid representation of the environment. A cell can be empty or not, and it can be of different types (floor, info sign, Point of Interest (PoI), ...). Based on the cell structure, discrete goal- and event-based action rules following the BDI model are employed to describe the interaction between agents and the environment.

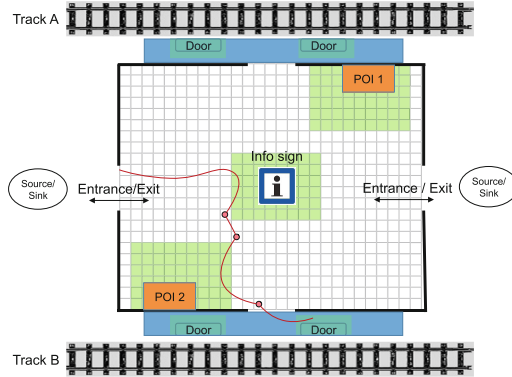


Fig. 2. Exemplary station environment

To model flexible, realistic movement and routing, we use a hybrid control architecture to integrate routing and movement control as proposed in [14]:

Routing: The Jump Point SearchPlus (JPS+) routing algorithm calculates a list of landmarks (filled dots in Fig. 2) from starting to goal position. We extend JPS+ by preprocessing to define suitable initial landmarks. During runtime, JPS+ can be executed for any agent individually, to choose landmarks that reflect individual preferences of the agent.

Movement control: A passenger’s actual movement trajectory (indicated by the line starting from the entrance in Fig. 2) depends on its planned route, but also on the current situation including other passengers. To this end, we use a Social Force Model approach [13], and added a simple reactive collision detection².

In the following, we highlight two aspects related to our agent-based simulation model: The first is our approach to solve the problem of modeling **perception** in a scalable fashion. Second, we illustrate the above-mentioned BDI modeling of passenger behavior.

Dynamic Perception Algorithm. In every simulation step, all passenger agents in the station need to update their perception, i.e., access the environmental state. This is constrained by limited perception ranges which need to be taken into account. We address this by enclosing the relevant simulation entity by a bounding box of configurable size. This also allows a fine-grained definition of information exchange between objects. For example, the info sign box in Fig. 2 can detect a passenger within its (the info sign’s) bounding box. Listing 2.1 shows our algorithm, which processes the movement trajectory of a passenger by the Liang-Barsky line-clipping algorithm [18], which we found to produce good results for detection and length calculation within the bounding box.

² see scenario in [2] or <https://lightjason.github.io/news/2017-02-video/>.

Listing 2.1. Detection algorithm

```

1  input: agent PASSENGER, agent BOX
2  begin
3      LINE ← liangbarsky( PASSENGER.startposition . PASSENGER.
        endposition , BOX.topleft , BOX.bottomright )
4      LENGTH ← || LINE ||2
5      if LENGTH = 0
6          return
7      if not BOX.contains( PASSENGER )
8          PASSENGER.trigger( "enter" , BOX )
10     BOX.objects.add( PASSENGER );
11     BOX.trigger( "moving" , PASSENGER )
12 end

```

In each simulation step, for each agent in the simulation, the agent cycle is executed. This means that for each agent, information is updated, goals/events checked, and plans expanded by, e.g., creating and instantiating the actions for moving through the station while following the landmarks. In the example of the info sign, its bounding box will determine lists of passengers that entered, left, or moved through it within an agent cycle. The agent which belongs to the bounding box checks if another agent has left the box. Based on this information, the box state is updated. In particular, passengers who left the bounding box are informed about this by a *leave* message, which can trigger new plans in the passenger agent.

Pedestrian Movement. As mentioned above, the routing algorithm JPS+ calculates landmarks for passengers moving through the station, while basic collision detection/avoidance is factored into the environment. The detailed movement behavior of passengers in the station is expressed through a set of BDI rules, which are modeled in a rule-based scripting language (see Sect. 3 for details). Listing 2.2 illustrates basic elements of an exemplary BDI program consisting of one initial piece of information (= belief) and one plan.

Listing 2.2. Example BDI program fragment for passenger agent

```

1  train( id("ICE 751" ) , platform(3) ) .
3  +!catchtrain
4  : >>( train( id(T) , platform(P) )
5  <-
6  route/set(P);
7  !movement/walk/forward .

```

In this example, we assume that the agent has the initial information that its train ICE 751 will depart from platform 3. This information is encoded in a belief (line 1). Line 3ff shows a plan. Plans are triggered by events, such as a new belief ('+'), retracted belief ('-'), new goal ('!') or retracted goal ('!'),

(line 3). In the example, the plan is triggered by a goal *catchtrain*. The applicability of a plan can be further subject to additional context conditions; e.g., in line 4 it is required that the agent has the information that its train has arrived on a platform. The conditions are evaluated and variables bound using logical unification (indicated by the ‘ \gg ’ operator). Event and condition together form the *antecedent* of the plan rule; its *consequent* is the plan body, which consists of calculations, the execution of actions (e.g., the calculation of a route (line 6)), and the creation of new subgoals (line 7).

Section 3 elaborates on the principles for the execution of agent programs. A richer code example describing passenger moving behavior is displayed in the Appendix to this paper.

2.4 Transition Between Submodels

As specified so far, passengers switch between the two submodels of the simulation. Technically, a seamless transition between the two submodels is accomplished by modeling passengers as agents with unique identities over their complete life-cycle, and by a common object model (cf. Sect. 3). The interface through which passengers switch from the railway network to the station is modeled by the doors of the trains and the platforms.

Every station is made up of a set of uniquely named *platforms*. Each platform technically maintains a collection of passengers currently standing on or moving across it, as well as a reference to any train dwelling at that platform. Each passenger maintains its itinerary as a list of train rides, stations, departure platforms and times. In order to simulate the effects of passengers alighting and boarding, every train has a collection of *doors*. A train can only depart from a platform at a station when all its doors are closed and locked. Once the departure time is reached, a command is sent to the doors to close and lock. Only when all doors are locked will the train start driving. Technically, it will then inform the platform about its departure. Analogously, when arriving at a station, it will inform the arrival platform and the passengers inside itself, and unlock the doors.

Doors also maintain information about their current state, including two queues of passengers for alighting or boarding, respectively. In a first approximation, we assume that a door can only be used by one passenger at a time, and whenever the door is open and free, it will trigger either the next passenger of the alighting queue, if any, or otherwise the next passenger of the boarding queue, if any. To simulate the effect of the safety light-barriers installed in many trains to prevent doors from closing when there are people in it, the door keeps track how long they have not been used by a passenger. It can only close if it has not been used by a passenger for a pre-specified amount of time, usually a few seconds. The process of closing is also simulated, which can be interrupted by an adamant passenger arriving at the door just then, forcing it to open again.

To conclude, a passenger boarding a train performs the following steps:

1. Upon announcement of a train, compare it to the current itinerary entry whether it is the one to board.

2. If the train is to be boarded, queue at one of the train’s doors for entrance.
3. When the door is open and there are no preceding passengers in the queue, enter the train.
4. After entering (which takes a pre-specified amount of time), release the door. Unregister from the platform and register with the train.

The steps for alighting from a train are analogous to those for boarding.

3 Agent-Based Simulation with *LightJason*

In Sect. 2.1, we have argued the case for using agent-based models for microscopic simulation of railway network and railway station environments. Agent-based modeling and simulation (ABMS) [3] is a computational paradigm in which the concepts of agents and multi-agent systems form the metaphor underlying the simulation model. From the modeling perspective, the concept of a multi-agent system (MAS) allows a reduction of complexity. Concepts such as reactivity, proactiveness, and social ability, as generally attributed to agents [28] are helpful for microscopic behavioral modeling. Agents are active simulation entities, which are defined by a BDI architecture, allowing a fine-grained modeling of their knowledge, behavior, and goal-driven planning/decision-making. The essential domain entities (agents) in our simulation scenario, i.e., passengers, trains, and the dispatcher, have been described in Sect. 2.

In selecting an appropriate simulation platform, there are a number of choices and tradeoffs to consider. The “silver bullet” would doubtlessly be a simulation software that provides rich microscopic domain models (i.e., in our case, railway network *and* railway station) and support rich agent-based models. Also, this simulation software should be open-source in contrast to commercial tools such as MassMotion, SimWalk, or VISSIM/Viswalk), to enable extension and validation not only of the models but also of the underlying platform. Unfortunately, while there are quite a few platforms out there, to our knowledge none of them satisfies all of these requirements. MATSim [12] seems to be close; however, it cannot be used straightforwardly, as it does not support microscopic agent-based flow simulation but rather mesoscopic queuing models. Thus, efficiently coupling railway network simulation with traveler simulation *either* requires the integration of existing systems, linking, e.g., an existing agent platform with an existing railway and/or railway station simulation, *or* a from-scratch implementation based on an ABMS platform.

Our approach has been to choose a suitable ABMS platform and to develop the railway network and railway station modules in that platform, using a clean software architecture that will enable us at a later point in time to replace the domain simulations components by different (academic or commercial) components. Core requirements for a MAS platform to be used are scalability (support a large number of agents, particularly in the network part), and ease of integration with/into other systems. When building an agent-based application, it is necessary to *agentify* simulation entities, e.g. to turn them into software agents and thus allowing the simulation runtime system to execute them in a controlled

fashion. A major obstacle for this to be possible are built-in runtime systems in conjunction with hard-wired inflexible software architecture. From the perspective of suitable runtime systems and frameworks, we refer to [1] for a discussion of the state-of-the-art and requirements. We summarize key requirements for a MAS simulation framework [2]:

1. Simulate a large set of different, heterogeneous agent types
2. Fine-grained parameterization for modeling individual agent behavior
3. Highly asynchronous execution mechanisms
4. Concurrent execution on high performance/cloud platforms
5. High abstraction of software developing to separate domain-specific behavior and coding behavior.
6. Modular, exchangeable run-time component and clean software architecture for ease of integration.

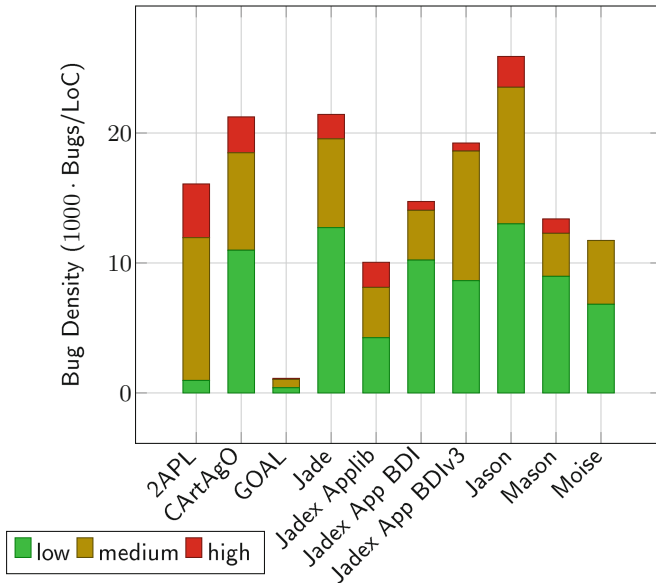


Fig. 3. Bug density analysis for agent-oriented programming frameworks

An analysis of available agent-based modeling/programming frameworks reveals that these requirements are only partly met. Main limitations relate to scalability due to proprietary runtime systems [2]. Also, the platforms often do not support state-of-the-art Object-Oriented methodologies and architectures, because their documentation and code quality is poor; therefore including them in an existing code-base is very difficult. As an example, Fig. 3 shows results of a comparison of the code qualities of popular open-source agent platforms³. It is

³ Evaluation was performed in January 2017 using the tools *FindBugs* and *J-Depend*.

easy to see that almost all platforms bear considerable numbers of errors of high and medium severity. In particular, none of the platforms provides a modular runtime system architecture. This among others led us to develop the framework *LightJason* [2]. *LightJason* is inspired by Jason [4]; however, it builds on a completely new code-base and extends the descriptive language *AgentSpeak(L)* [21], which is used to script agent behavior in Jason, to a newly designed language ASL+⁴.

ASL+ extends the (Java-based) Object-Oriented paradigm with the BDI concept and the execution mechanism known from Procedural Reasoning System (PRS) [11]. Its main features include lambda expressions, multi-plan and -rule definition, multi-variable assignments, concurrent execution mechanisms, and a fuzzy inference concept. For a brief glimpse of the ASL+ language, we refer to the example discussed in Sect. 2.3, Listing 2.2, and to the more complex example shown in the appendix to this paper. Regarding plan execution semantics, the following principles have been designed into *LightJason*:

1. Multiple matching plans are executed in parallel; synchronization can be enforced through setting context conditions.
2. Multiple subgoals created in a plan body are triggered concurrently; instantaneous, sequential execution can be enforced with using the goal prefix `!!+` instead of `!+` in the body of the plan specification.
3. Multiple actions in a plan body are executed sequentially; this default semantics can be changed to parallel execution through a `@Parallel` annotation at the beginning of the plan specification.

The system architecture underlying the *LightJason* framework is based on a layered architecture (see Fig. 4) which combines functional, object-oriented, and logic programming/modeling paradigms. *LightJason* is open-source (see <https://github.com/LightJason>). It supports easy integration with third-party systems and services by incorporating standard interfaces such as Representational State Transfer (REST) or GraphQL by means of built-in actions.

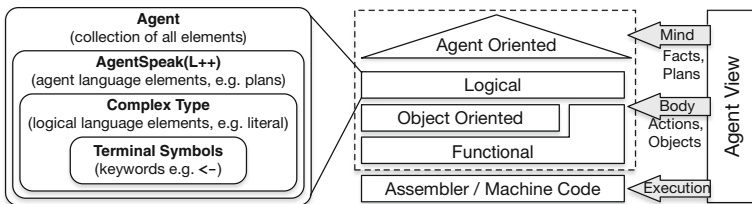


Fig. 4. *LightJason* conceptual architecture

⁴ ASL+ stands for *AgentSpeak(L++)*.

Up-to-date key indicators related to code quality (including the statistics obtained using *FindBugs* and *J-Depend*) can be checked on the github projects page⁵. Initial *LightJason* benchmark results providing evidence for the scalability of the framework are available in the *LightJason* documentation⁶. For more detailed information, we refer to the *LightJason* online reference⁷.

4 Experiments

4.1 Example Scenario

For a first validation of the model described in Sect. 2, we use a small scenario in which two trains *A* and *B* meet at a station. For example, train *A* is scheduled to arrive at 10:00 and train *B* is scheduled to leave at 10:04, i.e., the available time between the planned arrival of train *A* and the planned departure of train *B* is 4 min. We simulate passengers that want to transfer from train *A* to train *B*. The average time for changing from the platform of train *A* to the platform of train *B* is assumed to be 2 min. The situation is depicted in Fig. 5.

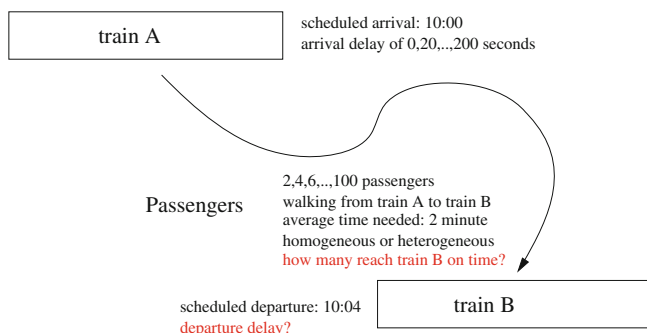


Fig. 5. The simulated scenario: passengers transfer from train *A* to train *B* in a station.

The model parameters which we use to describe a scenario are the following: We mainly investigate the arrival delay of train *A* and the number of passengers who would like to change from train *A* to train *B* in this station. The minimal, the maximal, and the average *transfer time*, i.e., the time a passenger needs between deboarding train *A* and boarding train *B*, are used to model the behavior of the passengers. Increasing the average transfer time has the same effect as increasing the arrival delay of train *A* and is hence not further investigated. Heterogeneous passengers (some with large baggage, some without baggage, elderly people, children, students) are modeled by a larger span between minimal and maximal

⁵ <http://lightjason.github.io/AgentSpeak/project-reports.html>.

⁶ <https://lightjason.github.io/benchmark/>.

⁷ <http://lightjason.org>.

transfer time while homogeneous passengers (e.g. in the morning traffic where most passengers are traveling only with small briefcases, moving through the station with roughly the same speed) are modeled by a small span. We can also change the time, the light-barrier needs before it closes the door. Altogether we simulated nearly 10,000 different parameter combinations, each of them once.

For simplicity we used only one door for each train that is used for all passengers, and we assume that all passengers in train *A* wish to continue their journeys with train *B*; i.e., there are no other passengers that alight from train *A* in our station and no other passengers that board train *B*. As we will see, even this simple situation shows that the passengers' behavior in the stations must not be neglected.

4.2 Case 1: Punctual Arrival of Train *A*

We first consider the case in which train *A* arrives without delay. In this situation one would expect that all passengers can transfer and that train *B* can depart punctual. This is also what classic delay management models would use. However, as Fig. 6 shows, this is not the case if many passengers want to transfer: In our simulation the departure delay is zero if no more than 28 passengers want to transfer but it increases if 30 or more passengers wish to transfer since they need some time to board train *B* one after another. One could argue that this effect may be neglected, as usually not too many passengers transfer to the same train, and they distribute among several doors. However, we see from Sect. 4.3, we can expect significant effects for small numbers of passengers if train *A* arrives with some delay, or if passengers are not homogeneous.

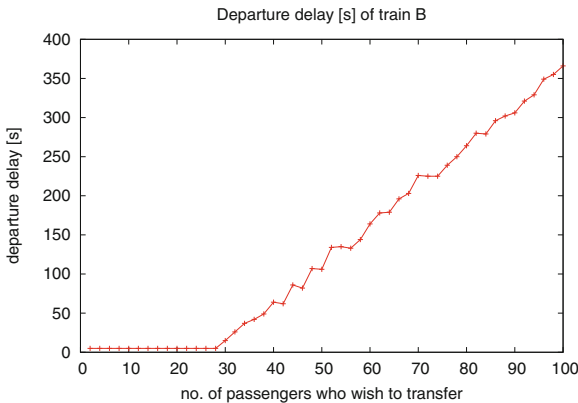


Fig. 6. Departure delay of train *B* with respect to the number of passengers. If more than 30 passengers want to transfer, they delay the departing train.

4.3 Case 2: Delayed Arrival of Train A

We simulate what happens if train A arrives at the station with some delay. Recall that passengers need on average 2 min to walk from the arrival platform of train A to the platform on which train B departs, and that the scheduled time for this transfer is 4 min. The passengers hence have a *transfer buffer time* of 2 min. Consequently, it is common to assume that passengers are able to board train B if the delay of train A is less than 2 min and that in this case all passengers make the transfer and train B leaves on time. For the case that the arrival delay of train A is larger than 2 min, the classic models assume that the transfer cannot be made since train B has already departed before the passengers from train A arrive at its platform. I.e., in both cases it is assumed that train B leaves punctual. Simulating these situations shows that all these common assumptions may be wrong, see Fig. 7 for our results.

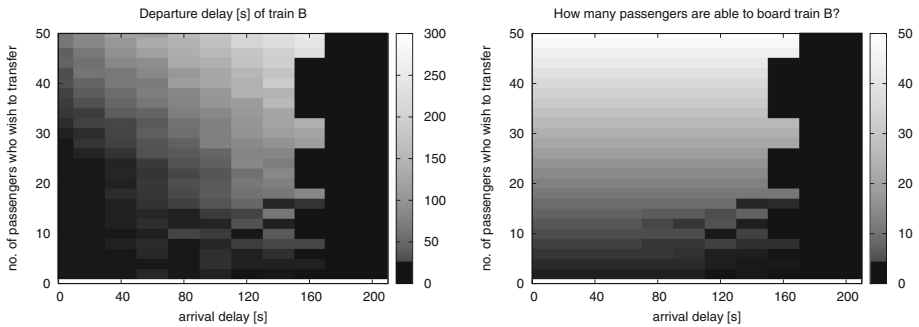


Fig. 7. Departure delay of train B (the lighter the color the more delay) and number of passengers who are able to board train B (the lighter the color the more passengers).

The figure shows our evaluations for different numbers of passengers, and different arrival delays of train A , namely for

- $n \in \{2, 4, 6, \dots, 98, 100\}$ passengers and
- an arrival delay of train $A \in \{0, 20, 40, \dots, 200\}$ (in seconds).

The left part of Fig. 7 is a heat map showing the resulting departure delay of train B (in seconds). Note that classical models would assume that train B is always punctual, i.e., this heat map would be completely black. The right part of the figure shows how many passengers are able to board train B for each simulated arrival delay and each simulated number of passengers.

Looking at the left part of Fig. 7 we first see that for an arrival delay of zero, we have no departure delay (graphed in black) if less than 28 passengers wish to transfer as we already know from Fig. 6. The departure delay starts increasing if more passengers transfer. We also see that for an arrival delay of more than 3 min

(180s) the departure delay of train B is zero since nobody reaches the platform of train B before it departs. The following two effects explain the gray-values:

1. As already noted in Fig. 6, not all passengers can board train B at the same time; if many passengers transfer they delay train B . This explains that the departure delay of train B increases with the number of passengers who wish to transfer.
2. Not all passengers walk from train A to train B with the same speed. In particular, if many passengers wish to transfer, it is likely that there is one who is fast enough to reach train B before its scheduled departure. While this passenger boards train B , the next one arrives and boards, too, and so on. This “trickling effect” makes it likely that most of the passengers can board even if the arrival delay of train A is larger than the 2 min transfer buffer time.

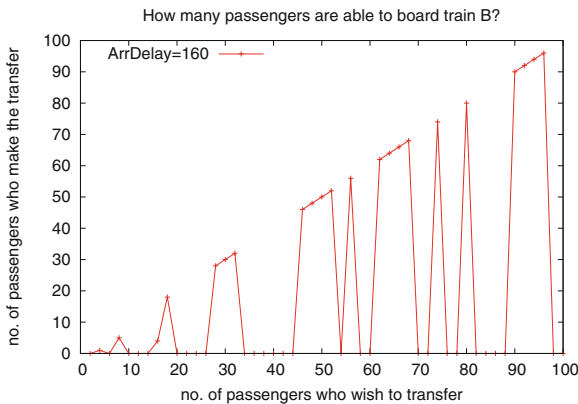


Fig. 8. Number of passengers who manage to board train B compared to number of passengers who wish to do so.

The number of passengers who make the transfer between train A and train B is shown in the right part of Fig. 7. If there is no or a small delay only, all passengers reach B on time and can board. If the delay is larger than 180s, nobody is able to make it. In between, it depends if there is a fast passenger able to reach train B . Then it might be that all passengers can board (again, due to the “trickling effect”), or that only a group of fast passengers reaches the train before it closes its doors and departs. The right part of Fig. 7 shows that the latter gets unlikely if many passengers want to transfer: In our experiments, if more than 18 passengers wish to transfer, either all of them manage to board train B or none of them.

Figure 8 shows the interesting case of an arrival delay of 160s in more detail. The function depicted maps the number of passengers who wish to transfer to train B to the number of passengers who really manage to do so. We see that this is by chance: If there is a fast passenger who reaches train B before it departs, we

have many cases in which all passengers manage to board train B . For a small number of passengers who wish to transfer (in our figure for 16 passengers) we have that not all, but only the faster ones manage to board train B . The specific shape of the function is random.

4.4 Homogeneous and Heterogeneous Passengers

We finally investigate the effect of homogeneity (see Sect. 4.1) of the group of transferring passengers.

Figures 9a and b consider homogeneous passenger groups. Here we see an ‘all or nothing’ effect: Either nobody manages to board train B or the whole group can. Also concerning the departure delay (Fig. 9a), the outcome is not random any more. Nobody is able to catch train B if the arrival delay of train A is larger than 120 s, and the departure delay increases with the number of passengers who wish to transfer.

Figures 9c and d on the other hand show heterogeneous groups. Here we see that it is hard to predict what is going to happen, but it is random if there is a

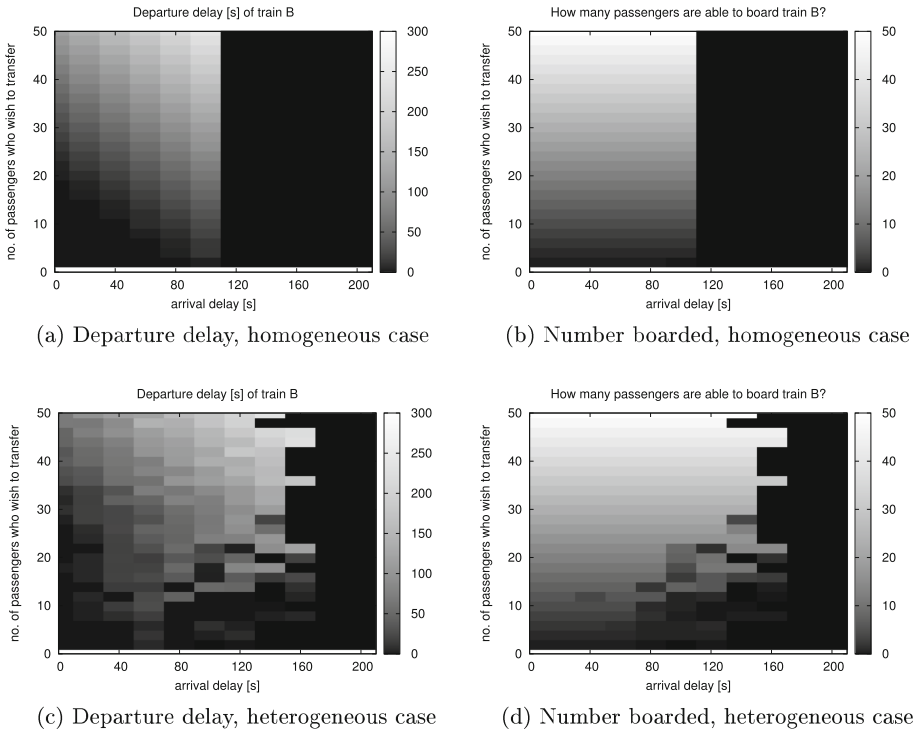


Fig. 9. Departure delay of train B and number of passengers who are able to board, for heterogeneous and homogeneous passengers. Again, the lighter the color, the more delay and the more passengers are observed.

fast passenger in the group of transferring passengers who reaches the platform of train *B* on time. The figure shows that this gets more likely if the number of transferring passengers increases, it is even possible that the whole group manages to board train *B* if the arrival delay of train *A* is 160s which is way more than the transfer buffer time of 2 min.

5 Conclusion and Outlook

This paper describes how to integrate macroscopic railway network simulation with microscopic simulation of passengers' behavior at railway stations, put in the context of delay management. The contribution of this paper is twofold. *First*, we present a novel conceptual model for describing the 'two worlds' of railway network and station as well as their integration in a simulation. *Second*, we report insights gained from preliminary experiments aiming at substantiating the hypothesis that a detailed microscopic simulation of passengers' behavior is needed in the context of delay management applications. The main result drawn from these experiments is that the behavior of the passengers at a station can have a significant influence on the departure delay of trains. This effect has not been taken into account in any delay management models we are aware of. Its intensity increases as more passengers wish to transfer and as the connecting times between trains decrease. The latter is in particular the case if the train from which passengers wish to transfer (train *A* in our scenario, see Fig. 5) arrives at the station with some delay. We have also seen that heterogeneity of the passengers plays an important role in delay creation and propagation and hence adds to former studies such as [7, 19].

These results form the baseline for future research directions: First, we will increase richness and realism of passenger behavior models in the *station* (e.g., by considering PoIs for routing, or more realistic modeling of passenger types, luggage, and passenger groups), on the *platforms* (e.g. by considering realistic train/door topologies including the width of the door which can be modeled by the speed of boarding and deboarding, but also the use of elevators, escalators or ramps to reach a platform), and on the *train*. The latter includes capacity restrictions on trains. More difficult, we also plan to take into account that passengers may dynamically change their planned itineraries, which is known to be hard to treat within optimization models (see [23]). Also, more advanced dynamic traffic demand models, as, e.g., provided by MATSim [12] will be considered. A second important aspect is the modeling of the information state of passengers and the impact of informedness on passengers' behavior (e.g., through interaction with dynamic traffic signs or smart-phones).

The final goal of this research is to integrate the passengers' behavior into delay management (e.g., [9, 10]) by evaluating different delay management strategies and hence help the disposition centers in taking good *wait-depart decisions* in case of delays.

Appendix: Traveller Movement Behaviour, Expressed in *AgentSpeak(L++)* Code

```

1 // walk straight forward towards the goalposition
2 +!movement/walk/forward <-
3     move/forward();
4     !movement/walk/forward.

6 // if walking straight fails, then go left
7 -!movement/walk/forward <-
8     !movement/walk/left.

10 // plan for turning/walking left
11 +!movement/walk/left <-
12     move/left();
13     !movement/walk/forward.

15 // if walk left fails, then go right
16 -!movement/walk/left <-
17     !movement/walk/right.

19 // plan for turning/walking right
20 +!movement/walk/right <-
21     move/right();
22     !movement/walk/forward.

24 // if walking right fails, then wait a random time
25 -!movement/walk/right <-
26     T = math/statistic/randomsimple() * 10 + 1;
27     T = generic/type/toint( T );
28     T = math/min( 5, T );
29     generic/sleep(T).

31 // if the agent has come to standstill, try to speed up
32 +!movement/standstill <-
33     >>attribute/speed(S);
34     S = generic/type/toint(S) + 1;
35     +attribute/speed( S );
36     !movement/walk/forward.

38 +!position/achieve(P, D) <-
39     route/next;
40     !movement/walk/forward.

42 // on waking up after sleeping, set speed and keep on moving
43 +!wakeuper <-
44     +attribute/speed( 1 );
45     !movement/walk/forward.

```

References

1. Ahlbrecht, T., Dix, J., Fiekas, N., Köster, M., Kraus, P., Müller, J.P.: An architecture for scalable simulation of systems of cognitive agents. *Int. J. Agent Oriented Softw. Eng. (IJAOSE)* **5**, 232–265 (2016)
2. Aschermann, M., Kraus, P., Müller, J.P.: LightJason: a BDI framework inspired by Jason. In: Criado Pacheco, N., Carrascosa, C., Osman, N., Julián Inglada, V. (eds.) *EUMAS/AT -2016. LNCS (LNAI)*, vol. 10207, pp. 58–66. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59294-7_6
3. Bazzan, A.L.C., Klügl, F.: Agent-based modeling and simulation. *AI Mag.* **33**(3), 29–40 (2013)
4. Bordini, R.H., Hübner, J.F., Wooldridge, M.: *Programming Multi-agent Systems in AgentSpeak Using Jason*, vol. 8. Wiley, Chichester (2007)
5. Camus, B., Galtier, V., Caujolle, M., Chevrier, V., Vaubourg, J., Ciarletta, L., Bourjot, C.: Hybrid co-simulation of FMUs using DEV&DESS in MECASYCO. In: *Proceedings of the Symposium on Theory of Modeling & Simulation*, pp. 8:1–8:8. Society for Computer Simulation International, San Diego (2016)
6. Codd, E.F.: *Cellular Automata*. Academic Press Inc., Orlando (1968)
7. Conte, C., Schöbel, A.: Identifying dependencies among delays. In: *Proceedings of IAROR 2007* (2007). ISBN 978-90-78271-02-4
8. Dollevoet, T., Huisman, D., Schmidt, M., Schöbel, A.: Delay management with rerouting of passengers. *Transp. Sci.* **46**(1), 74–89 (2012)
9. Dollevoet, T., Huisman, D., Schmidt, M., Schöbel, A.: Delay propagation and delay management in transportation network. In: Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T. (eds.) *Handbook of Optimization in the Railway Industry. International Series in Operations Research & Management Science*, vol. 268, pp. 285–317. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72153-8_13
10. Dollevoet, T., Schmidt, M., Schöbel, A.: Delay management with re-routing of passengers. In: Clausen, J., Stefano, G.D. (eds.) *ATMOS 2009. Dagstuhl Seminar Proceedings* (2009). <http://drops.dagstuhl.de/opus/volltexte/2009/2143>
11. Georgeff, M.P., Lansky, A.L.: Reactive reasoning and planning. In: *Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI 1987*, vol. 2, pp. 677–682. AAAI Press (1987)
12. Grether, D., Nagel, K.: Extensible software design of a multi-agent transport simulation. *Procedia Comput. Sci.* **19**, 380–388 (2013). <http://www.sciencedirect.com/science/article/pii/S1877050913006601>
13. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. *Phys. Rev. E* **51**, 4282–4286 (1995)
14. Johora, F.T., Kraus, P., Müller, J.P.: Dynamic path planning and movement control in pedestrian simulation. In: van Dam, K.H., Thompson, J. (eds.) *Pre-proceedings of 2nd International Workshop on Agent-Based Modelling Of Urban Systems (ABMUS 2017)*, Sao Paulo, Brazil, May 2017, accepted for publication. [arXiv:1709.08235](https://arxiv.org/abs/1709.08235)
15. Kirchoff, F.: Modelling delay propagation in railway networks. *Oper. Res. Proc.* **2013**, 237–242 (2014)
16. Kirchoff, F.: *Verspätungfortpflanzung in Bahnnetzen, Modellierung und Berechnung mit Verteilungsfamilien*. Ph.D. thesis, University of Technology Claustal, Germany (2015)

17. Li, W., Zhu, W.: A dynamic simulation model of passenger flow distribution on schedule-based rail transit networks with train delays. *J. Traffic Transp. Eng.* **3**(4), 364–373 (2016). <http://www.sciencedirect.com/science/article/pii/S2095756415305833>
18. Liang, Y.D., Barsky, B.A.: A new concept and method for line clipping. *ACM Trans. Graph. (TOG)* **3**(1), 1–22 (1984)
19. Manitz, J., Harbering, J., Schmidt, M., Kneib, T., Schöbel, A.: Source estimation for propagation processes on complex networks with an application to delays in public transportation systems. *J. R. Stat. Soc. Ser. C* **66**, 521–536 (2017)
20. Pachl, J.: *Railway Operations and Control*, 2nd edn. VTD Rail Publishing, Mountlake Terrace (2014)
21. Rao, A.S.: Agentspeak(1): BDI agents speak out in a logical computable language. In: *Agents Breaking Away*, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 22–25 January 1996, Proceedings, pp. 42–55 (1996). <https://doi.org/10.1007/BFb0031845>
22. Rückert, R., Lemnian, M., Blendinger, C., Rechner, S., Müller-Hannemann, M.: Panda: a software tool for improved train dispatching with focus on passenger flows. *Public Transp.* **9**(1–2), 307–324 (2017)
23. Schmidt, M., Schöbel, A.: The complexity of integrating routing decisions in public transportation models. *Networks* **65**(3), 228–243 (2015)
24. Schöbel, A.: Capacity constraints in delay management. *Public Transp.* **1**(2), 135–154 (2009)
25. Shalaby, A.S., King, D., Srikukenthiran, S.S.: Using simulation to analyze crowd congestion and mitigation measures at Canadian subway interchanges: case of Bloor-Yonge station, Toronto, Ontario. *Transp. Res. Rec. J. Transp. Res. Board* **2417**, 27–36 (2014)
26. Srikukenthiran, S.S., Shalaby, A.S.: Enabling large-scale transit microsimulation for disruption response support using the Nexus platform: proof-of-concept case study of the Greater Toronto area transit network. *Public Transp.* **9**(1–2), 411–435 (2017)
27. Wales, J., Marinov, M.: Analysis of delays and delay mitigation on a metropolitan rail network using event based simulation. *Simul. Model. Pract. Theory* **52**, 52–77 (2015). <http://www.sciencedirect.com/science/article/pii/S1569190X1500012X>
28. Wooldridge, M.: *An Introduction to Multiagent Systems*. Wiley, New York (2009)
29. Zhang, Q., Han, B., Li, D.: Modeling and simulation of passenger alighting and boarding movement in Beijing metro stations. *Transp. Res. Part C Emerg. Technol.* **16**(5), 635–649 (2008). <http://www.sciencedirect.com/science/article/pii/S0968090X07000927>



Impacts of Vehicle Sharing with Driverless Cars on Urban Transport

Markus Friedrich, Maximilian Hartl^(✉), and Christoph Magg

Institute for Road and Transport Science, University of Stuttgart,
Pfaffenwaldring 7, 70569 Stuttgart, Germany
{markus.friedrich,maximilian.hartl,
christoph.magg}@isv.uni-stuttgart.de

Abstract. Autonomous vehicles (=AV) enabling driverless transport may change the ways of traveling and traffic volumes dramatically. To estimate potential impacts of AV on traffic in an urban area nine scenarios are examined, varying the rate of carsharing, ridesharing and the availability of rail services. The number of required vehicles, vehicle kilometers and the necessary number of parking spaces quantify each scenario.

The study builds on an existing travel demand model of the Stuttgart Region. An algorithm extends this model for bundling person trips in ridesharing systems and by an algorithm for vehicle blocking. The results show that the size of the car fleet can be reduced considerably. The vehicle kilometers traveled in the network, can only be reduced in cases where most travelers use ridesharing instead of carsharing or privately owned cars. However, an increase of the car kilometers traveled is more likely and may lead to a lower quality of traffic flow.

Keywords: Autonomous vehicle · Automated driving · Self-driving car
Carsharing · Ridesharing · Public transport

1 Motivation

Autonomous vehicles (=AV) enabling driverless transport may change the ways of traveling and traffic volumes dramatically. Currently it seems impossible to forecast the point in time, when driverless cars will be ready to service the entire road network on level 5 according to the SAE standards ([1] p. 17). The probability, however, that this time will come is high. Therefore, transport planning should address this topic. AV allow a different kind of transport supply. It is expected that this new type of transport supply will have the following features:

- Car traffic becomes safer.
- Car traffic becomes more comfortable as in-vehicle time can be used for activities not related to driving.
- The capacity of the road network increases at least on highways.

A driverless relocation of vehicles enables new mobility services in carsharing and ridesharing. It also facilitates better intermodal trips when a sharing vehicle operates in areas with a poor public transport supply to take passengers to the railway station.

These changes of transport supply will affect travel demand. Improvements in transport supply can lead to an increase in trip distances and influence the mode choice. The impact on travel demand and traffic flow can be more or less desirable as there are competing objectives in transport planning such as a good service quality, the protection of resources, a city friendly transport and a high level of traffic safety. Several studies and position papers [2–7] describe the spectrum of possible scenarios with AV ranging from “driverless nightmare” to “driverless utopia” ([3] p. 9ff) and from “death of public transport” to “integrated part of public transport”.

This paper presents results from the research project MEGAFON [8], which examines possible impacts of (shared) AV on traffic in urban regions with the help of scenarios. The application is done for the Stuttgart Region. The approach of this project is based on the study “Urban Mobility System Upgrade: How shared self-driving cars could change city traffic” of the International Transport Forum of the OECD [4]. In this study, the impacts of AV on urban traffic are presented taking the city of Lisbon as an example. The study defines eight scenarios in which the trips in motorized traffic are performed to a different extent with autonomous carsharing or ridesharing systems. The number of required vehicles, vehicle kilometers and the necessary number of parking spaces are used to quantify the impacts of each scenario.

2 Assumptions, Modeling and Scenarios

2.1 Data Base

The study builds on demand and supply data from the micro- and macroscopic travel demand model of the Stuttgart Region provided by the Verband Region Stuttgart (VRS). The travel demand is taken from the microscopic model *mobiTopp* [9]. The Institute for Transport Studies at Karlsruhe Institute of Technology developed the model [10]. The implementation in *mobiTopp* (microscopic multi-agent demand model) combines the supply data of the macroscopic demand model with the microscopic travel demand.

mobiTopp covers the travel demand of 2.7 million inhabitants and distinguishes five modes: walk, bike, public transport, car-driver and car-passenger. Every trip is described by origin, destination, mode of transport, day of week, departure and arrival time. The approx. 1,000 traffic zones of the model serve as origins and destinations. Each person is represented as an agent. The mobility behavior of the agents is modeled in the context of their households as observed by the German Mobility Panel [11]. For each agent an activity schedule for one week is created. It is described by a sequence of activities with trip purpose and planned starting time. The microscopic demand for a typical weekday is used as input for the macroscopic travel demand model.

2.2 Considered Modes

The introduction of AV will change the availability of vehicles. The study assumes that AV will replace buses completely. Bus lines with fixed line routes and schedules will no longer exist in their present form. AV then cover trip legs formerly served by buses.

However, rail transport will continue to exist in most scenarios. Rail and AV can be connected to provide intermodal services. The study distinguishes the following motorized modes:

- Private AV (AV-NS): Privately owned vehicles only used by members of one family (NS = NoSharing). These vehicles will not perform empty trips.
- Public AV for individual use (AV-CS): Vehicles used by several travelers consecutively as part of a carsharing system. These vehicles can be relocated to serve the next passenger. The size of these vehicles corresponds to the standard size of a private car.
- Public AV for collective use (AV-RS): Vehicles used by several travelers simultaneously as part of a ridesharing system. The vehicles collect several travelers and take them to their specific destinations. If required empty vehicles are relocated. The size of the vehicles can vary. In the following a uniform vehicle size is assumed, which can carry six persons.
- Public rail services (heavy and light rail): The transport supply and the schedule correspond to the schedule planned for 2025 in the study area.
- Rail + CS: Combination of rail and carsharing as feeder.
- Rail + RS: Combination of rail and ridesharing as feeder.

2.3 Assumptions

The model calculations are based on the following assumptions:

- The total travel demand does not change with the introduction of AV. This means that the number of person trips and total distance traveled, i.e. the destination choice remain unchanged. Mode choice between AV and rail is not influenced by prices.
- The share of non-motorized trips stays the same. The model only considers motorized travel demand, which currently uses public transport or cars as car-driver and car-passenger.
- There are no capacity restraints in rail transport. Every person that intends to travel by rail can do so. The attractiveness of this mode does not suffer from overcrowding.
- All scenarios are based on the assumption that bus transport is completely replaced. Trip legs using buses today are covered by AV. AV can transport travelers on a direct route to their destination or connect them to rail services. Transfers of intermodal services combining AV and rail use railway stations offering the shortest travel time. These transfer stations may be different from the stations used in the current system with buses.
- Access and egress walking time to rail results from the distance between the centroid of the traffic zone to the corresponding railway station. Walking speed is set at 5 km/h. A traffic zone can be connected to one or more railway stations. The ride time within the rail system includes transfer-waiting times according to the schedule, if transfers between trains are required. The following waiting times are distinguished resulting in the waiting times for combination of modes shown in Table 1.

Table 1. Waiting time in combination with selection of mode of transport

| Combination of modes | Waiting times | Components |
|------------------------|---------------|-------------------------------------|
| Walk - Rail - Walk | 4 min | Origin wait time |
| Walk - Rail - AV | 8 min | Origin wait time, transfer time |
| AV - Rail - Walk | 8 min | Registration time, transfer time |
| AV - Rail - AV | 12 min | Registration time, 2x transfer time |
| AV-direct ^a | 8 min | Registration time, collection time |

^aIn order to simplify matters the project assumes that the travel times for AV-CS and AV-RS are identical. In reality, trips completed by AV-CS will be somewhat faster than trips by AV-RS as there are no time losses for collecting other passengers.

- Origin wait time for boarding a rail service at the origin: 4 min
- Transfer times for transfers between AV and rail: 4 min
- Registration time for calling a shared AV: 4 min
- Collection time for collecting and distributing travelers: 4 min
- The speed for car transport corresponds to the typical speed during peak hours in the current state without AV. Lower or higher speed from more or less car trips or from changes in capacity are not considered. The study assumes reduced speed limits in urban areas so that AV can travel safely in mixed traffic:
 - Arterial roads 50 km/h (as today)
 - Main roads 30 km/h (today 50 km/h)
 - Feeder roads 20 km/h (today 30 km/h)
- Trips start and end in traffic zones. Time losses resulting from collecting and distributing passengers and time requirements for intrazonal relocation of AV are estimated with a function that considers the size of the traffic zone.
- Mode choice is not modeled with a utility-based decision model, which considers the characteristics of the competing modes (i.e. in-vehicle time, access and egress time, number of transfers, prices) and the characteristics of the users (for instance car availability). Instead, the distribution between the modes AV-CS and AV-RS is based on predetermined shares defined in the scenarios. The choice between rail and AV depends only on travel time. Travelers always select the faster alternative.

These assumptions neglect expected impacts of privately owned autonomous vehicles (AV-NS), which may include additional traffic from empty trips or private vehicles, comfort-induced trips and in the long term more urban sprawl.

2.4 Scenarios

The scenarios define the availability of rail services (heavy and light rail) and the shares of the travel demand assigned to the modes private AV (no sharing, AV-NS), AV-Carsharing (AV-CS) and AV-Ridesharing (AV-RS). Table 2 gives an overview of the examined scenarios. Scenarios 1 to 6 correspond to scenarios analyzed in the OECD study ([4] p. 18ff). Scenarios 7 to 9 vary the share of the three car modes. In scenario 9, the two sharing modes are combined. In every scenario, bus transport is completely replaced. Scenarios 3 and 4 also assume no rail transport.

Table 2. Scenarios

| Scenario | | Distribution of demand | | | |
|----------|--|------------------------|-------|-------|-------|
| | | Rail | AV-NS | AV-CS | AV-RS |
| 1 | 0% AV-NS, 100% AV-CS, with rail | Yes | 0% | 100% | 0% |
| 2 | 0% AV-NS, 100% AV-RS, with rail | Yes | 0% | 0% | 100% |
| 3 | 0% AV-NS, 100% AV-CS, without rail | No | 0% | 100% | 0% |
| 4 | 0% AV-NS, 100% AV-RS, without rail | No | 0% | 0% | 100% |
| 5 | 50% AV-NS, 50% AV-CS, with rail | Yes | 50% | 50% | 0% |
| 6 | 50% AV-NS, 50% AV-RS, with rail | Yes | 50% | 0% | 50% |
| 7 | 75% AV-NS, 25% AV-CS, with rail | Yes | 75% | 25% | 0% |
| 8 | 75% AV-NS, 25% AV-RS, with rail | Yes | 75% | 0% | 25% |
| 9 | 50% AV-NS, 25% AV-CS, 25% AV-RS, with rail | Yes | 50% | 25% | 25% |

2.5 Modeling Travel Demand

The calculation of the travel demand is done in three steps.

In the first step, the present demand of person trips in motorized traffic (car-driver, car-passenger, public transport) is assigned to the modes railway, AV without sharing (AV-NS) and AV with sharing (AV-CS, AV-RS). The distribution of the travel demand assumes that each traveler chooses the fastest alternative. Because of this rule, many trips that presently combine bus and railway are covered completely by AV in the future. This avoids unnecessary detours and waiting times for the traveler. In order to keep the number of person trips at today's level, speed limits in urban motorized traffic are reduced. However, even with a similar number of passenger trips, the total distance traveled by rail decreases by about 30%. Commuter trains with frequent stops are mainly affected. Metro and express rail can maintain their current ridership level. The express trains offer a high travel speed and the metro benefits from the reduced speed limits in urban areas.

In the second step, the matrix of the person trips with AV-Sharing is transformed into a matrix of vehicle trips. For carsharing, the occupancy rate is set at 1.3 persons. This corresponds to the mean occupancy rate in the Stuttgart Region. For ridesharing the person trips are bundled. For this bundling procedure, a ride-matching algorithm described in [12, 16] is applied. The algorithm can be integrated in existing travel demand models. It works likewise with an integer and non-integer demand representation. The algorithm compares the path sets from the assignment for each origin-destination pair. The matching process compares a reduced representation of a path. The sequence of links is transformed to a sequence of zones. The zone representation works as a buffer along the path. The buffer stands for pick-up and drop-off locations along the path.

In the third step, the vehicle trips with sharing vehicles are concatenated to vehicle blocks in such a way that the number of required vehicles is minimal. This results in additional empty vehicle trips for relocating vehicles. Each vehicle block represents a

required vehicle. The scenarios require the handling of up to 3 million vehicle trips. For this reason, a simple Nearest Neighbor Algorithm is applied for the vehicle blocking. The algorithm selects as successor either the vehicle trip requiring the shortest empty vehicle trip or the shortest waiting time. The conditions for concatenating two vehicle trips are relaxed in four steps. Two vehicle trips are concatenated, if the following conditions hold for the predecessor and the successor trip:

- Hard spatial and hard temporal constraints: Vehicle trips are concatenated if the end of the predecessor and the start of the successor vehicle trip are in the same zone and the waiting time is $< \varepsilon$ ($\varepsilon = 60$ min). The required relocation trip within the zone is determined approximately. The relocation time and distance depends on the zone size with a linear correlation. It is at least 4 min and 100 m.
- Relaxed spatial and hard temporal constraints: Vehicle trips are concatenated if the end of the predecessor and the start of the successor vehicle trip are in the same region (urban district, part of an administrative district) and the waiting time is $< \varepsilon$ ($\varepsilon = 60$ min). This results in short relocation trips.
- Soft spatial and soft temporal constraints: Vehicle trips are concatenated if the end of the predecessor and the start of the successor vehicle trip are in the same region (urban district, part of an administrative district). The waiting time ε is neglected ($\varepsilon = \infty$ minutes). This again results in short relocation trips. Unnecessary long relocation trips are avoided at the expense of longer waiting times.
- No spatial and temporal constraints: Vehicle trips are concatenated without spatial or temporal constraints if the successor trip departs after the arrival of the predecessor trip including the time required to relocate the vehicle. If more than one vehicle trip can be concatenated, the trip with the shortest relocation time is selected. Minimum waiting times and minimum relocation times within zones are considered.

The vehicle blocking algorithm provides the number of required vehicles, person kilometers traveled, person hours spent, AV transport time (service time, relocation time, waiting time) and AV transport distance (service distance, relocation distance).

3 Results

In order to quantify the results of each scenario the following indicators are calculated and presented in Table 3:

- Share of trips completed with public transport
- Number of vehicle (base year = 100%)
- Vehicle kilometers per day (base year = 100%)
- Share of time the vehicles are not in operation
- Medium occupancy rate

Table 3. Overview of selected parameters in the scenarios

| Scenario | | Public transport share | Number of vehicles | Vehicle km | Share of time vehicles are not in use ^a | Medium occupancy rate |
|----------|---|------------------------|--------------------|------------|--|-----------------------|
| 0 | Current State | 16% | 100.0% | 100.0% | 96% | 1.26 |
| 1 | 0% AV-NoSharing 100% AV-Carsharing with rail | 11% | 19.2% | 118.5% | 70% | 1.30 |
| 2 | 0% AV-NoSharing 100% AV-Ridesharing with rail | 11% | 7.1% | 63.9% | 64% | 2.43 |
| 3 | 0% AV-NoSharing 100% AV-Carsharing without rail | 0% | 24.3% | 138.6% | 72% | 1.30 |
| 4 | 0% AV-NoSharing 100% AV-Ridesharing without rail | 0% | 9.2% | 80.6% | 65% | 2.26 |
| 5 | 50% AV-NoSharing 50% AV-Carsharing with rail | 11% | 63.7% | 115.2% | 92% | 1.30 |
| 6 | 50% AV-NoSharing 50% AV-Ridesharing with rail | 11% | 58.2% | 93.5% | 93% | 1.60 |
| 7 | 75% AV-NoSharing 25% AV-Carsharing with rail | 11% | 86.0% | 112.9% | 94% | 1.56 |
| 8 | 75% AV-NoSharing 25% AV-Ridesharing with rail | 11% | 83.2% | 103.9% | 95% | 1.69 |
| 9 | 50% AV-NoSharing 25% AV-Carsharing 25% AV-Ridesharing with rail | 11% | 61.1% | 100.0% | 93% | 1.50 |

^aReferred to all types of vehicles (NS, CS und RS)

3.1 Number of Vehicles

With a level of motorization of approximately 590 vehicles per 1,000 residents, the households in the Stuttgart Region own about 1.6 million cars (state 2010). On a normal workday only 1.0 million of these cars are in use. During peak hours, less than 12% of the 1.0 million cars run simultaneously. These cars are used approximately 60 min per day. Considering all 1.6 million cars, the operating time falls to 40 min per day. In a scenario with 100% ridesharing combined with rail transport, the number of required vehicles decreases to only 7% of today's fleet size. On average, the vehicles are then in use for around 8.6 h. 1.4 h of this time is required for empty vehicle trips and buffer times.

3.2 Total Number of Vehicle Kilometers

Figure 1 shows the vehicle kilometers for the modes AV-NS and AV-Sharing. For sharing vehicles, the vehicle kilometers are displayed for loaded and empty trips. Related to the current state, the vehicle kilometers in the scenarios vary between:

- 64% in scenario 2 (100% ridesharing and railway) and
- 139% in scenario 3 (100% carsharing without railway).

Vehicle kilometers traveled in the network can only be reduced in scenarios with ridesharing. Carsharing does not lead to a relief in traffic volumes. The share of empty kilometers related to the kilometers traveled by all vehicles (including NoSharing) lies between 1% and 6%. Empty trips are responsible for 4 to 9% of the vehicle kilometers of sharing vehicles.

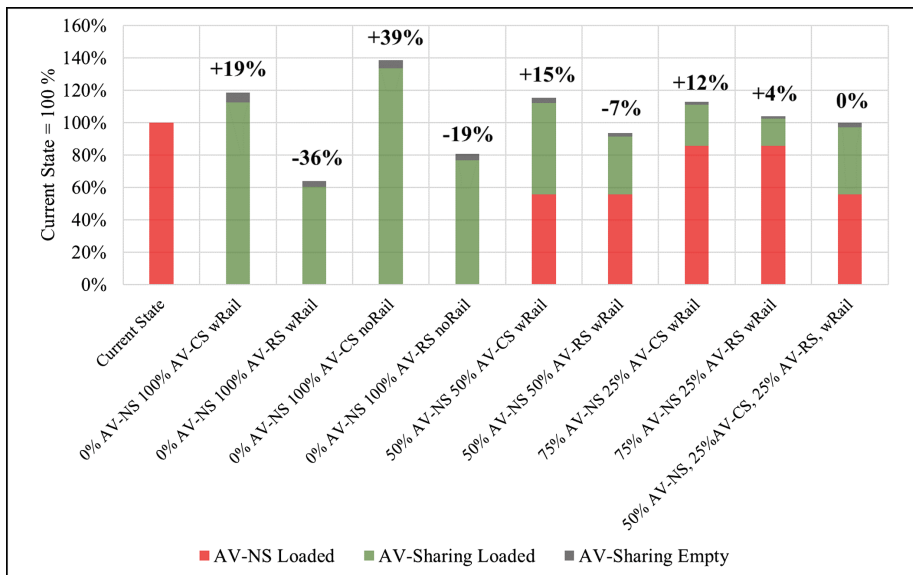


Fig. 1. Vehicle kilometers by mode and loaded/empty trips

3.3 Traffic Volume in the Network

The traffic volume in the scenarios change considerably compared to the current state. Figure 2 shows the relative changes of traffic volumes for the scenarios S1 to S4. For the changes, the six classes (≤ 75 , 75–95, 95–105, 105–120, 120–140 and >140) are distinguished:

- Classes < 95 (green): Traffic volumes are reduced.
- Classes 95 to 105 (grey): Changes in traffic volumes are small.

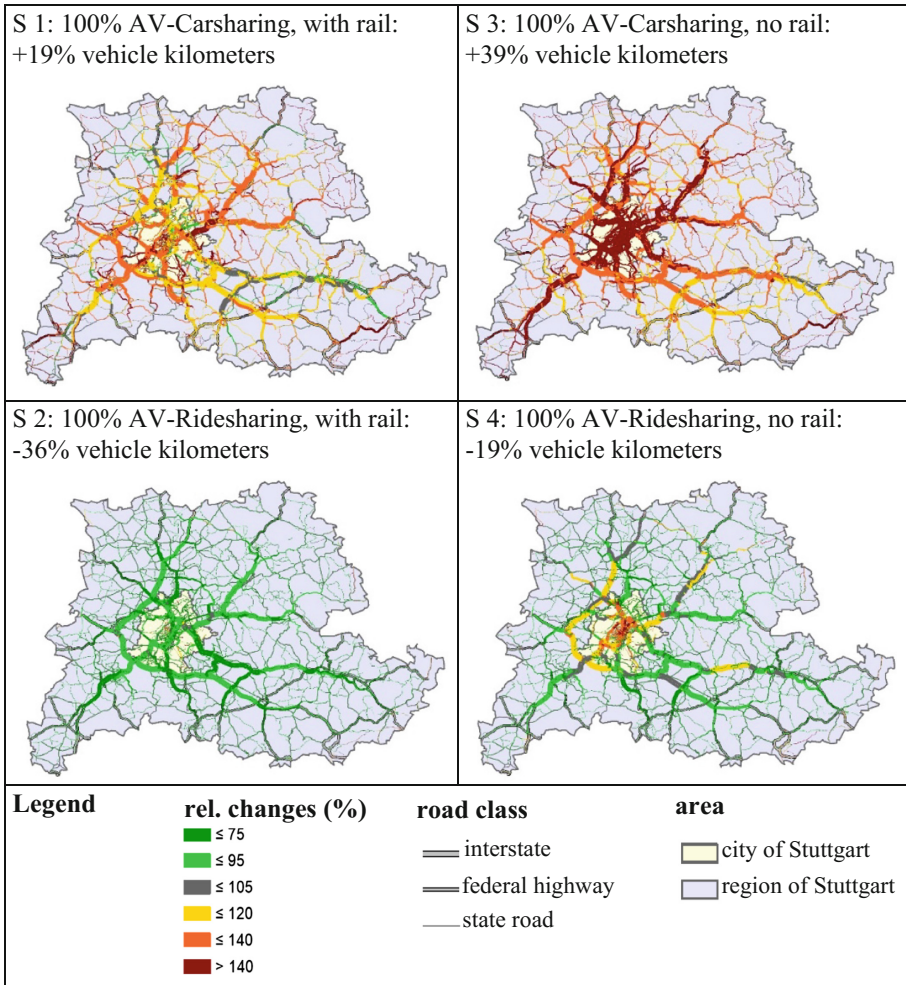


Fig. 2. Relative changes of the car traffic volume when empty trips are added in the scenarios 1 to 4 (current state = 100%) (Color figure online)

- Classes > 105 (yellow or red): Traffic volume increases. Based on the assumption that AV in urban areas can increase the road capacity by about 40% [17], the traffic flow on streets of the classes > 140 will deteriorate.

This leads to the following findings:

- Shifts from public transport to cars due to the elimination of buses and due to shifts from commuter rail in the scenario with 100% carsharing (scenario 1) will lead to an increase of vehicle kilometers. This increase can be compensated on motorways by the increase of the road capacity. In urban areas, this increase might lead to a lower level of service.

- The scenarios 3 and 4, in which a complete abolition of public transport is assumed, show in urban areas an increase of traffic volumes that cannot be compensated, not even with an increased road capacity.

3.4 Vehicle Occupancy

For carsharing vehicles, the occupancy rate is set to 1.3 persons corresponding to the current occupancy rate of private cars. For ridesharing, the occupancy rate varies between 1 and 6 persons per vehicle.

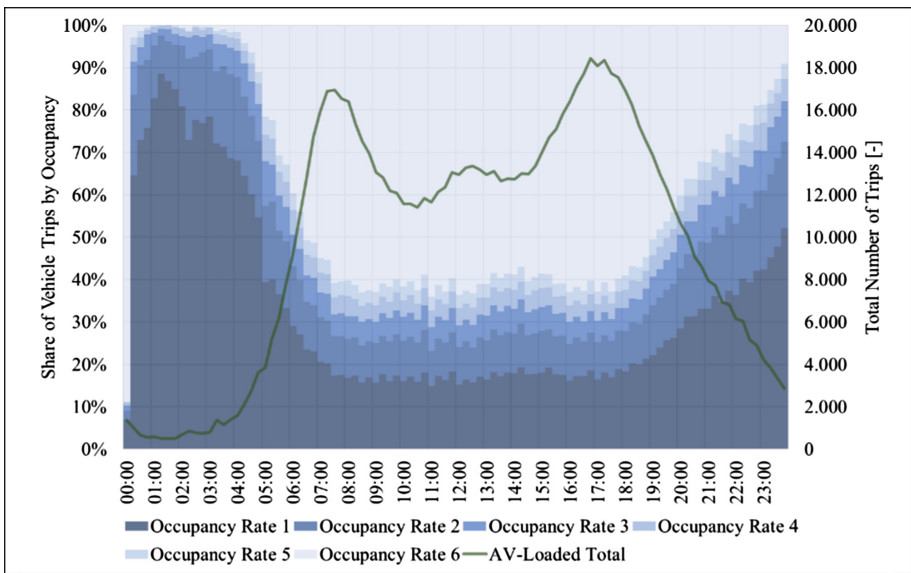


Fig. 3. Occupancy rate for scenario 2 with 100% ridesharing

Figure 3 displays the daytime dependent occupancy rate for scenario 2 with 100% ridesharing. The illustration shows the shares of vehicle trips with 1, 2, 3, 4, 5 and 6 passengers. The occupancy rate of a ridesharing vehicle varies during the course of a vehicle trip similar to a public transport vehicle as passengers are boarding and alighting. The occupancy shown in the figure is the maximum occupancy of the vehicle trip. The figure shows that occupancies with 1 and with 6 passengers are the most common occupancies. Low occupancies occur mainly during off-peak periods and in rural areas. During peak-hours, a higher number of trips can be bundled. The occupancy increases if the total amount of the ridesharing trips increases. The average occupancy rate is 2.5, which is approximately twice the current occupancy rate of a car. The frequent occurrence of a rate of 6 indicates that larger vehicles may be reasonable on certain routes. It also implies that vehicles must be designed in a way that an appropriate amount of privacy and short passenger transfer times are guaranteed.

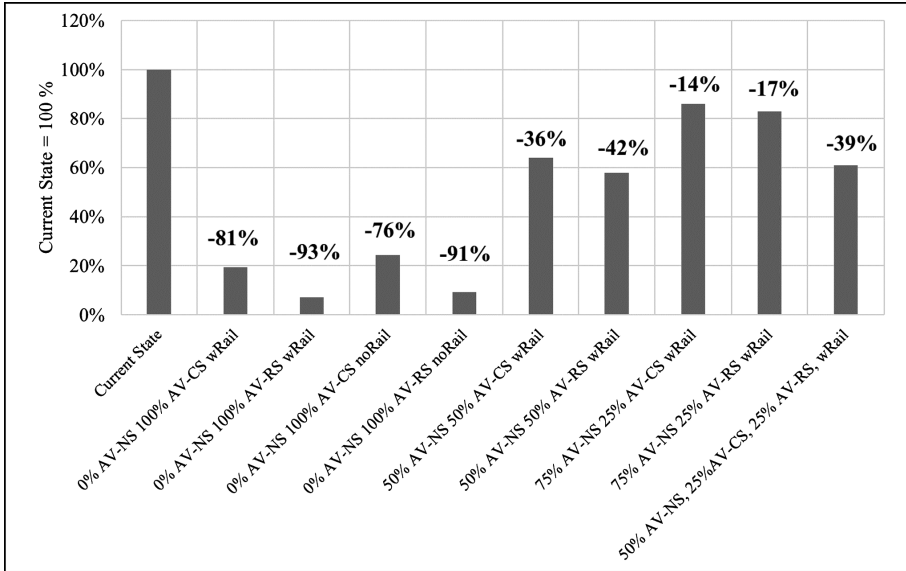


Fig. 4. Number of required vehicles for each scenario

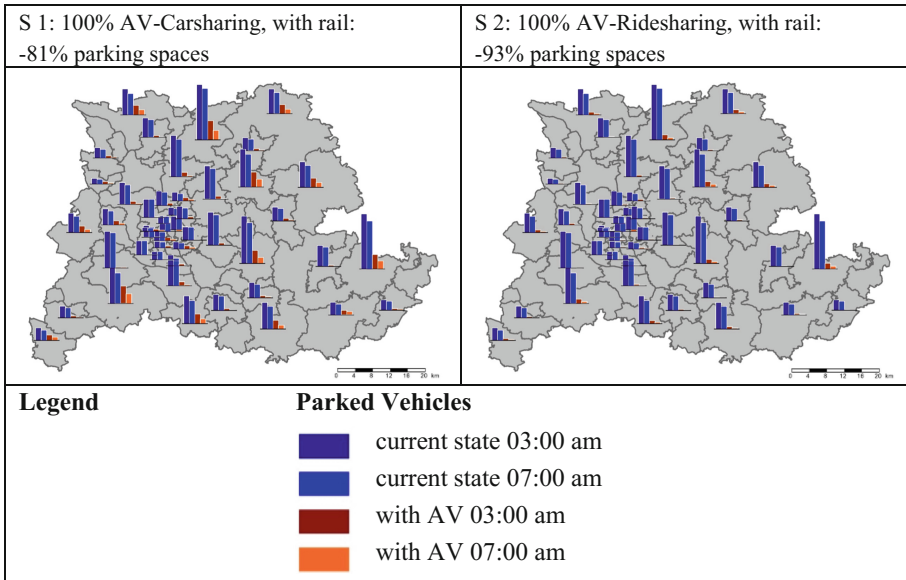


Fig. 5. Number of required parking spaces

3.5 Parking Spaces

Figure 4 shows the reduction of parking spaces for each scenario. The required number of parking spaces correlates to the number of vehicles. Most of the trips happen during the day and define the number of required vehicles in total. During the night, most of the vehicles are waiting for the next trip to be served. In that period, the vehicles occupy parking spaces in regions where the next trip starts. The spatial parking situation is very similar over all scenarios. For instance, the parking situation for scenario 1 (left) and scenario 2 (right) are shown in Fig. 5. The maps illustrate the state with many parked vehicles during the night (03:00 am) and the state with less parked vehicles during the morning peak hour (07:00 am).

4 Conclusion

The introduction of AV may change the transport supply substantially, providing new mobility options for travelers. This will have a major impact on the modal and spatial pattern of travel demand, which are not considered in the scenarios examined in this paper.

The scenarios analyzed in the presented MEGAFON study do not describe states, which are likely to happen. They rather provide estimates to what extent fleets of shared AV may change traffic volumes assuming a constant total travel demand. The scenarios set fixed shares for each mode, which in a future state with AV will be influenced by trip times and trip costs. However, the results for Stuttgart are in the same range as in other studies for Lisbon, Stockholm, Austin and New York [7, 13–15].

Negative impacts might appear if AV are exclusively used in carsharing mode, especially in scenarios with a total replacement of public transport. More cars are required to serve the same demand. This leads to more vehicles on roads, which may cause longer travel times. The impacts of congestion are not explicitly modeled in this study although congestion may influence route choice, ridesharing performance and vehicle blocking. This must be investigated in further research. AV can have positive impacts if the mode choice is influenced in such a way that a high capacity public transport supply (rail-bound or bus rapid transit) is maintained or improved and a high number of travelers choose ridesharing. Such a desirable development requires supporting measures. Without these measures, a decrease of public transport travel demand must be expected due to the following reasons:

- AV will be more comfortable than today's cars, as even persons without driving license will be able to use this mode of transport.
- Trip time will decrease as valet parking eliminates parking search, access and egress walking times.
- On many roads, AV will increase capacity. As a result, trip times by car decrease even in cases higher traffic volumes. For a state with 100% AV, Friedrich [17] estimates a capacity increase of about 40% on urban roads and of about 80% on freeways.

- Privately owned cars with AV-capabilities will not be considerably more expensive than a car of today, but offering the user additional benefits compared to a shared vehicle.
- Reasonably priced services with AV-CS or AV-RS will offer time advantages for journeys without fast and direct public transport connections. These time advantages will reduce public transport ridership on journey currently combining bus and rail, thus leading to fewer person kilometers traveled by rail.
- Transport Network Companies (TNC) of AV-CS and AV-RS could introduce low-cost services leading to a situation where the public authorities can no longer protect public transport.

Examining the impacts of new mobility services on future traffic states is a core task of transport planning. Travel demand models are built for supporting the decision making process of transport planners and policy makers. The results of the presented study provide insights into the impact of new mobility services on transport networks. With the objective of reducing potential negative impacts of AV, the following policy recommendations can be derived from the findings of this study:

- Reduction of speed limits in cities:
The current speed limit in cities is reduced to 30 km/h. On feeder roads, where AV, cyclists and pedestrians share the same road space, the speed limit is reduced to 20 km/h. On these roads, cars no longer have priority over other modes of transport. This measure was already assumed for the presented study.
- Development of specialized ridesharing vehicles:
In order to make ridesharing an attractive alternative for travelers, vehicles should be designed in a way allowing a high degree of privacy. Additionally, there should be enough space for transporting shopping items, school bags and suitcases.
- Introduction of Bus Rapid Transit:
Preservation of bus lines along high demand axes by upgrading the line to a Bus Rapid Transit system.
- Road tolls and parking fees:
To influence the prices of privately operated mobility services, the public authority can implement road tolls. The toll levels should depend on space and time with higher prices in inner cities and during peak hours. Additionally, prices could depend on the occupancy rate of vehicles. Public ridesharing systems or ridesharing systems meeting certain standards could be excluded from these charges. Parking fees and parking regulations could be modified in such a way that every parking place is priced and residential parking is no longer privileged.
- Access restrictions:
In inner city areas where the demand for parking space is very high (station forecourts) access restrictions for individually used cars (AV-NS and AV-CS) could be implemented.
- Unified booking platform:
Development of a uniform platform for public transport services covering bookings and payment.

However, the experience of transport planning over the last decades shows that implementing restrictive measures is a difficult task. Measures like lower speed limits, road tolls and access restrictions could already be implemented today bringing benefits to urban transport in general and to public transport in particular.

Acknowledgement. The research project MEGAFON was financed by the Ministry for Transport of the Land Baden-Württemberg, the VDV (Association of German Transport Companies), the public transport operator SSB (Stuttgarter Straßenbahnen AG) and the transport association VVS (Verkehrs- und Tarifverbund Stuttgart). The Stuttgart Region (VRS) provided the database for the calculations.

References

1. SAE International Standard J3016: Surface vehicle recommended practice (2016)
2. Hazan, J., Lang, N., Ulrich, P., Chua, J., Doubara, X., Steffens, T.: Will Autonomous Vehicles Derail Trains? The Boston Consulting Group (2016). <https://www.bcgperspectives.com/content/articles/transportation-travel-tourism-automotive-will-autonomous-vehicles-derail-trains/>. Accessed 6 Jan 2017
3. Isaac, L.: Driving Towards Driverless – A Guide for Government Agencies. WSP Parsons Brinckerhoff (2015)
4. OECD, International Transport Forum: Urban Mobility System Upgrade. How shared self-driving cars could change city traffic (2015)
5. VDV: Zukunftsszenarien autonomer Fahrzeuge – Chancen und Risiken für Verkehrsunternehmen, Positionspapier (2015)
6. Meyer, J., Becker, H., Bösch, M., Axhausen, K.: Autonomous vehicles: the next jump in accessibilities? *Res. Transp. Econ.* **62**, 80–91 (2017)
7. Maurer, M., Gerdes, C., Lenz, B., Winner, H.: Autonomous Driving. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-48847-8>. Accessed 6 Apr 2018
8. Friedrich, M., Hartl, M.: MEGAFON – Modellergebnisse geteilter autonomer Fahrzeugflotten des öffentlichen Nahverkehrs, Schlussbericht, gefördert von: Ministerium für Verkehr Baden-Württemberg, Verband Deutscher Verkehrsunternehmen e. V., Stuttgarter Straßenbahnen AG, Verkehrs- und Tarifverbund Stuttgart GmbH (2016). <https://www.vdv.de/multimodale-mobilitaet.aspx>. Accessed 6 Apr 2018
9. Mallig, N., Kagerbauer, M., Vortisch, P.: mobiTopp – a modular agent-based travel demand modelling framework. *Procedia Comput. Sci.* **19**, 854–859 (2013). <https://doi.org/10.1016/j.procs.2013.06.114>
10. mobiTopp – Multi-Agenten-Simulation mobiTopp. <http://mobitopp.ifv.kit.edu/28.php>. Accessed 6 Apr 2018
11. MOB – German Mobility Panel. <https://mobilitaetspanel.ifv.kit.edu/>. Accessed 6 Apr 2018
12. Hartl, M., Magg, C., Friedrich, M.: Ridesharing – Ein Modellierungsansatz für das Matching von Fahrtwünschen in makroskopischen Verkehrsnachfragemodellen. Tagungsbericht Heureka 17, FGSV Verlag, Köln (2017)
13. Rigole, P.: Study of a shared autonomous vehicles based mobility solution in Stockholm. Master of Science thesis (2014). <http://kth.diva-portal.org/smash/get/diva2:746893/FULLTEXT01.pdf>. Accessed 6 Apr 2018
14. Fagnant, D.J., Kockelman, K.M.: The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transp. Res. Part C Emerg. Technol.* **40**, 1–13 (2014). <https://doi.org/10.1016/j.trc.2013.12.001>

15. Shen, W., Lopes, C.: Managing autonomous mobility on demand systems for better passenger experience. In: Chen, Q., Torrioni, P., Villata, S., Hsu, J., Omicini, A. (eds.) PRIMA 2015. LNCS (LNAI), vol. 9387, pp. 20–35. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25524-8_2. Accessed 6 Apr 2018
16. Friedrich, M., Hartl, M., Magg, C.: A modeling approach for matching ridesharing trips within macroscopic travel demand models. Compendium of Papers of 97th Annual TRB Meeting, Transport Research Board, Washington, D.C. (2018). Paper is recommended for publication in Transportation Special Issue
17. Friedrich, B.: The effect of autonomous vehicles on traffic. In: Maurer, M., Gerdes, C., Lenz, B., Winner, H. (eds.) Autonomous Driving, pp. 331–350. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-48847-8>. Accessed 6 Apr 2018



Combining Simulation and Optimization for Extended Double Row Facility Layout Problems in Factory Planning

Uwe Bracht¹, Mirko Dahlbeck^{2,3(✉)}, Anja Fischer³, and Thomas Krüger¹

¹ Technische Universität Clausthal, Clausthal-Zellerfeld, Germany
{uwe.bracht, thomas.krueger}@imab.tu-clausthal.de

² Georg-August-Universität Göttingen, Göttingen, Germany
m.dahlbeck@math.uni-goettingen.de

³ Technische Universität Dortmund, Dortmund, Germany
anja2.fischer@tu-dortmund.de

Abstract. We investigate the so called Double Row Facility Layout Problem (DRFLP). Given a set of departments with given lengths and pairwise transport weights between them, the aim is to assign the departments to two rows such that the weighted sum of the distances between them is minimized and such that the departments do not overlap. The DRFLP is known to be rather challenging. Even with the best approach known in literature, which is based on an enumeration over all row assignments of the departments and where only the center-to-center distances are measured, the largest instance solved to optimality contains only 16 departments. In this paper we show how the existing models can be extended in various directions in order to handle more aspects that are important in real-world applications such as vertical distances between the departments and restricting the size of the layout area. We also show how the structure of real-world instances, which often contain several departments of the same type, can be exploited in mathematical optimization. This allows us to solve a realistic instance with 21 departments in reasonable time. Furthermore, we propose a new approach which combines optimization and simulation. Here simulation allows the evaluation of the optimized solutions with respect to several performance indicators which play an important role for a smooth production apart from the weighted transport distances. If problems are detected, this information is included in the mathematical models by extending these.

Keywords: Facility layout problem · Exact solution · Simulation

1 Introduction

Globalization, the growing dynamics of the markets, the increase in customized products, decreasing product life cycles and technological innovations are only some of the challenges manufacturing enterprises have to cope with. As a result,

manufacturing enterprises are forced to implement a cost efficient production in order to remain competitive. The layout of the production areas and operating equipments (assets and departments) is one of the main influencing factors and provides a basis to uphold the long-term productivity and competitiveness [19]. In this work we present a combined optimization-simulation approach for determining a good start solution for the layout of the departments along both sides of a single path. For this we extend the mathematical optimization approach in [7]. The start solution obtained via mathematical optimization is then the basis for the following steps of the factory planners on a much finer level of detail. In order to handle many of the requirements posed on the layout in real-world production, the existing mathematical optimization models have to be extended.

From a mathematical point of view, the described factory planning problem leads to so called facility layout problems, which are widely studied [4]. Several methods have been developed in this area ranging from graphical methods, heuristics, which allow deriving solutions rather fast but without some knowledge of the quality of the solutions, and exact optimization methods. Unfortunately, solving even small instances exactly without additional restrictions on the path structure is extremely challenging. So deriving an exact solution or even a good solution with appropriate solution guarantees for small to medium-sized instances is often rather time-consuming. For this reason, one often concentrates on special cases where one restricts the structure of the layout and the paths. We investigate the so called *Double Row Facility Layout Problem* (DRFLP). Given n departments with positive lengths ℓ_i , $i \in \{1, \dots, n\} =: [n]$, and symmetric pairwise transport weights c_{ij} , $i, j \in [n], i < j$, between them, the classic DRFLP asks for an assignment of n departments to two rows (the two sides of a path) and horizontal positions of the departments such that the weighted sum of the center-to-center distances, measured in horizontal direction, is minimized. Moreover, two departments in the same row may not overlap. So we look for a vector $p \in \mathbb{R}^n$ of positions and a vector $r \in \{1, 2\}^n$ of the assignment of the departments to the two rows such that

$$\begin{aligned} \min \quad & \sum_{\substack{i, j \in [n] \\ i < j}} c_{ij} |p_i - p_j| \\ \text{subject to} \quad & |p_i - p_j| \geq \frac{\ell_i + \ell_j}{2}, \quad i, j \in [n], i < j, \text{ if } r_i = r_j. \end{aligned}$$

In [9], Chung and Tanchoco present a model for the DRFLP (see also [24]) which can solve instances with up to 8 departments in about 10 min. Amaral suggests a mixed-integer program that can solve instances with up to 12 departments in less than one hour [3]. The current best known approach for solving the DRFLP is presented in [12], where Fischer et al. solve instances with up to 16 departments in less than 12 h. Beside the exact methods, there are several heuristic approaches for solving the DRFLP and extensions of it, see, e. g., [17, 25]. Highly related to the DRFLP is the *Single Row Facility Layout Problem* (SRFLP), introduced in [22].

In contrast to the DRFLP, in the SRFLP the departments are arranged in only one row. With the best known approach, presented in [13,14], Hungerländer and Rendl are able to solve instances with up to 42 departments to optimality and they receive very small gaps for instances with up to 110 departments. For an overview of layout problems in general we refer to [4,11].

Our paper is structured as follows. In Sect. 2, we summarize the current best approach for the DRFLP [12]. There, one combines a strong model for the DRFLP with fixed row assignment, i. e., the row assignment of each department is known in advance, with a branching scheme enumerating over all possible row assignments.

In Sect. 3, we extend this approach in various directions. We consider departments as 2-dimensional objects which have a length and a width. In real-world applications the size of a factory is limited, so in Sect. 3.1 we show how to restrict the area used for the DRFLP layout. In Sect. 3.2, we allow the consideration of vertical distances between the departments. Afterwards, in Sect. 3.3 we consider instances that contain departments of the same type, i. e., these departments have the same length and the same transport weights to all other departments. We exploit this structural property of departments of the same type by reducing the number of relevant row assignments significantly.

Unfortunately, in the classic mathematical DRFLP models only the transport weights and so the weighted transport loads are taken into account. But there are several further indicators that are important for guaranteeing a smooth production, e. g., the throughput of the factory, the cycle times of the products or the used storage and buffer capacities. Therefore, we combine in Sect. 4 the mathematical DRFLP model with a simulation of the production that allows determining various key performance indicators. Thus, we can detect potential problems and conflicts.

In Sect. 5, we solve a realistic instance with 21 departments in less than 14 h by exploiting that there are several departments of the same type. We compare our solutions obtained via mathematical optimization to solutions derived by applying classic methods used in factory planning. The results obtained via simulation are then the starting point in Sect. 5.2 for extending the mathematical models such that the transport distances are considered not only in an aggregated form, but for each single product. We summarize our results and give suggestions for future work in Sect. 6.

2 Basic Model for the DRFLP

In the classic models for the DRFLP, see, e. g., [3,9,12], the following three assumptions are made

1. the total size of the area needed for the arrangement is not limited,
2. vertical distances between the departments are neglected,
3. each department can be assigned to any of the two rows.

In contrast to the SRFLP, there might occur free spaces between departments in the same row in optimal DRFLP solutions.

The current best solution approach for the DRFLP is described in [12]. The main idea is to enumerate over all possible row assignments and solve the DRFLP with fixed row assignment (FR-DRFLP). In the following, we summarize the model of [12] for solving the FR-DRFLP. First we add two dummy departments $n + 1$ and $n + 2$ representing the left and right border of the layout. The lengths and transport weights of the dummy departments are set to zero, i. e., $\ell_{n+1} = \ell_{n+2} = 0$, $c_{i(n+1)} = c_{i(n+2)} = 0$ for $i \in [n]$ and $c_{(n+1)(n+2)} = 0$.

In order to consider a fixed row assignment, let $R = \{1, 2\}$ be the set of rows and $r_i \in R$, $i \in [n]$, be an assignment of the departments to the two rows. For $h \in R$ we will write: $j \in R_h \Leftrightarrow r_j = h$. The dummy departments $n + 1$ and $n + 2$ are assigned to both rows and we define $\tilde{R}_h = R_h \cup \{n + 1, n + 2\}$. We use betweenness variables

$$x_{ikj} = x_{jki} = \begin{cases} 1, & k \text{ lies between } i \text{ and } j \text{ in the same row,} \\ 0, & \text{otherwise,} \end{cases}$$

for $l \in R$, $i, j, k \in \tilde{R}_l$, $i \neq k \neq j$, $i < j$. The betweenness variables induce an order of the departments in each row, because $x_{(n+1)ij}$ is equal to 1 if and only if department i is left to department j in the same row. We consider the following integer linear programming model

$$x_{ijk} + x_{ikj} + x_{jik} = 1, \quad l \in R, i, j, k \in \tilde{R}_l, i < j < k, \quad (1)$$

$$x_{(n+1)i(n+2)} = 1, \quad i \in [n], \quad (2)$$

$$x_{ikj} = 0, \quad l \in R, i, j \in R_l, i < j, k \in \{n + 1, n + 2\}, \quad (3)$$

$$x_{(n+1)ij} = x_{ij(n+2)}, \quad l \in R, i, j \in R_l, i \neq j, \quad (4)$$

$$x_{ihj} + x_{ihk} + x_{jhk} \leq 2, \quad l \in R, i, j, k, h \in \tilde{R}_l, |\{i, j, k, h\}| = 4, i < j < k, \quad (5)$$

$$-x_{ihj} + x_{ihk} + x_{jhk} \geq 0, \quad l \in R, i, j, k, h \in \tilde{R}_l, |\{i, j, k, h\}| = 4, \quad (6)$$

$$x_{ijk} \in \{0, 1\}, \quad l \in R, i, j, k \in \tilde{R}_l, |\{i, j, k\}| = 3, i < k. \quad (7)$$

If three departments lie in the same row, by (1) exactly one of them lies in the middle. The constraints (2)–(4) ensure that every department lies between the dummy departments $n + 1$ and $n + 2$, i. e., the dummy departments are the left and right border of the layout. Inequalities (5)–(6) imply that the departments satisfy certain transitivity properties. According to [2], (1) and (5)–(7) induce a correct ordering of the departments in each row.

Next we need to calculate the distance between two distinct departments. The horizontal position p_i of the center of department i , $i \in [n]$, is given by $d_{i(n+1)} = p_i$. The value $d_{i(n+2)}$ is defined as the distance between the right border of the layout, i. e., department $n + 2$, and department i for $i \in [n]$. The distance between the left and the right border of a layout is given by $d_{(n+1)(n+2)}$. The distance is calculated according to (see [12])

$$d_{ji} = d_{ij} \geq |p_i - p_j| = |d_{(n+1)i} - d_{(n+1)j}|,$$

for $i, j \in [n] \cup \{n+2\}, i < j$. We set $M := \sum_{i=1}^n \ell_i$ and we obtain in the following a model for the FR-DRFLP, which we call $\text{IP}_{\text{FR-DRFLP}}$. This model is given in its basic form in [12] and we add the inequalities (9) and (10), because we later want to extend this model.

$$\min \sum_{\substack{i, j \in [n] \\ i < j}} c_{ij} d_{ij}$$

s. t. (1)–(7),

$$d_{j(n+1)} - d_{i(n+1)} \geq M(x_{(n+1)ij} - 1) + \frac{\ell_i + \ell_j}{2}, \quad l \in R, i, j \in R_l, i \neq j, \quad (8)$$

$$d_{j(n+2)} - d_{i(n+2)} \geq M(x_{ji(n+2)} - 1) + \frac{\ell_i + \ell_j}{2}, \quad l \in R, i, j \in R_l, i \neq j, \quad (9)$$

$$d_{i(n+1)} + d_{i(n+2)} = d_{(n+1)(n+2)}, \quad i \in [n], \quad (10)$$

$$d_{i(n+1)} \geq \frac{\ell_i}{2}, \quad d_{i(n+2)} \geq \frac{\ell_i}{2}, \quad i \in [n], \quad (11)$$

$$d_{ik} + d_{kj} \geq d_{ij}, \quad \begin{array}{l} i, j, k \in [n+2], i < j, \\ |\{i, j, k\}| = 3, \end{array} \quad (12)$$

$$d_{ij} \geq 0, \quad i, j \in [n+2], i < j. \quad (13)$$

By inequalities (8), (9) and (11) there is a minimal distance of $\frac{\ell_i + \ell_j}{2}$ between the centers of the departments i and j if they lie in the same row and with respect to the dummy departments. Inequalities (12) are triangle inequalities that also connect departments lying in different rows. We argued above that the betweenness inequalities (1) and (5)–(7) induce a correct ordering of the departments in each row. Combining this with (8)–(13) we get the following theorem:

Theorem 1. *The model (1)–(13) is correct for the FR-DRFLP.*

This result follows immediately from [12] which we adapted only slightly such that also the distance between some department $i, i \in [n]$, and department $n+2$ is calculated correctly.

Furthermore, we can add a lower bound on the distance between the centers of two departments i and j in the same row by summing up the lengths of all departments between i and j

$$d_{ij} \geq \frac{\ell_i + \ell_j}{2} + \sum_{k \in R_l \setminus \{i, j\}} \ell_k x_{ikj}, \quad l \in R, i, j \in \tilde{R}_l, i < j.$$

We want to point out that the distance between two departments might be greater than this bound, because in an optimal solution of the FR-DRFLP there might occur free space between two neighboring departments.

In order to solve the DRFLP using the model above, we have to test exponentially many row assignments. We can reduce the number of distinguishable row

assignments by reducing M , the big- M -value in inequalities (8) and (9), which is also an upper bound on the sum of the lengths of the departments in each single row. Certainly we can assume that in an optimal solution the sum of the lengths of the departments in row one is the same as or larger than the sum in row two. Let the leftmost department i of the considered layout start at position $p_{n+1} = 0$ with its center $p_i = \frac{\ell_i}{2}$ and let the rightmost department $j \in [n]$ apart from $n + 2$ of this layout finish at t with its center $p_j = t - \frac{\ell_j}{2}$.

Lemma 1 ([12]). *Given a DRFLP instance that satisfies, w. l. o. g., $\ell_i \leq \ell_{i+1}$ for $i \in [n - 1]$, there always exists an optimal DRFLP layout on the interval $[0, t]$ with*

$$t \leq \sum_{i=\lfloor \frac{n+1}{3} \rfloor + 1}^n \ell_i. \tag{14}$$

Moreover, this bound is tight.

Due to [12], we can neglect all row assignments where the sum of the lengths of the departments in one of the rows exceeds t .

Usually, in factory planning the incoming warehouse and the shipping warehouse of a factory are arranged at the left and at the right border, respectively. If this is the case, the dummy departments $(n + 1, n + 2)$ can be interpreted as these warehouses. For an illustration we refer to Fig. 1. Of course, we might obtain a better overall solution value if we drop the restriction on the position of both warehouses. In this case they are treated as ordinary departments that have transport connections to other departments and need a certain space. Later in Sect. 5 we compare the quality of the solutions with and without this restriction on the positions.

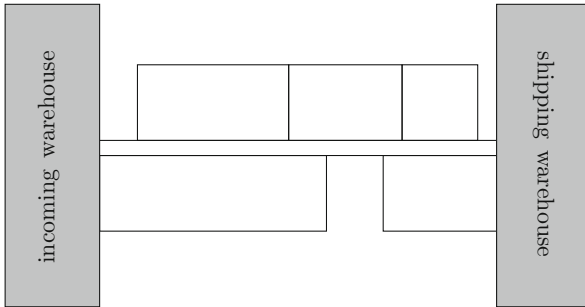


Fig. 1. Visualization of an extended DRFLP layout where we fixed the incoming and the shipping warehouse to the border of the layout. One motivation for this arrangement is that one hopes to receive rather linear transport flows between the departments.

3 Extensions of the Basic DRFLP Model

In this section we show how the DRFLP and the FR-DRFLP can be extended such that further aspects, which are relevant in practice, can be handled in optimization.

3.1 Restricted Area of the Whole Layout and Blocked Areas

We consider the case that the departments not only have a length but also a width, i. e., they are given as 2-dimensional objects. Our aim is to place the departments in a restricted area. In [21], a restricted area is taken into account by a penalty function. However, we will restrict the area by additional constraints. In factory planning the layout area is usually defined as follows:

Definition 1. *The area of a given layout is defined as the area of the minimum boundary rectangle containing all departments.*

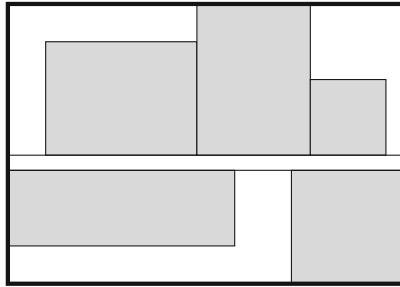


Fig. 2. Minimum boundary rectangle of a layout (marked black) enclosing five 2-dimensional departments. The area of the layout corresponds to the size of this rectangle.

By definition the area of a layout is equal to $d_{(n+1)(n+2)} \cdot w$, where w is the width of the layout (in our FR-DRFLP model). An example is illustrated in Fig. 2. Let w_i denote the width of department $i \in [n]$. In the FR-DRFLP the row assignment is fixed, so we compute the width of the layout by summing up the width of the department with the largest width in each row plus the width of the path w_{path} between the two rows, i. e., $w = \max_{i \in R_1} w_i + \max_{i \in R_2} w_i + w_{path}$. In particular, for a fixed row assignment the width of a layout is constant. Thus, we only need to restrict the distance $d_{(n+1)(n+2)}$ in an appropriate way to restrict the used area. Assume the area of the layout may be at most $F \in \mathbb{R}_{\geq 0}$. Then the linear inequality $d_{(n+1)(n+2)} \leq \frac{F}{w}$ ensures that the area of the layout is bounded by F .

So, given a row assignment, all departments lie in the interval $[0, \frac{F}{w}]$. Thus, we can neglect all row assignments where the sum of the lengths of the departments

in the same row exceeds $\frac{F}{w}$. Moreover, we can improve the big- M -value used in inequalities (8) and (9) to

$$M = \min \left\{ \frac{F}{w}, \sum_{i=1}^n \ell_i \right\}.$$

Hence, for every row assignment we have to compute a new big- M -value. Note that the upper bound t , as computed in (14), is not valid anymore.

Apart from a restriction of the used area, there might appear so called blocked areas in real-world factory planning problems. It is not allowed to place departments in these areas. This might be due to already existing departments or due to safety restrictions. Let $B_1 = \{[b_1, b_1 + g_1], \dots, [b_u, b_u + g_u]\}$ be the blocked areas in row 1 and $B_2 = \{[b_{u+1}, b_{u+1} + g_{u+1}], \dots, [b_v, b_v + g_v]\}$ be the blocked areas in row 2 for given $b_k, g_k \geq 0, k \in [v]$, $b_k + g_k \leq b_{k+1}, k \in [v] \setminus \{u\}$. For each blocked area we introduce a new dummy department, which we will call blocked department, with length equal to the length of the blocked area. We place the center of the blocked department in the middle of the blocked area. So we get the blocked departments $n + 3, n + 4, \dots, (n + 2 + |B_1| + |B_2|)$ with length $\ell_{n+2+k} = g_k$ for $k \in [v]$. The row assignment of the blocked departments is fixed, namely $R^1 = \{(n + 3), \dots, (n + 2 + |B_1|)\}$ are assigned to row 1 and $R^2 = \{(n + 2 + |B_1| + 1), \dots, (n + 2 + |B_1| + |B_2|)\}$ to row 2. To ensure that the blocked department $n + 2 + k$ lies exactly on the interval $[b_k, b_k + g_k]$, we set the distance variable to

$$d_{(n+1)(n+2+k)} = b_k + \frac{g_k}{2}, \quad k \in [v].$$

Additionally, we extend the inequalities (8) such that they are satisfied for all departments $i, j \in R_l \cup R^l$ for $l \in \{1, 2\}$. Apart from this we can fix the betweenness variables to belong to each three departments with index at least $n + 1$. For correctness of the model an update of the big- M -value to $\tilde{M} = \max\{b_u + g_u, b_v + g_v\} + t$, where t is defined as in (14), is needed, since it might happen that in an optimal solution the n departments are all arranged right to the blocked departments (possible if the area of the layout is not additionally bounded; this is possible by restricting $d_{(n+1)(n+2)}$). Note that M has to be further enlarged if the blocked areas have non-zero transport weights to departments in $[n]$. This can happen if we extend an existing factory and do not want to move some of the old departments.

3.2 Vertical and Inter-row Distances

We want to overcome the second assumption of the basic model as stated in Sect. 2—vertical distances between the departments are neglected—by adding inter-row distances between departments in distinct rows as well as vertical distances between departments in the same row. For this, note again that we assume that the departments are 2-dimensional objects. The center-to-center distance

between two departments i and j is computed as follows: First we add the distance between the center of i to the path, then, as in the 1-dimensional case, we compute the distance $|p_i - p_j|$ and afterwards we add the distance from the path to the center of j . Furthermore, if i and j are in distinct rows, we add the width of the path w_{path} . An example is illustrated in Fig. 3.

In order to solve this extended DRFLP, we use again our fixed-assignment model $IP_{FR-DRFLP}$. For the FR-DRFLP, the inter-row distances and associated transport weights are constant. The inter-row weights are calculated by

$$\sum_{\substack{j \in R_1 \\ k \in R_2}} \left(\frac{w_j + w_k}{2} + w_{path} \right) c_{jk}$$

and for departments in the same row we get

$$\sum_{\substack{j, k \in R_1 \\ j < k}} \frac{w_j + w_k}{2} c_{jk} + \sum_{\substack{j, k \in R_2 \\ j < k}} \frac{w_j + w_k}{2} c_{jk}.$$

All in all, in our setting we only need to add a constant to the objective value of some FR-DRFLP to include inter-row distances and compare the total objective values in the enumeration scheme. We want to note again, that the reduction of M according to (14) is not possible because of this constant.

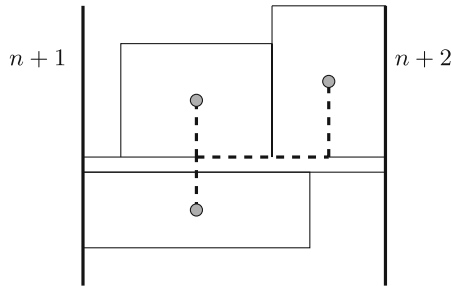


Fig. 3. Calculation of the vertical and horizontal distances between three departments

Naturally, the third assumption mentioned at the beginning of Sect. 2—each department can be assigned to any of the two rows—can easily be dropped. If the row assignment of some departments is fixed in advance, this only helps us because the number of possible row assignments decreases.

3.3 Departments of the Same Type

In order to compute an optimal DRFLP layout we enumerate over all row assignments of the departments and solve some $IP_{FR-DRFLP}$ in each step. In principle,

we have to solve the $\text{IP}_{\text{FR-DRFLP}}$ for all distinguishable row assignments (some layouts might be neglected due to further considerations). So restricting the number of distinguishable row assignments is essential. Let n denote the number of departments. In general, there are $\frac{1}{2} \cdot 2^n$ distinguishable row assignments because by assumption we can place every department in row 1 or in row 2 and we can fix the row assignment of exactly one department. In realistic instances there appear departments of the same type, see also our test case in Sect. 5.1, i. e., the departments have the same length and the same transport weight c to all other departments. We use this additional information to reduce the number of distinguishable row assignments significantly.

Theorem 2. *Let m denote the number of different department types and let a_i be the number of departments of type $i \in [m]$. Then there are at most*

$$\left\lceil \frac{1}{2} \prod_{i \in [m]} (a_i + 1) \right\rceil$$

distinguishable row assignments.

Proof. We will prove this result by induction on m . Let $m = 1$. We only take row assignments into account that contain at least as many departments in row 1 as in row 2. By symmetry, these are all distinguishable row assignments. So we assign $a_1, \dots, \lceil \frac{a_1}{2} \rceil$ departments to row 1 and we obtain $a_1 - \lceil \frac{a_1}{2} \rceil + 1 = \lceil \frac{a_1+1}{2} \rceil$ distinguishable row assignments. Let us now assume that the result is true for m and we consider $m + 1$ department types.

Case 1: a_{m+1} is odd. The idea of the proof is to assign more departments of type $m + 1$ to row 1 than to row 2. By this method, we take all distinguishable row assignments into account. For $a_{m+1}, \dots, \lceil \frac{a_{m+1}}{2} \rceil$ departments of type $m + 1$ in row 1 we obtain $\prod_{i=1}^m (a_i + 1)$ distinguishable row assignments in each subcase. Altogether we obtain

$$\begin{aligned} \left(a_{m+1} - \left\lceil \frac{a_{m+1}}{2} \right\rceil + 1 \right) \prod_{i=1}^m (a_i + 1) &= \left\lceil \frac{a_{m+1} + 1}{2} \right\rceil \prod_{i=1}^m (a_i + 1) \\ &= \left\lceil \frac{1}{2} \prod_{i=1}^{m+1} (a_i + 1) \right\rceil \end{aligned}$$

distinguishable row assignments.

Case 2: a_{m+1} is even. We assume that a_i is even for all $i \in [m]$, otherwise the proof is analogous to Case 1. Similar to Case 1 we assign $a_{m+1}, \dots, \frac{a_{m+1}}{2} + 1$ departments of type $m + 1$ to row 1 and obtain

$$\frac{a_{m+1}}{2} \prod_{i=1}^m (a_i + 1)$$

distinguishable row assignments. It remains to consider the case with $\frac{a_{m+1}}{2}$ departments of type $m + 1$ in row 1. Then, there are also $\frac{a_{m+1}}{2}$ departments of type $m + 1$ in row 2. Now we use our induction hypothesis to create distinguishable row assignments. Altogether we get

$$\begin{aligned} \frac{a_{m+1}}{2} \prod_{i=1}^m (a_i + 1) + \left\lceil \frac{1}{2} \prod_{i=1}^m (a_i + 1) \right\rceil &= (a_{m+1} + 1) \frac{1}{2} \prod_{i=1}^m (a_i + 1) + \frac{1}{2} \\ &= \left\lceil \frac{1}{2} \prod_{i=1}^{m+1} (a_i + 1) \right\rceil \end{aligned}$$

distinguishable row assignments. \square

This formula is also correct if all departments have a different type, because then $a_i = 1$ for all $i \in [m]$ and $m = n$. We illustrate the advantages of Theorem 2 by a realistic example, see [18] and Sect. 5.

Example 1. We are given $n = 21$ departments, where two departments appear four times, three departments twice and seven departments just once. Without reduction, we have to test $2^{20} = 1048576$ row assignments. By Theorem 2 we obtain at most $\frac{1}{2} \cdot 5 \cdot 5 \cdot 3 \cdot 3 \cdot 3 \cdot 2^7 = 43200$ distinguishable row assignments.

Apart from reducing the number of row assignments if there are several departments of the same type, we additionally can strengthen our model. Indeed, we can break some symmetries of the arrangement by fixing the order of departments of the same type in the same row. This symmetry breaking is done in such a way that at least one optimal solution is preserved. Let a_{i_1} departments of the same type $i, i \in [m]$, be in row 1. We denote these departments, w.l.o.g., by $1, \dots, a_{i_1}$. Then, we fix the order of these departments by additional constraints, w.l.o.g., we use an ascending order. Since these departments are of the same type, they have the same length and we can add

$$d_{(n+1)1} \leq d_{(n+1)2} + \ell_1 \leq \dots \leq d_{(n+1)(a_{i_1})} + (a_{i_1} - 1) \cdot \ell_1$$

to our model. It follows immediately that we can set the ordering variables to

$$x_{(n+1)kl} = \begin{cases} 1, & k, l \in [a_{i_1}], k < l, \\ 0, & k, l \in [a_{i_1}], k > l. \end{cases}$$

Similar equations can be added for department $n + 2$. Furthermore, we fix the associated betweenness variables

$$x_{kuv} = \begin{cases} 1, & k, u, v \in [a_{i_1}], k < u < v, \\ 0, & k, u, v \in [a_{i_1}], k < v \text{ and } (u < k \text{ or } u > v). \end{cases}$$

4 Iterative Combination of Optimization and Simulation

Using an extended version of the algorithm of [12] we are now able to solve the DRFLP. For further details on the software we used as well as on our test environment we refer to Sect. 5. As already mentioned in the introduction, in the mathematical models usually only the transport loads are taken into account. But for a successful production system, which highly depends on the decisions made during the factory planning process, several further key performance indicators play a significant role. To determine these we use simulation. If problems are detected, the optimization model is extended.

To verify the quality of the extended DRFLP model we apply discrete event simulation, see also [7]. As a software tool we use Tecnomatix Plant Simulation [23]. Starting point is the development of a basic simulation model that includes different controls and import functions. These controls are necessary for

- the management of processing sequences and times,
- the implementation of imported processing parameters or production programs and
- an automated generation of the layout specific simulation model after the import of the DRFLP solution.

In addition, the controls allow

- the consideration of different distribution strategies for the material flow,
- adding different products,
- adjusting the processing sequences, i. e., in which order the products have to be processed, and
- adjusting a production program which includes a production schedule and the number of products.

By running the simulation of the processes on and between the departments or machines, which are arranged according to some layout, we generate dynamic and realistic information about the transport processes.

Additionally, we implemented some statistical tools for the evaluation of the respective layouts. Related to the input data for the DRFLP we analyze the total product distances [1], specific product distances and the transport momentum. Furthermore, a first benefit of the simulation is that we can consider additional key performance indicators of the production systems, among them output, throughput times, inventory, capacities, utilization of resources. Apart from this a second benefit is the visualization of processes that simplifies the understanding of complex relations [8].

An analysis of the results including the key performance indicators is then the basis to see needs for improvement. So, if the current layout has to be improved, the DRFLP models are customized by extending or adapting the mathematical models and the interplay between optimization and simulation continues as illustrated in Fig. 4. One big advantage of our iterative layout creation

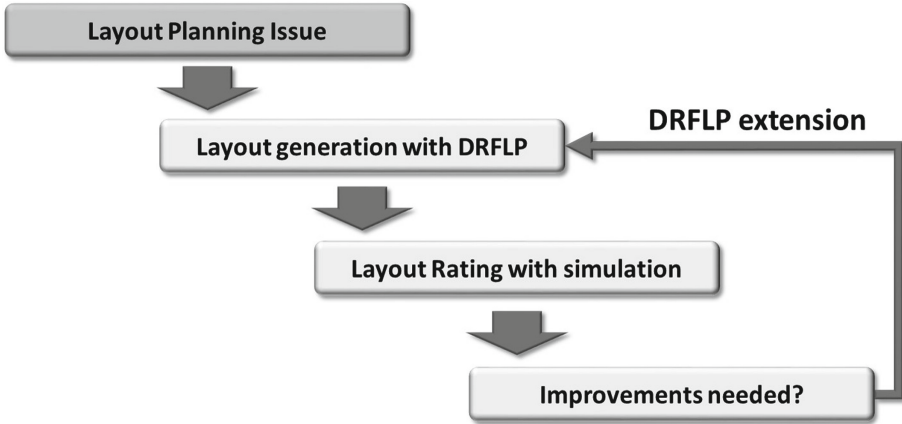


Fig. 4. Extended and evaluated DRFLP-method: The quality of a layout determined via mathematical optimization for some layout planning issue is measured using simulation. If improvements are needed, the mathematical models are extended.

is that afterwards we can nicely compare the found solutions with respect to several indicators. So, the effect of certain decisions becomes clear.

The simulation might show that the distances between certain departments are too large for a smooth production. Then, we can restrict these distances in the model. Furthermore, the simulation gives some information about the size of the storage and buffer areas needed during production. If more or less space is needed, the sizes of the departments have to be adapted in the next step. Additionally, in the mathematical model we always assume that the transport weights between each pair of departments is known in advance. If there are several departments of the same type we assume that the intermediate products are evenly distributed among the departments of the same type. With the help of simulation we can check whether this is a good distribution strategy by testing several ones and if necessary we can adapt our model.

Many simulation models are generated in 2D. This kind of department representation is quite abstract and impedes the intuitive understanding of the layout and the production process. Especially for layouts with an increased number of departments, the transparency of a DRFLP solution with the 2D simulation model is limited. The integration of 3D models provides a better overview for the planner as illustrated in Fig. 5. Especially the product flow can be demonstrated very quickly. Using a 3D simulation model on basis of the optimized layout can simplify the virtual validation of the planned production area and the detection of bottlenecks. All in all, simulation allows to control whether it will be possible to achieve the desired output of the production system afterwards in real production.

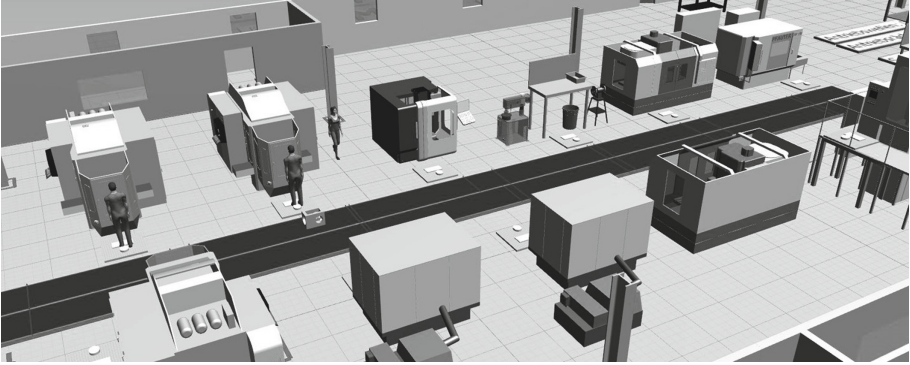


Fig. 5. Illustration of a 3-dimensional simulation including the workers where the departments are arranged on both sides of a common path

5 Computational Experiments

In this section, we present our computational results. All experiments were conducted on an INTEL-Core-I7-4770 (4×3400 MHz, 8 MB Cache) with 32 GB RAM in single processor mode using openSUSE Linux 42.1. We used CPLEX 12.7.0 [15]. As mentioned above all simulations were done with Tecnomatrix Plant Simulation [23].

5.1 Test Case and Computational Results

For testing our new solutions of the extended DRFLP models in the simulation we use a well-known application example [18]. It represents a real gearbox production and includes 21 departments (with 12 types) and eight different products which are combined in an assembly department to an end product. This example provides a solid data basis for the layout planning problem and the simulation model. All necessary information like the transport matrix, processing sequences, processing times, set up times and production rates are given in [18]. For the convenience of the reader we present them here, where we only specify the transport amount between the single types. We have $m = 12$ types with multiplicities $a_1 = a_2 = a_4 = a_8 = a_{10} = a_{11} = a_{12} = 1$, $a_5 = a_6 = a_9 = 2$, $a_3 = a_7 = 4$ and so $n = 21$ departments. The lengths (given in meter) are $l_{a_1} = 4$, $l_{a_2} = 3.4$, $l_{a_3} = 4.6$, $l_{a_4} = 4$, $l_{a_5} = 4.7$, $l_{a_6} = 3.3$, $l_{a_7} = 4.5$, $l_{a_8} = 2.3$, $l_{a_9} = 3.8$, $l_{a_{10}} = 5.2$, $l_{a_{11}} = 4$, $l_{a_{12}} = 4$ and the transport weights $c_{ij} = c_{ji}$ between types i, j are given via

$$C = \begin{pmatrix} 0 & 240 & 204 & 0 & 0 & 570 & 0 & 0 & 120 & 0 & 0 & 0 \\ 240 & 0 & 240 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 204 & 240 & 0 & 60 & 60 & 144 & 0 & 0 & 180 & 0 & 0 & 0 \\ 0 & 0 & 60 & 0 & 60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 60 & 60 & 0 & 0 & 0 & 120 & 0 & 0 & 0 & 0 \\ 570 & 0 & 144 & 0 & 0 & 0 & 570 & 0 & 0 & 0 & 144 & 0 \\ 0 & 0 & 0 & 0 & 0 & 570 & 0 & 570 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 120 & 0 & 570 & 0 & 0 & 0 & 690 & 0 \\ 120 & 0 & 180 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 240 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 144 & 0 & 690 & 240 & 60 & 0 & 720 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 720 & 0 \end{pmatrix}.$$

In optimization, we assume that the transports are equally divided among the departments of the same type. In simulation also other strategies can be tested, but we only implemented a division of the transports according to a discrete uniform distribution. A 3D illustration of some layout for this instance, where the incoming and the shipping warehouse are arranged at the borders, is given in Fig. 6.

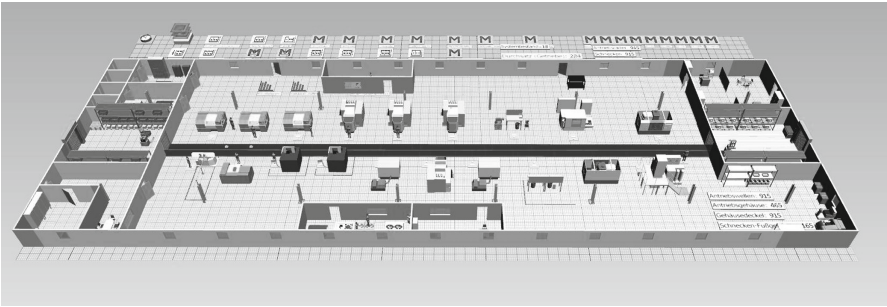


Fig. 6. 3D illustration of a DRFLP layout with 21 departments [18]. This simulation was derived using mathematical optimization. The incoming and the shipping warehouse are arranged at the left and the right border, respectively.

This example has already been taken into account in Example 1. Indeed, it contains several departments of the same type. In our computational tests we used Theorem 2 and the additional symmetry breaking constraints for departments of the same type in the same row. In Fig. 7 we show the development of the running times of our optimization approach if we enlarge the number of the departments. The instance of [18] contains departments of 12 different types. In the tests we start with 12 departments and successively add one department in each step. So, department type 3 appears twice when we consider 13 departments and three times when we consider 14 departments and so on. Figure 7 shows that the original instance with all departments could be solved in less

than 14 h, although it contains 21 departments and so five departments more than the largest DRFLP instance solved to optimality in the literature before.

In the simulation we tested the following five different solutions:

1. A solution determined according to criteria usually used in factory planning where apart from the transport weights one had a special look at the linearity of the flows. The heuristic of Schmigalla [16, 20] was applied and afterwards the solution was improved by hand. Incoming and shipping warehouse were arranged at the border.
2. A solution determined according to criteria usually used in factory planning where all departments of a type were interpreted as one big block and then these blocks were arranged. The number of blocks that had to be arranged is smaller than the total number of departments. So it was easier to build this layout by hand. Incoming and shipping warehouse were arranged at the border.
3. We used our mathematical DRFLP model for deriving a solution but as it is often done in practice incoming and shipping warehouse were arranged at the border (see end of Sect. 2). A 3D visualization of this warehouse can be found in Fig. 6.
4. We used our mathematical DRFLP model for deriving a solution and the incoming and shipping warehouse were arranged at the border. Additionally, all departments of the same type were interpreted as one big block and then these 12 blocks were arranged.
5. Solution derived using our mathematical DRFLP model with arbitrary position of all departments as well as of the incoming and the shipping warehouse (in our model these are departments, too). This approach was also used for deriving the results in Fig. 7.

In all five simulations we manufactured 36000 end products and determined afterwards the average distance of each single product and the total distance traveled. The results can be found in Table 1. The end product, which is obtained by combining all eight products in an assembly department, is denoted as product 9. The second column in Table 1 shows the number of transports (“Trans”) needed for each product 1, . . . , 9, and the next ten columns show the distances for the five simulation variants where the left column (“Single”) for each type contains the information on the average transport distance of each single product and the right column the total distance (“Total”) traveled for all products of the same type.

The results show that the use of our optimization model allows to improve the solution significantly in comparison to the solution determined by hand, especially if we do not restrict the position of the warehouses (Layout 5). Comparing Layout 1 and Layout 3, where the two warehouses have been fixed to the border, the solution obtained by optimization is better than the solution obtained by applying the heuristic of Schmigalla followed by some improvement steps by the factory planners. But even the optimized solution with blocks and fixed border, illustrated in Layout 4, is better than the solutions determined

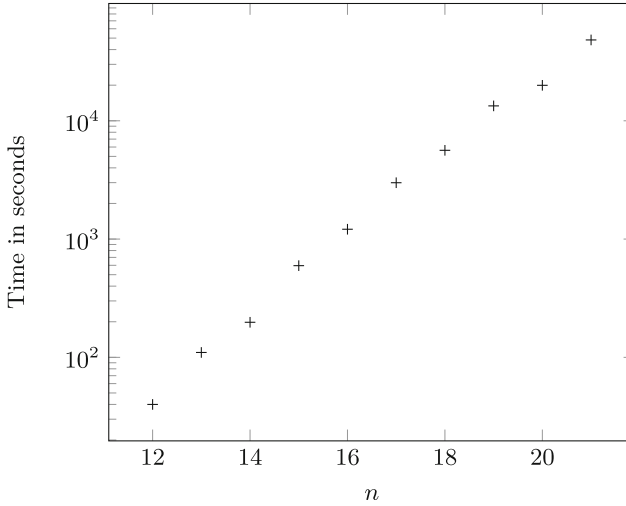


Fig. 7. Running times in seconds for variations of some realistic instance from [18]. We start with 12 different department types and 12 departments and we add departments successively according to the following order of the number of departments of each of the 12 types: 1 1 4 1 2 2 4 1 2 1 1 1. The largest instance contains 21 departments.

Table 1. Visualization of the results of the five simulations for our test case. In this production all eight products are combined to an end product, denoted as product 9, see column “P”. The entry “Total” in the first column refers to the total distance traveled in each of the five layouts. Note that using our approach from mathematical optimization with arbitrary department positions reduces the total distances significantly in comparison with the other four variants.

| P | Trans | Layout 1 | | Layout 2 | | Layout 3 | | Layout 4 | | Layout 5 | |
|-------|-------|----------|---------|----------|---------|----------|----------|----------|----------|----------|---------|
| | | Single | Total | Single | Total | Single | Total | Single | Total | Single | Total |
| 1 | 480 | 37.90 | 18192.0 | 38.70 | 18576.0 | 28.70 | 13776.00 | 29.15 | 13992.00 | 16.31 | 7828.8 |
| 2 | 90 | 37.90 | 3411.0 | 38.70 | 3483.0 | 28.70 | 2583.00 | 29.15 | 2623.50 | 16.31 | 1467.9 |
| 3 | 144 | 38.50 | 5544.0 | 44.85 | 6458.4 | 29.32 | 4222.08 | 35.91 | 5171.04 | 30.75 | 4428.0 |
| 4 | 60 | 37.90 | 2274.0 | 37.60 | 2256.0 | 28.70 | 1722.00 | 29.15 | 1749.00 | 39.60 | 2376.0 |
| 5 | 30 | 65.60 | 1968.0 | 71.90 | 2157.0 | 69.20 | 2076.00 | 70.40 | 2112.00 | 66.51 | 1995.3 |
| 6 | 60 | 55.96 | 3357.6 | 49.03 | 2941.8 | 55.66 | 3339.60 | 54.18 | 3250.80 | 54.27 | 3256.2 |
| 7 | 30 | 66.12 | 1983.6 | 71.89 | 2156.7 | 66.42 | 1992.60 | 70.90 | 2127.00 | 66.51 | 1995.3 |
| 8 | 120 | 37.90 | 4548.0 | 37.60 | 4512.0 | 28.70 | 3444.00 | 29.15 | 3498.00 | 39.00 | 4680.0 |
| 9 | 720 | 2.00 | 1440.0 | 2.00 | 1440.0 | 10.80 | 7776.00 | 11.20 | 8064.00 | 0.00 | 0.0 |
| Total | | | 42718.2 | | 43980.9 | | 40931.28 | | 42587.34 | | 28027.5 |

by hand, illustrated in Layout 2. A visualization of all layouts can be found in Fig. 8, where blocks consisting of departments of the same type are highlighted with lines in bold type.

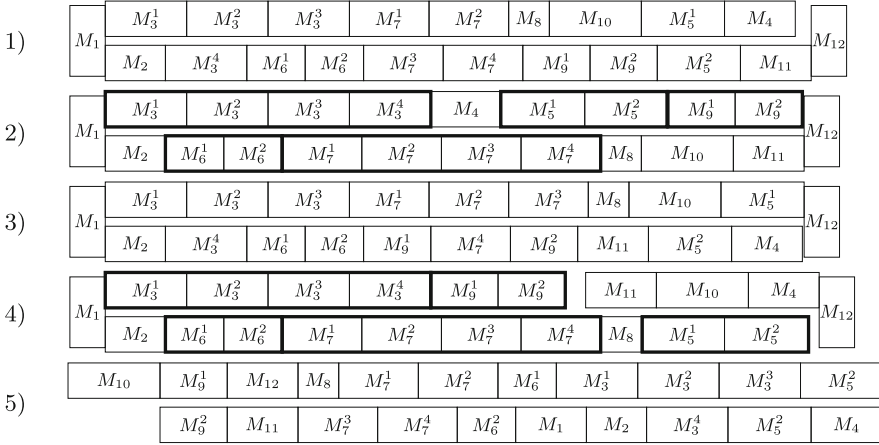


Fig. 8. Visualization of the five layouts tested in the simulations. These five layouts were constructed by the five variants stated above, i. e., Layout 1 and Layout 2 were derived according to general approaches used in factory planning and Layout 3, Layout 4 and Layout 5 were derived using the optimization model. The last layout shows an optimal solution if the position of none of the departments is restricted.

In the simulation the different products that are produced are considered separate, but in the mathematical models the transport weights are based on aggregated information for the transports of all products. Studying the results and indicators provided by the simulation of the five layout variants we realized that the time needed for the transport processes of different products can vary significantly. Next we show how to overcome this situation by adapting our optimization model.

5.2 Manufacturing Different Products

Let us assume that we manufacture an amount of different products. Let P denote the set of products and h_p be the desired number of product $p \in P$. Every product has its own transport matrix C^p . The ij -th entry of the matrix C^p denotes the transport weights between department i and j for producing product p . The transport weight matrix C that we used before is built on the sum of the transport matrices times the associated desired number of products, i. e., $C = \sum_{p \in P} h_p \cdot C^p$. Our aim is now to investigate the influence of single products to the whole production.

Definition 2. For a product $p \in P$ we define the transport distance as

$$\sum_{\substack{i,j \in [n] \\ i < j}} c_{ij}^p d_{ij},$$

where c_{ij}^p is the ij -th entry of the matrix C^p .

The simulation showed that these transport distances might be rather high in an optimal layout if the number of products of this type is small in comparison to the others. But high transport distances can increase the cycle time. So, for a smooth production we want to bound the transport distances associated to single products. Therefore, we present two possibilities: At first we can restrict this by an upper bound on the transport distance.

A second way is to set up a desired distance $d_p \in \mathbb{R}_+$ for the transport distance for each single product $p \in P$. Of course, the desired transport distance depends on the amount of products h_p for $p \in P$. If this value is exceeded, we want to penalize this with a quadratic function \tilde{f} which is later approximated by some piecewise linear function. Let $a \geq d_p$ be the highest possible transport distance of product $p \in P$. Then we set $\tilde{f}: [0, a] \rightarrow [0, \tilde{f}(a)]$ such that

$$\tilde{f}(x) = \begin{cases} d_p - x, & x < d_p, \\ 0, & x = d_p, \\ (x - d_p)^2, & x > d_p. \end{cases}$$

To avoid a non-linear objective function, we approximate \tilde{f} on the interval $[d_p, a]$ with a piecewise linear, continuous, convex function f . Therefore, we use linear interpolation [6]. Let a product $p \in P$ and points h_1, \dots, h_m be given with $m \geq 1$ and $h_i \geq d_p$ sorted in ascending order for $i \in [m]$. Then we compute a linear approximation of \tilde{f} between the points $(h_i, (h_i - d_p)^2)$ and $(h_{i+1}, (h_{i+1} - d_p)^2)$ for $i = 1, \dots, m - 1$. The resulting function f is piecewise linear and can be written as $f(x) = \max_{i=1, \dots, m} (a_i)^T x + b_i$ for $a_i, b_i \in \mathbb{R}$ and $i \in [m]$. We add the following term to the objective function of our model (1)–(13)

$$h_p \cdot f \left(\sum_{i,j \in [n], i < j} c_{ij}^p d_{ij} \right). \quad (15)$$

This term can be linearized by replacing (15) with

$$h_p \cdot t$$

in the objective function and adding the constraints

$$a_i^T x + b_i \leq t, \quad i \in [m].$$

We may set up such a penalty function for every product $p \in P$.

6 Conclusion and Future Work

In this paper we presented a new approach that allows combining mathematical optimization and simulation in facility layout planning. We concentrated on the Double Row Facility Layout Problem. In contrast to the literature we showed

how the existing models can be extended in order to cover several aspects important in practice. To evaluate the facility layout we used simulation to determine further key performance indicators. If problems occur, the mathematical models have to be adapted appropriately. For the first time we were able to solve an instance with 21 departments to optimality in reasonable time. We compared our mathematical model with classic methods from factory planning and we could reduce the total transport distance significantly, especially by using arbitrary positions for the warehouses.

It remains for future work to include more aspects in the mathematical models. One important topic is the treatment of asymmetric transport weights in combination with input and output positions of the departments that might not lie in the center of the department. Furthermore, due to safety restrictions or quality requirements certain clearance conditions between departments have to be satisfied. From the mathematical point of view it is interesting to further study the polyhedral structure of the associated models as well as to develop new (mixed-) integer programming models that combine the assignment of the department to the rows as well as the positioning of the departments in each row. The hope would be that intelligent branching orders can reduce the overall running time. Apart from the Double Row Facility Layout Problem it seems worth to consider more complex path structures in the shape of a T or an X or along some closed path.

A further important goal is the inclusion of robustness aspects because the facility layout decision has an impact for several years, but the production program, which is the basis for the transport weights, might change. Apart from an extension of the mathematical models, simulation allows testing different scenarios for future production programs [5] easily. Using simulation, different layout variants can be evaluated with regard to changing production requirements [10]. So, a flexible and adaptable production layout can be identified.

Acknowledgement. This work was supported by the Simulation Science Center Clausthal-Göttingen.

References

1. Altinkilinc, M.: Simulation-based layout planning of a production plant. In: Proceedings of the 36th Conference on Winter simulation, pp. 1079–1084 (2004)
2. Amaral, A.R.: A new lower bound for the single row facility layout problem. *Discrete Appl. Math.* **157**(1), 183–190 (2009)
3. Amaral, A.R.: Optimal solutions for the double row layout problem. *Optim. Lett.* **7**(2), 407–413 (2013)
4. Anjos, M.F., Vieira, M.V.: Mathematical optimization approaches for facility layout problems: the state-of-the-art and future research directions. *Eur. J. Oper. Res.* **261**(1), 1–16 (2017)
5. Arnhold, D.: Digitale Produktionsprozessplanung variantenreicher Produkte unter Berücksichtigung von intervallbasierten Eingangsdaten. Shaker (2013)
6. Blu, T., Thévenaz, P., Unser, M.: Linear interpolation revitalized. *IEEE Trans. Image Process.* **13**(5), 710–719 (2004)

7. Bracht, U., Fischer, A., Krüger, T.: Mathematische Anordnungsoptimierung und Simulation - ein kombinierter Ansatz zur Fabriklayoutplanung. *Werkstattstechnik online* **107**(4), 200–207 (2017)
8. Bracht, U., Geckler, D., Wenzel, S.: *Digitale Fabrik: Methoden und Praxisbeispiele*. Springer, Deutschland (2011). <https://doi.org/10.1007/978-3-662-55783-9>
9. Chung, J., Tanchoco, J.: The double row layout problem. *Int. J. Prod. Res.* **48**(3), 709–727 (2010)
10. Dombrowski, U., Ernst, S.: Scenario-based simulation approach for layout planning. *Procedia CIRP* **12**, 354–359 (2013)
11. Drira, A., Pierreval, H., Hajri-Gabouj, S.: Facility layout problems: a survey. *Ann. Rev. Control* **31**(2), 255–267 (2007)
12. Fischer, A., Fischer, F., Hungerländer, P.: New exact approaches to row layout problems. Technical report 2015–11, Preprint-Reihe, Institut für Numerische und Angewandte Mathematik, Georg-August Universität Göttingen (2015)
13. Hungerländer, P., Rendl, F.: A computational study and survey of methods for the single-row facility layout problem. *Comput. Optim. Appl.* **55**(1), 1–20 (2013)
14. Hungerländer, P., Rendl, F.: Semidefinite relaxations of ordering problems. *Math. Program.* **140**(1), 77–97 (2013)
15. IBM ILOG CPLEX Optimization Studio 12.7 (2017)
16. Kettner, H., Schmidt, J., Greim, H.-R.: *Leitfaden der systematischen Fabrikplanung*. Hanser München (1984)
17. Murray, C.C., Smith, A.E., Zhang, Z.: An efficient local search heuristic for the double row layout problem with asymmetric material flow. *Int. J. Prod. Res.* **51**(20), 6129–6139 (2013)
18. Prêt, U.: Komplexes Fallbeispiel Teilefertigungs- und Montageprojekt “Schneckengetriebeproduktion” (2017). <http://www.uwe-pret.de/getriebe.pdf>. Accessed 06 Apr 2018
19. Rooks, T.: *Rechnergestützte Simulationsmodellgenerierung zur dynamischen Absicherung der Montagelogistikplanung bei der Fahrzeugneutypplanung im Rahmen der digitalen Fabrik*. Shaker (2009)
20. Schmigalla, H.: *Fabrikplanung: Begriffe und Zusammenhänge*. Hanser Verlag (1995)
21. Scholz, D., Petrick, A., Domschke, W.: STaTS: a slicing tree and tabu search based heuristic for the unequal area facility layout problem. *Eur. J. Oper. Res.* **197**(1), 166–178 (2009)
22. Simmons, D.M.: One-dimensional space allocation: an ordering algorithm. *Oper. Res.* **17**(5), 812–826 (1969)
23. Tecnomatrix Plant Simulation. Siemens PLM Software (2017)
24. Zhang, Z., Murray, C.C.: A corrected formulation for the double row layout problem. *Int. J. Prod. Res.* **50**(15), 4220–4223 (2012)
25. Zuo, X., Murray, C.C., Smith, A.E.: Solving an extended double row layout problem using multiobjective tabu search and linear programming. *IEEE Trans. Autom. Sci. Eng.* **11**(4), 1122–1132 (2014)



Interactive Multiobjective Robust Optimization with NIMBUS

Yue Zhou-Kangas^(✉), Kaisa Miettinen, and Karthik Sindhya

University of Jyväskylä, Faculty of Information Technology,
P.O.Box 35, Agora, 40014 University of Jyväskylä, Finland
yue.y.zhou-kangas@jyu.fi

Abstract. In this paper, we introduce the MuRO-NIMBUS method for solving multiobjective optimization problems with uncertain parameters. The concept of set-based minmax robust Pareto optimality is utilized to tackle the uncertainty in the problems. We separate the solution process into two stages: the pre-decision making stage and the decision making stage. We consider the decision maker's preferences in the nominal case, i.e., with the most typical or undisturbed values of the uncertain parameters. At the same time, the decision maker is informed about the objective function values in the worst case to support her/him to make an informed decision. To help the decision maker to understand the behaviors of the solutions, we visually present the objective function values. As a result, the decision maker can find a preferred balance between robustness and objective function values under the nominal case.

Keywords: Multiple criteria decision making · Uncertainty
Robustness · Interactive methods · Robust Pareto optimality

1 Introduction

Many real-life optimization problems involve multiple (conflicting) objectives. Multiobjective optimization methods (see e.g., [11, 18]) solve these problems by optimizing the conflicting objectives simultaneously. For multiobjective optimization problems, there usually is a set of mathematically equally good solutions with different trade-offs among the multiple objectives. These solutions are called Pareto optimal solutions. In most cases, only one Pareto optimal solution is chosen as the final solution to implement. This solution is usually found by utilizing preferences of a decision maker, who is an expert in the problem domain.

Different types of methods can be identified depending on the role of the decision maker [11]. In interactive multiobjective optimization methods [3], the decision maker actively directs the solution process towards a most preferred solution by iteratively specifying her/his preferences. With an active involvement, which is not possible in other types of methods, the decision maker can

gradually learn about the problem and its feasible solutions as well as how attainable her/his preferred solutions are. In this way, interactive methods can best support the decision maker to find the most preferred solution.

In addition to multiple objectives, the presence of uncertainty in real-life optimization problems should be considered due to imprecise data, uncertain operation environments, and uncertain future developments, etc. The uncertainty can be reflected in parameters or decision variables in problem formulations. In this paper, we concentrate on problems with uncertain parameters in objective functions. With different realizations of uncertain parameters, the corresponding outcomes (i.e., objective function values) are different.

On one hand, without considering the uncertainty, the outcome corresponding to a deterministic Pareto optimal solution can become very bad when the uncertain parameters realize differently. Many robustness concepts have been defined for multiobjective optimization problems (see e.g., [9, 19]). They guarantee the immunity of solutions to uncertainty by transforming uncertain problems to deterministic ones with respect to the worst case. On the other hand, the outcomes in the nominal case are very important for the decision maker, because the nominal case describes the most typical behavior of uncertain parameters. In addition, the robustness and quality of solutions, i.e., the outcome in the nominal case, usually conflict with each other [1]. In other words, objective function values of a robust Pareto optimal solution are usually not as good as those of a deterministic Pareto optimal solution in the nominal case.

When considering uncertainty, the decision maker faces the challenge of making a decision with respect to different possible outcomes because of different realizations of uncertain parameters. Considering multiple possible realizations simultaneously can be too challenging for the decision maker. In addition, it is desirable for the decision maker to find a preferred balance between robustness and quality of the solutions. With the help of multiobjective robust optimization, we can guarantee the robustness of solutions by finding the best solutions with respect to the worst case but at the same time, the decision maker needs support to find a most preferred balance between robustness and quality of solutions.

In the literature, most research efforts have been devoted to different definitions of robust Pareto optimality and only a few solution methods have been developed (e.g., in [5, 10]). In addition, in [2], necessary and sufficient conditions for scalarizing functions with some special properties are discussed, which can be used to transform a multiobjective optimization problem to a single-objective one. In [7, 8, 14, 15], interactive methods have been utilized to find a final solution for multiobjective optimization problems with uncertainty.

In [7, 8], a robust version of the augmented weighted Chebyshev method [17] was developed for multiobjective linear optimization problems by extending the concept of the budget of uncertainty [1] to multiobjective optimization problems. Uncertainty was tackled in a so-called all-in-one approach in [14], where the decision maker considers all possible realizations of uncertain parameters simultaneously. During the solution process, the decision maker chooses the possible realizations to concentrate on and formulates her/his preferences with respect

to them. In [15], the decision maker is expected to specify weights to alter the relative importance of objectives and robustness when they are combined to formulate a single-objective optimization problem.

In this paper, we develop an interactive method called MuRO-NIMBUS to better support the decision maker. The MuRO-NIMBUS method integrates the concept of set-based minmax robustness [5] into the NIMBUS framework, which to the best of our knowledge, is the first interactive method for supporting a decision making to find set-based minmax robust Pareto optimal solutions.

The properties of desirable interactive methods were summarized in [16] in terms of understandability, easiness to use, and features of being supportive. In order to ensure those properties in MuRO-NIMBUS, we first guarantee the robustness of solutions by utilizing the set-based minmax robust Pareto optimality to find a set of best possible solutions in the worst case. For this step, we develop a robust achievement scalarizing function approach, which can also be used independently. Then we incorporate the preferences of the decision maker to find a solution corresponding to a most preferred outcome in the nominal case. At the same time, the decision maker is informed about the worst possible values. In order to support the decision maker to understand the solution in terms of its objective function values in the nominal case and the objective function values in the worst case, we augment the value path visualization (see e.g., [6]) to visually present different types of information. In this way, we can support the decision maker to grasp a total balance in the robustness and quality of solutions during the solution process.

By applying MuRO-NIMBUS, the decision maker is not expected to consider all possible realizations of the uncertain parameters simultaneously as in [14]. Unlike in [7, 8] where solutions once discarded cannot be recovered, the decision maker can move freely from one robust Pareto optimal solution to another. Instead of providing preferences as weights which do not have concrete meanings as in [15], MuRO-NIMBUS allows the decision maker to concretely consider the objective function values of a more desired solution.

The rest of the paper is organized as follows: in the next section, we introduce some basic concepts. In Sect. 3, we introduce MuRO-NIMBUS. We simulate the solution process of a multiobjective ship design problem as a numerical example in Sect. 4 to demonstrate the application of the new method. Finally, we conclude the paper in Sect. 5.

2 Basic Concepts

2.1 Deterministic Multiobjective Optimization

A deterministic multiobjective optimization problem is of the form

$$\begin{aligned} & \text{minimize or maximize} && \{f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\} \\ & \text{subject to} && \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{1}$$

involving objective functions (objectives) $f_i : \mathcal{X} \rightarrow \mathbb{R}$ to be simultaneously optimized, where $1 \leq i \leq k$ and $k \geq 2$. Objective vectors $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))^T$

consist of objective function values which are the images of decision vectors $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$. Decision vectors belong to the nonempty feasible set $\mathcal{X} \subset \mathbb{R}^n$ and their components are called decision variables. In this paper, we refer to decision vectors as solutions and objective vectors as outcomes or objective function values of solutions. For two feasible solutions, we say a solution dominates the other when the value of at least one of the objectives is better and others are at least as good as that of the other. For simplicity, we assume that the objective functions are to be minimized.

Definition 1. A solution $\mathbf{x}^* \in \mathcal{X}$ is said to be Pareto optimal or efficient if there does not exist another solution $\mathbf{x} \in \mathcal{X}$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \dots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one j .

With the help of the nonnegative ordering cone $\mathbb{R}_{\geq}^k = \{\mathbf{z} \in \mathbb{R}^k \mid z_i \geq 0 \text{ for } i = 1, \dots, k\}$, we say that \mathbf{x}^* is Pareto optimal if there does not exist $\mathbf{x} \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}) \in \mathbf{f}(\mathbf{x}^*) - \mathbb{R}_{\geq}^k$. We refer to the set of Pareto optimal solutions as the Pareto optimal set.

For (1), the set of Pareto optimal solutions usually contains more than one element. For the decision maker, it is often useful to know the ranges of the objective function values in the Pareto optimal set. The ranges are given by the ideal objective vector $\mathbf{z}^* = (z_1^*, \dots, z_k^*)^T$ and the nadir objective vector $\mathbf{z}^{nad} = (z_1^{nad}, \dots, z_k^{nad})^T$. The ideal objective vector is formed by individual optima of each objective function in the feasible set. For computational reasons, we use the utopian objective vector \mathbf{z}^{**} , which is strictly better than \mathbf{z}^* . In practice, z_i^{**} is set as $z_i^* - a$ for $i = 1, \dots, k$, where $a > 0$ is a small scalar. The nadir objective vector, which represents the worst objective function values, can be approximated for example by a so-called pay-off table (see [11] for further details). If the objective function values have different magnitudes, \mathbf{z}^{nad} and \mathbf{z}^{**} can be used to normalize them for computing purposes.

For calculating Pareto optimal solutions, one approach is to scalarize, i.e., to formulate a single objective optimization problem such that its optimal solution is a Pareto optimal solution for (1). In this, a single objective solver which is appropriate for the characteristics of the problem must be used. The achievement scalarizing function [20] is one of the widely used scalarizing functions. In this paper, we consider the achievement scalarizing function of the following form:

$$\begin{aligned} & \text{minimize} \quad \max_i [w_i(f_i(\mathbf{x}) - \bar{z}_i)] + \rho \sum_{i=1}^k w_i(f_i(\mathbf{x}) - \bar{z}_i) \\ & \text{subject to} \quad \mathbf{x} \in \mathcal{X}, \end{aligned} \quad (2)$$

where ρ is a small scalar binding the trade-offs, $\bar{\mathbf{z}}$ is a reference point and its component \bar{z}_i is the aspiration level which represents the desired value of the objective function f_i given by the decision maker. The positive weight vector \mathbf{w} sets a direction toward which the reference point is projected onto the Pareto optimal set.

As discussed in the literature (e.g., [3, 11, 20]), the optimal solution of (2) is a Pareto optimal solution for (1) and any Pareto optimal solution with trade-offs bounded by ρ can be found by changing $\bar{\mathbf{z}}$. The achievement scalarizing function has many advantages, for example, the reference point can be feasible or infeasible and the problem can be convex or nonconvex.

2.2 Uncertain Multiobjective Optimization Problems and Set-Based Minmax Robustness

For multiobjective optimization problems with uncertain parameters, given an uncertainty set $\mathcal{U} \subseteq \mathbb{R}^m$, the uncertain multiobjective optimization problem is given as a collection of deterministic multiobjective optimization problems:

$$\left\{ \begin{array}{l} \text{minimize } \mathbf{f}(\mathbf{x}, \boldsymbol{\xi}) \\ \text{subject to } \mathbf{x} \in \mathcal{X} \end{array} \right\}_{\boldsymbol{\xi} \in \mathcal{U}}. \quad (3)$$

Every problem in the collection is called an instance, which is characterized by a particular element $\boldsymbol{\xi} \in \mathcal{U}$. Depending on different realized values of $\boldsymbol{\xi}$, a decision vector can have different corresponding outcomes. As a result, we have a set of outcomes corresponding to a feasible decision vector. We denote the set of outcomes (i.e., the objective vectors) of a solution $\mathbf{x} \in \mathcal{X}$ for all $\boldsymbol{\xi} \in \mathcal{U}$ as $f_{\mathcal{U}}(\mathbf{x}) = \{f_{\mathcal{U}}(\mathbf{x}, \boldsymbol{\xi}) : \boldsymbol{\xi} \in \mathcal{U}\}$ as in [5].

As briefly mentioned, among all the possible realizations of uncertain parameters, the nominal case $\hat{\boldsymbol{\xi}}$ describes the most typical behavior of the uncertain parameters. It usually comes from previous experiences or the expert knowledge of the decision maker. The worst case describes the situation where the objective functions attain their worst values within \mathcal{U} . For a fixed solution $\mathbf{x} \in \mathcal{X}$, we need to solve the following problem to find the worst case:

$$\begin{array}{ll} \text{maximize} & \{f_1(\mathbf{x}, \boldsymbol{\xi}), \dots, f_k(\mathbf{x}, \boldsymbol{\xi})\} \\ \text{subject to} & \boldsymbol{\xi} \in \mathcal{U}. \end{array} \quad (4)$$

If the components of $\boldsymbol{\xi}$ do not relate to each other, there is a single worst case. If they are related to each other, there can be multiple worst cases. With the found worst case, the corresponding outcomes for the solution in question can be calculated. The worst case does not necessarily realize in practice, but the information on the outcomes provides the upper bounds of the objective function values of a solution within \mathcal{U} .

Analogously to the definition of Pareto optimality for deterministic problems, set-based minmax Pareto optimality was defined in [5] by comparing the sets of all outcomes corresponding to solutions.

Definition 2. A solution \mathbf{x}^* is a set-based minmax robust Pareto optimal solution for (3), if there does not exist another $\mathbf{x} \in \mathcal{X}$ such that $f_{\mathcal{U}}(\mathbf{x}) \subseteq f_{\mathcal{U}}(\mathbf{x}^*) - \mathbb{R}_{\geq}^k$.

In other words, a feasible solution \mathbf{x}^* is a set-based minmax robust Pareto solution if there does not exist another feasible solution \mathbf{x} such that for all

outcomes $f(\mathbf{x}, \xi) \in f_{\mathcal{U}}(\mathbf{x})$, there exists an outcome $f(\mathbf{x}^*, \xi) \in f_{\mathcal{U}}(\mathbf{x}^*)$ with $f_i(\mathbf{x}, \xi) \leq f_i(\mathbf{x}^*, \xi)$ for all $i = 1, \dots, k$. We apply this concept in MuRO-NIMBUS to be introduced. With this concept, the decision maker can understand that for all set-based minmax robust Pareto optimal solutions, there does not exist a feasible solution with better objective function values in every possible realization of the uncertain parameters.

By interpreting the supremum of a set as the set itself, the robust counterpart of (3) which transforms (3) to a deterministic problem to identify robust Pareto optimal solutions is given in [5] as:

$$\begin{aligned} & \text{minimize } \sup_{\xi \in \mathcal{U}} f(\mathbf{x}, \xi) \\ & \text{subject to } \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{5}$$

Set-based minmax robust Pareto optimal solutions are the best possible solutions in the worst case because they are obtained by minimizing the suprema of the sets of outcomes. As explained earlier, finding the worst case outcomes for a fixed solution $\mathbf{x} \in \mathcal{X}$ requires solving a multiobjective optimization problem with objectives to be maximized as (4). The notation sup in (5) denote the supreme of the outcome sets which is used to identify the worst case outcomes. For simplicity, in what follows, we refer to set-based minmax robust Pareto optimal solutions as robust Pareto optimal solutions.

2.3 Interactive Multiobjective Optimization

As mentioned, in interactive methods, the decision maker directs the solution process towards a most preferred solution by iteratively specifying her/his preferences. A typical solution process (e.g., [3]) starts by presenting a Pareto optimal solution to the decision maker. If the decision maker is satisfied, the final solution is found. If the decision maker is not satisfied, (s)he is expected to specify preferences for a more desired solution. Based on the preferences, a new Pareto optimal solution which satisfies the preferences best is found and presented to her/him. The solution process continues until the decision maker finds a most preferred solution.

NIMBUS [11, 13] is a family of classification-based interactive methods. In NIMBUS, the decision maker can classify the objectives to indicate what kind of objective vector would be more preferred than the current one. The objective functions can be assigned to up to five different classes including:

- $I^<$ for those to be improved (i.e., decreased in case of minimizing, increased in case of maximizing),
- I^{\leq} for those to be improved until some desired aspiration level \hat{z}_i ,
- $I^=$ for those that are satisfactory at their current level,
- I^{\geq} for those that may be impaired till a bound ϵ_i , and
- I^{\diamond} for those that are temporarily allowed to change freely.

If aspiration levels or bounds are used, the decision maker is expected to provide them. If the classification is feasible, i.e., the decision maker allows at least one of the objectives to be impaired to improve some objectives, a scalarizing problem is solved to find a new Pareto optimal solution reflecting the preferences. In the so-called synchronous NIMBUS method, up to four different solutions can be found in each iteration by solving different scalarizing problems. Since we have to consider robustness and quality of the solutions, we limit the cognitive load to the consideration of only one solution at a time. We will return later to the variant of the NIMBUS scalarizing problems we use in MuRO-NIMBUS.

MuRO-NIMBUS inherits the advantage of classifying the objectives. First, classification can remind the decision maker that it is not possible to improve all objective function values at the same time but impairment in some objective(s) must be allowed. Second, the decision maker deals with objective function values and (s)he does not need to connect different types of information. Instead, (s)he needs to know what kind of changes (s)he desires for a new solution.

3 MuRO-NIMBUS

In this section, we introduce MuRO-NIMBUS. To be able to present it, we first introduce some building blocks that we need for designing the method.

3.1 Building Blocks of MuRO-NIMBUS

As a building block of MuRO-NIMBUS, we first present the robust version of (2). Based on it, we introduce the robust achievement scalarizing function (ASF) approach to calculate a set of robust Pareto optimal solutions.

Based on the concept of robust Pareto optimality and the robust counterpart as introduced in Sect. 2, the robust version of (2) can be formulated as:

$$\begin{aligned} & \text{minimize} && \sup_{\xi \in \mathcal{U}} \max_i [w_i(f_i(\mathbf{x}, \xi) - \bar{z}_i)] + \rho \sum_{i=1}^k w_i(f_i(\mathbf{x}, \xi) - \bar{z}_i) \\ & \text{subject to} && \mathbf{x} \in \mathcal{X} \text{ for all } \xi \in \mathcal{U}. \end{aligned} \quad (6)$$

Just like (2), the robust version involves a reference point and a weight vector. We now prove the sufficient condition of the robust Pareto optimality:

Theorem 1. *Given an uncertain multiobjective optimization problem (3), if \mathbf{x}^* is an optimal solution to (6) for some $\bar{\mathbf{z}}$ and \mathbf{w} , and $\max_{\xi \in \mathcal{U}} f_i(\mathbf{x}, \xi)$ exists for all $\mathbf{x} \in \mathcal{X}$ and for all $i = 1, \dots, k$, then \mathbf{x}^* is a robust Pareto optimal solution for (3).*

Proof. Assume that \mathbf{x}^* is not a robust Pareto optimal solution for (3). Then there exists $\mathbf{x}' \in \mathcal{X}$ such that $f_{\mathcal{U}}(\mathbf{x}') \subseteq f(\mathbf{x}^*) - \mathbb{R}_{\geq}^k$. Based on Lemma 3.4 in [5], for all $\xi \in \mathcal{U}$, there exists $\eta \in \mathcal{U}$ such that $\bar{f}_i(\mathbf{x}', \xi) \leq f_i(\mathbf{x}^*, \eta)$ for $i = 1, \dots, k$ and for at least one i the strict inequality holds. Since $w_i > 0$,

we have $\max_i [w_i(f_i(\mathbf{x}', \boldsymbol{\xi}) - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}', \boldsymbol{\xi}) - \bar{z}_i) < \max_i [w_i(f_i(\mathbf{x}^*, \boldsymbol{\eta}) - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}^*, \boldsymbol{\eta}) - \bar{z}_i)$, where for all $\boldsymbol{\xi} \in \mathcal{U}$ there exists a $\boldsymbol{\eta} \in \mathcal{U}$ which satisfy the inequality. Further, we know that $\max_{\boldsymbol{\xi} \in \mathcal{U}} \max_i [w_i(f_i(\mathbf{x}', \boldsymbol{\xi}) - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}', \boldsymbol{\xi}) - \bar{z}_i) < \max_{\boldsymbol{\eta}' \in \mathcal{U}} \max_i [w_i(f_i(\mathbf{x}^*, \boldsymbol{\eta}') - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}^*, \boldsymbol{\eta}') - \bar{z}_i)$. So $\max_{\boldsymbol{\xi}' \in \mathcal{U}} \max_i [w_i(f_i(\mathbf{x}', \boldsymbol{\xi}') - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}', \boldsymbol{\xi}') - \bar{z}_i) < \max_{\boldsymbol{\eta}' \in \mathcal{U}} \max_i [w_i(f_i(\mathbf{x}^*, \boldsymbol{\eta}') - \bar{z}_i)] + \rho \sum_{i=1}^k (f_i(\mathbf{x}^*, \boldsymbol{\eta}') - \bar{z}_i)$. This contradicts with the assumption that \mathbf{x}^* is the optimal solution for (6). So \mathbf{x}^* is a robust Pareto optimal solution for (3).

This result agrees with the sufficient condition presented in Theorem 4.4 in [2] for strongly increasing scalarizing functions, which states that the optimal solution of a strongly increasing scalarizing function is set-based minmax Pareto optimal to (3). In [2], the detailed proof was omitted. The necessary condition and the proof for strictly increasing scalarizing function are given in Theorem 4.1 in [2]. As a strongly increasing scalarizing function, (6) is also a strictly increasing scalarizing function. For the properties of strongly and strictly increasing scalarizing function see [2, 20]. Based on (6), we introduce the robust ASF approach with (3) as the input to calculate a set of robust Pareto optimal solutions X_{rpo} as the output:

Step 1. Set $X_{rpo} = \emptyset$ and generate a set of reference points \mathcal{Z} .

Step 2. If $\mathcal{Z} = \emptyset$, stop.

Step 3. Choose a $\bar{\mathbf{z}} \in \mathcal{Z}$, and set $\mathcal{Z} = \mathcal{Z} \setminus \{\bar{\mathbf{z}}\}$.

Step 4. Find an optimal solution \mathbf{x}^* to (6) using $\bar{\mathbf{z}}$ as the reference point and set \mathbf{w} accordingly, e.g., $w_i = \frac{1}{z_i^{**} - \bar{z}_i}$, where \mathbf{z}^{**} is the utopian objective vector. Set $X_{rpo} = X_{rpo} \cup \{\mathbf{x}^*\}$.

Step 5. Go to step 2.

In the robust ASF approach, we alter $\bar{\mathbf{z}}$ and set \mathbf{w} accordingly for efficiently gaining a good representative set of robust Pareto optimal solutions X_{rpo} . When we evaluate their outcomes in the nominal case $\hat{\boldsymbol{\xi}}$, some of them can be dominated. We should only present nondominated solutions to the decision maker. So we refer to the robust Pareto optimal solutions whose corresponding outcomes are nondominated as nominal nondominated robust Pareto optimal solutions: a robust Pareto optimal solution \mathbf{x}^* is a nominal nondominated robust Pareto optimal solution if there does not exist another $\mathbf{x} \in X_{rpo}$ such that $\mathbf{f}(\mathbf{x}, \hat{\boldsymbol{\xi}}) \in \mathbf{f}(\mathbf{x}^*, \hat{\boldsymbol{\xi}}) - \mathbb{R}_{\geq}^k$.

For finding a nondominated robust Pareto optimal solution based on a NIMBUS classification, we solve a variant of the synchronous NIMBUS scalarizing problem presented in [13]:

$$\begin{aligned}
& \text{minimize} && \max_{\substack{i \in I^< \\ j \in I^{\leq}}} [w_i(f_i(\mathbf{x}, \hat{\boldsymbol{\xi}}) - z_i^*), w_j(f_j(\mathbf{x}, \hat{\boldsymbol{\xi}}) - \hat{z}_j)] + \rho \sum_{i=1}^k w_i f_i(\mathbf{x}, \hat{\boldsymbol{\xi}}) \\
& \text{subject to} && \mathbf{x} \in X_{rpo} \\
& && f_i(\mathbf{x}, \hat{\boldsymbol{\xi}}) \leq f_i(\mathbf{x}^c, \hat{\boldsymbol{\xi}}) \text{ for all } i \in I^< \cup I^{\leq} \cup I^=, \\
& && f_i(\mathbf{x}, \hat{\boldsymbol{\xi}}) \leq \epsilon_i \text{ for all } i \in I^{\geq},
\end{aligned} \tag{7}$$

where $I^<$, $I^=$, I^{\geq} , I^{\leq} , and I^{\diamond} represent the corresponding classes of objectives and \mathbf{x}^c is the current solution.

Proposition 1. *The solution of (7) is a nominal nondominated robust Pareto optimal solution for problem (3).*

Proof. Problem (7) is equivalent to a deterministic problem in the nominal case with the feasible set X_{rpo} . The proof that the solution of (7) is Pareto optimal for deterministic problems was given in [13]. Thus it fulfills the requirements to be a nominal nondominated robust Pareto optimal solution.

3.2 MuRO-NIMBUS

Based on the building blocks discussed above, we introduce MuRO-NIMBUS which can support the decision maker to find a most preferred solution for (3). We first discuss the idea of MuRO-NIMBUS in general. Then we present its steps followed by a discussion on the technical details of each step.

As mentioned before, e.g., in [1], the robustness and the quality of solutions usually conflict with each other. If the decision maker is not willing to sacrifice some quality to gain robustness, we can solve (3) in the nominal case as a deterministic problem. On the other hand, if the decision maker is willing to make some sacrifice to gain robustness, (s)he prefers to have a robust Pareto optimal solution by bearing the fact that its quality may not be as good as a Pareto optimal solution in the nominal case. MuRO-NIMBUS is developed for solving (3) when the decision maker is willing to sacrifice some quality to gain robustness. Because outcomes in the nominal case are very important for the decision maker and robustness of solutions can be guaranteed by finding best possible solutions in the worst case, we have three tasks during the solution process.

First, we need to guarantee the robustness of the solutions. Second, the nominal case has to be considered in terms of corresponding outcomes of solutions to satisfy the decision maker's preferences as much as can. Third, to help the decision maker to make an informed decision, corresponding outcomes in the worst case should be found. It is not possible to guarantee the robustness and consider two different kinds of realizations of the uncertain parameters at the

same time during the solution process. So we separate the consideration into two stages in MuRO-NIMBUS: pre-decision making and decision making.

In the pre-decision making stage, we first concentrate on robustness, i.e., finding a set of robust Pareto optimal solutions. Then we consider the preferences of the decision maker in the decision making stage. Specifically, we support the decision maker to direct the solution process towards a most preferred robust Pareto optimal solution among the ones calculated. As a result, the final solution selected is robust Pareto optimal and at the same time corresponding to a most preferred outcome by the decision maker in the nominal case. In addition, the decision maker is informed of the outcome in the worst case.

We should be aware of the necessity of asking the decision maker whether (s)he is willing to sacrifice some quality to gain robustness before the solution process of a problem. Now we can present the overall algorithm of MuRO-NIMBUS as follows:

1. Pre-decision making

- (a) Calculate the set X_{rpo} with the robust ASF approach. Calculate also the ideal and nadir objective vectors in the nominal case.

2. Decision making

- (a) Classify all the objectives into the class $I^<$ of the NIMBUS classification and solve (7) (by including only the first constraint) to find an initial nominal nondominated robust Pareto optimal solution \mathbf{x}^c .
- (b) Present the ideal and nadir objective vectors calculated in the nominal case to the decision maker.
- (c) Present the outcomes in the nominal and the worst cases corresponding to \mathbf{x}^c to the decision maker. If the decision maker is satisfied, \mathbf{x}^c is the final solution. Otherwise, continue.
- (d) Ask the decision maker to classify the objectives at the current solution, i.e., the outcome in the nominal case. Then solve (7) to find a new nominal nondominated solution and set it as \mathbf{x}^c and go to step 2(c).

In step 1, the presence of the decision maker is not required. We use the robust ASF approach which can handle general problems (for example, the weighted-sum method in [5] assumes the problem to be solved is convex). In addition, in robust ASF, we apply the idea from [4] to alter the reference points $\bar{\mathbf{z}}$ and set \mathbf{w} accordingly to efficiently obtain the set X_{rpo} . As for efficiently solving the scalarized problem and handling the constraints which should be fulfilled for all the possible realizations of the uncertain parameters, we discretize the uncertainty set to reformulate (6).

After step 1, we start the stage where the decision maker actively participates in the solution process. The goal is to find the most preferred solution from the set X_{rpo} by considering the corresponding outcomes in the nominal case. As an inherited advantage, MuRO-NIMBUS only requires the decision maker to classify the objectives based on the outcome of the current solution.

The decision making stage starts by calculating an initial nominal nondominated robust Pareto optimal solution. Before presenting the initial solution,

the calculated ideal and nadir objective vectors are presented to the decision maker to help her/him to have a general idea on the ranges of the values of each objective function in the nominal case. With this information, when the outcome corresponding to the initial solution in the nominal case is presented, the decision maker can have a concrete understanding on its quality. As background information, the outcome(s) in the worst case is/are also shown to the decision maker to help her/him to make an informed decision.

As a tool for presenting the solutions to the decision maker, we utilize the value path visualization (see e.g., [6]). One can also modify some other visualization methods (see e.g., [12]) for this purpose. As said, depending on the characteristics of the involved uncertainty, there can exist multiple worst cases. We indicate the information on the outcomes in the worst cases accordingly in the visualization.

Figure 1 presents the idea of calculating the worst case objective function values in the visual presentation of a solution. In the figure, we have five different realizations of the uncertain parameters and the uncertain parameters do not relate to each other. The outcome in the nominal case is presented as the value path in the figure in blue. Outcomes with other realizations are presented in grey. By solving (4), we obtain the individual maxima of each objective in the uncertainty set as the outcome in the worst case. The corresponding outcome in the worst case is marked by triangles in the figure. The same idea applies when the uncertain parameters are related to each other. Instead of single values, we get ranges of values as the outcomes in the worst cases.

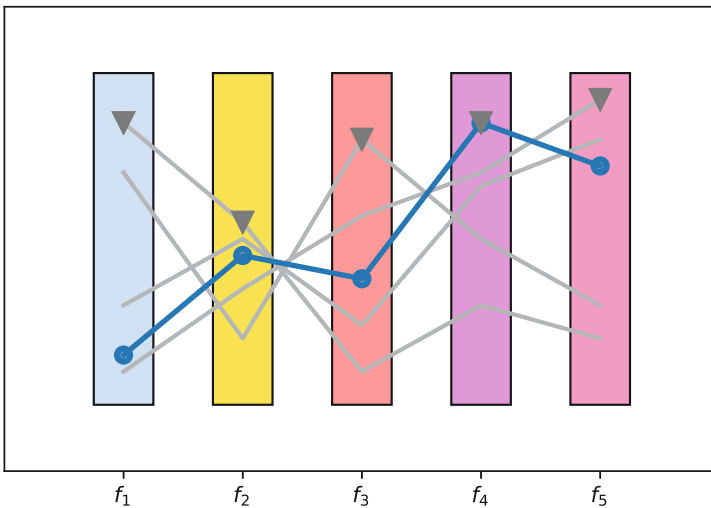


Fig. 1. Outcomes in the worst case (Color figure online)

After having seen the initial solution, the decision maker can classify the objectives into up to five classes as discussed in Sect. 2 to express her/his

preferences for a more desired solution. Based on the classification, we solve the scalarizing problem (7) to find a new nominal nondominated robust Pareto optimal solution which satisfies the classification best. The new solution is presented to the decision maker with an updated visualization. The solution process continues until the decision maker finds the most preferred nominal nondominated robust Pareto optimal solution.

4 Numerical Example

In this section, we simulate the solution process of the multiobjective ship design problem [21] to demonstrate the application of MuRO-NIMBUS. The problem has three objectives: minimizing the transportation cost, minimizing the light ship mass and maximizing the annual cargo. A detailed presentation of the problem in the deterministic case is in the Appendix A of [21].

The problem was originally studied as a deterministic problem. In the uncertain version studied in this paper, we consider two parameters which stem from given intervals: the fuel price and the round trip mileage. The fuel price affects the transportation cost. The round trip mileage affects both the transportation cost and the annual cargo. The fuel price can fluctuate for example due to the change of the energy market situation. The round trip mileage can vary if the weather conditions change. We treat the values of the two parameters in the deterministic formulation as their nominal values since they are supposed to describe the most typical values of the parameters. We implemented the problem in MATLAB[®] and used a build-in solver with MultiStart to find X_{rpo} .

Before the solution process, we communicated with the decision maker and she was willing to sacrifice some quality to gain robustness. In step 1 of MuRO-NIMBUS, we calculated a representative set of 150 robust Pareto optimal solutions with the robust ASF approach and we also calculated the ideal and nadir objective vectors in the nominal case. Based on our computational experiments, 150 solutions were sufficient for this problem. Then we started the first iteration of the decision making stage.

Step 2(a). We set the three objectives in $I^<$ and solved (7). We found an initial nominal nondominated Pareto optimal solution from X_{rpo} .

Step 2(b). We presented the ideal objective vector $\mathbf{z}^* = (9.479, 716.3, 0.8534)^T$ and the nadir objective vectors $\mathbf{z}^{nad} = (12.813, 2040.1, 0.372)^T$ in the nominal case to the decision maker. Their components corresponding to each objective are also shown in the visual illustration. In the visual presentation, we used 10^3 tonne as the unit, i.e., the ideal and nadir values for the light ship mass was marked as 0.7163 and 2.0401 respectively. To help the decision maker to quickly read the number, we used a million tonnes as the unit for annual cargo.

Step 2(c). Then we presented the initial outcome to the decision maker as illustrated in Fig. 2. In the nominal case, 10.5 pounds/tonne for the transportation cost, 1090 tonnes light ship mass and the ship can handle 0.58 million tonnes cargo annually. The outcome in the worst case is marked in the figure. Even

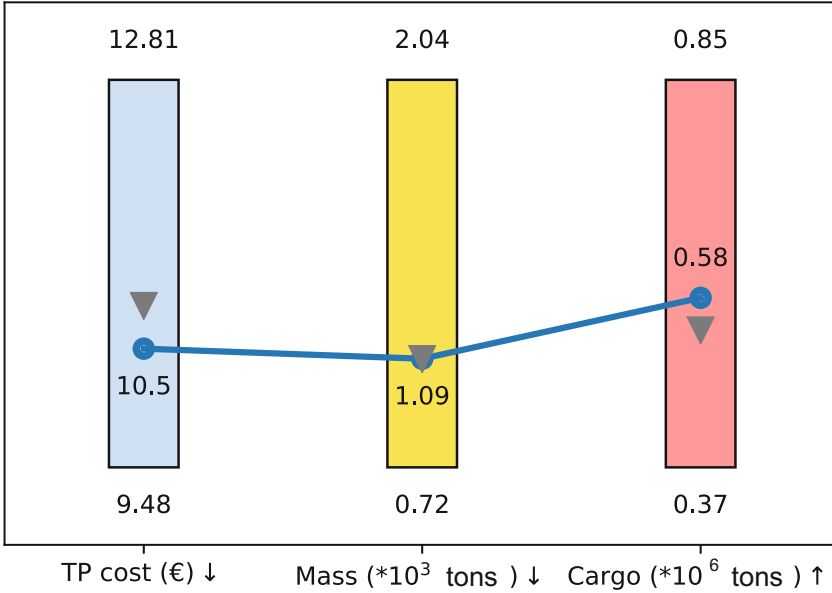


Fig. 2. Iteration 1 of ship design problem

though one of the considered uncertain parameters affects two objectives, we had only one worst case because the two objectives are not conflicting with each other. The decision maker was not satisfied with the solution and wanted to continue the solution process.

Step 2(d). The decision maker specified her preferences by classifying the objectives and wanted to improve the annual cargo as much as she can while allowing the light ship mass to be impaired until 1800 tonnes. In the NIMBUS classification, this corresponds to: $I^< = \{f_3\}$, $I^> = \{f_2\}$ with $\epsilon_2 = 1800$ and $I^\diamond = f_1$. Based on this classification, we solved (7). As a result, we got a new nominal nondominated robust Pareto optimal solution.

Iteration 2. We presented the new solution to the decision maker as in Fig. 3. The transportation cost was 9.51 pounds/tonne, and the light ship mass was 1640 tonnes while the annual cargo was 0.77 million tonnes in the nominal case. The decision maker observed in the visual presentation that the worst case outcome of the transportation cost did not degrade as much as in the initial outcome. Even though she seemed to have a solution whose outcome in the worst case did not degrade much compared to the outcome in the nominal case, she could not accept the light ship mass. So she decided to reduce the light ship mass to 1100 tonnes by allowing the transportation cost to increase until 10.9 pounds/tonne and the annual cargo to reduce until 0.5 million tonnes, i.e., she classified the objectives as $I^{\leq} = \{f_2\}$ with an aspiration level $\hat{z}_2 = 1100$ and $I^{\geq} = \{f_1, f_3\}$

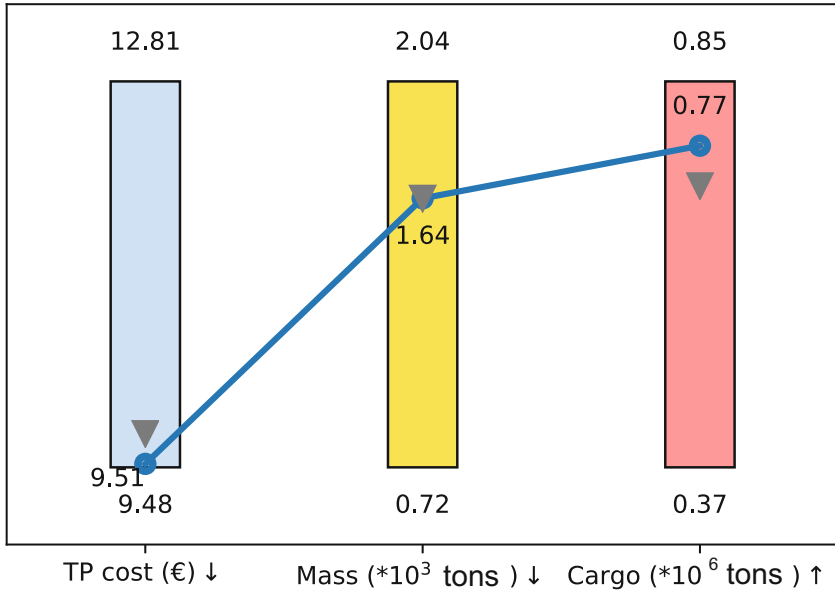


Fig. 3. Iteration 2 of ship design problem

with bounds $\epsilon_1 = 10.9$ and $\epsilon_3 = 0.5$. Based on this classification, problem (7) was solved to get a new solution.

Iteration 3. We presented the new solution to the decision maker as in Fig. 4 with 10.59 pounds/tonne for the transportation cost, 1040 tonnes as the light ship mass and 0.57 million tonnes annual cargo. With this solution, the decision maker noticed that even though the light ship mass was quite low, the other two objectives were at the same time approaching her specified bounds. She also observed that the value of the first objective function has higher degradation than the previous solution. She understood that she cannot have lower light ship mass if she is not willing to impair the other two objectives further and decided to stop. Naturally, if the decision maker is not satisfied, she can continue the solution process until she finds a most preferred solution.

During the solution process of the uncertain version of the multiobjective ship design problem, the decision maker was able to consider the outcomes in the nominal case with guaranteed robustness of solutions. Bearing in mind that the outcome in the nominal case of her final solution might not be as good as a deterministic Pareto optimal solution, she could still direct the interactive solution process towards a most preferred one among the robust Pareto optimal solutions according to their outcomes in the nominal case. Expressing her preferences by classifying the objectives did not bring her additional cognitive load. With the visualized information, she observed the outcomes of the solutions in the worst case in addition to the outcomes in the nominal case. Even though she could not interfere directly how the outcomes in the worst cases behaved, the

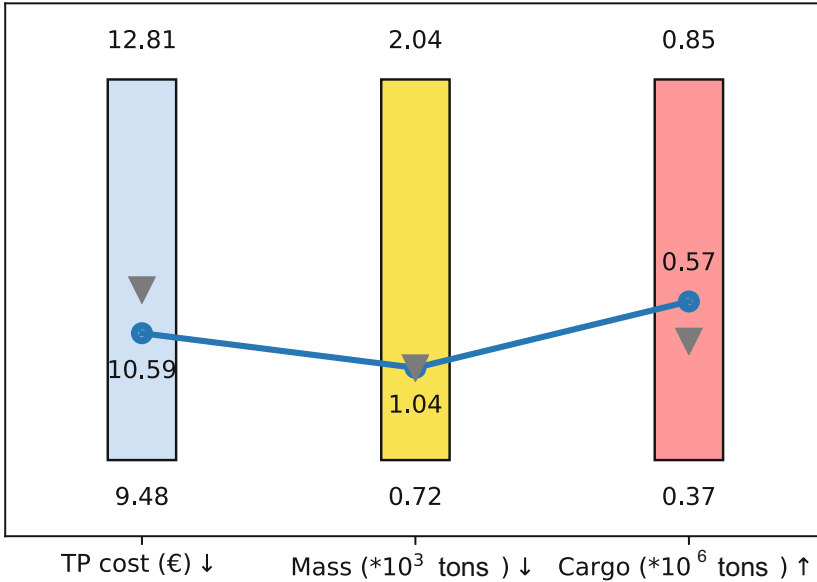


Fig. 4. Iteration 3 of ship design problem

information was critical for her to make an informed decision. In addition, if the worst case is realized, the solution the decision maker has would still be valid.

5 Conclusions

In this paper, we introduced MuRO-NIMBUS which is an interactive method for solving multiobjective optimization problems with uncertain parameters. In MuRO-NIMBUS, we support the decision maker to find a preferred balance by interacting in the nominal case but also following what happens in the worst case. We divided the consideration of the robustness and the outcomes in the nominal cases into the pre-decision making and the decision making stages. With the two-stage solution process, the decision maker finds a robust Pareto optimal solution with a preferred outcome in the nominal case and at the same time, the outcome in the worst case is also acceptable. In this way, the information provided to and requested from the decision maker is understandable in MuRO-NIMBUS. The decision maker can also be easily involved in the interactive solution process without much additional cognitive load. By providing the information in both nominal and worst cases, MuRO-NIMBUS supports the decision maker to make an informed decision. We demonstrated the application of MuRO-NIMBUS with an example problem.

The development of MuRO-NIMBUS has initiated many avenues for further research. First, some additional features on the decision making stage can be developed. We can allow the decision maker to choose whether (s)he would like

to find a most preferred solution based on the corresponding outcome in the nominal case (as is done in MuRO-NIMBUS), or in the worst case. This will allow the decision maker to consider different aspects during the decision making process. As an essential part to support the decision maker, we can also consider how to visualize the solutions more effectively. Second, a decision maker might want to find a robust Pareto optimal solution but with only a limited amount of sacrifice on the quality. To achieve this, we can study some other robustness concepts and analyze their properties from the decision making point of view aiming at finding a good trade-off between robustness and quality.

References

1. Bertsimas, D., Sim, M.: The price of robustness. *Oper. Res.* **52**(1), 35–53 (2004)
2. Bokrantz, R., Fredriksson, A.: Necessary and sufficient conditions for Pareto efficiency in robust multiobjective optimization. *Eur. J. Oper. Res.* **262**(2), 682–692 (2017)
3. Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.): *Multiobjective Optimization: Interactive and Evolutionary Approaches*. LNCS, vol. 5252. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-3-540-88908-3>
4. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(5), 773–791 (2016)
5. Ehrgott, M., Ide, J., Schöbel, A.: Minmax robustness for multi-objective optimization problems. *Eur. J. Oper. Res.* **239**(1), 17–31 (2014)
6. Geoffrion, A.M., Dyer, J.S., Feinberg, A.: An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Manage. Sci.* **19**(4), 357–368 (1972)
7. Hassanzadeh, F., Nemati, H., Sun, M.: Robust optimization for multiobjective programming problems with imprecise information. *Procedia Comput. Sci.* **17**, 357–364 (2013)
8. Hassanzadeh, F., Nemati, H., Sun, M.: Robust optimization for interactive multiobjective programming with imprecise information applied to R&D project portfolio selection. *Eur. J. Oper. Res.* **238**(1), 41–53 (2014)
9. Ide, J., Schöbel, A.: Robustness for uncertain multi-objective optimization: a survey and analysis of different concepts. *OR Spectr.* **38**(1), 235–271 (2016)
10. Kuhn, K., Raith, A., Schmidt, M., Schöbel, A.: Bi-objective robust optimisation. *Eur. J. Oper. Res.* **252**(2), 418–431 (2016)
11. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston (1999)
12. Miettinen, K.: Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectr.* **36**(1), 3–37 (2014)
13. Miettinen, K., Mäkelä, M.M.: Synchronous approach in interactive multiobjective optimization. *Eur. J. Oper. Res.* **170**(3), 909–922 (2006)
14. Miettinen, K., Mustajoki, J., Stewart, T.J.: Interactive multiobjective optimization with NIMBUS for decision making under uncertainty. *OR Spectr.* **36**(1), 39–56 (2014)
15. Sabioni, C.L., de Oliveira Ribeiro, M.F., de Vasconcelos, J.A.: Decision maker iterative-based framework for multiobjective robust optimization. *Neuro Comput.* **242**, 113–130 (2017)

16. Sawaragi, Y., Nakayama, H., Tanino, T.: Theory of Multiobjective Optimization. Academic Press, London (1985)
17. Steuer, R.E., Choo, E.U.: An interactive weighted Tchebycheff procedure for multiple objective programming. *Math. Program.* **26**(3), 326–344 (1983)
18. Steuer, R.: Multiple Criteria Optimization: Theory, Computation, and Applications. Wiley, New York (1986)
19. Wiecek, M.M., Dranichak, G.M.: Robust multiobjective optimization for decision making under uncertainty and conflict. In: Gupta, A., Capponi, A., Smith, J.C., Greenberg, H.J. (eds.) Optimization Challenges in Complex, Networked and Risky Systems, pp. 84–114 (2016)
20. Wierzbicki, A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spectr.* **8**(2), 73–87 (1986)
21. Yang, J.B.: Minimax reference point approach and its application for multiobjective optimization. *Eur. J. Oper. Res.* **126**(3), 541–556 (2000)



Heuristics and Simulation for Water Tank Optimization

Corinna Hallmann, Sascha Burmeister^(✉), Michaela Wissing^(✉),
and Leena Suhl^(✉)

Decision Support and Operations Research Lab,
Paderborn University, Paderborn, Germany
{hallmann,burmeister,wissing,suhl}@dsor.de

Abstract. In the last two decades, water consumption in Germany has been decreasing which causes water tanks and pipes in water distribution systems to work inefficiently. This paper presents a mathematical optimization model to optimize water tanks in a water distribution system. Due to the hydraulic properties in water distribution systems the model is a non-convex Mixed Integer Quadratically Constrained Program (MIQCP). For problem instances of realistic size, the model cannot be solved within reasonable time with exact solution methods. We use different heuristic solution methods to solve the problem, such as a Simulated Annealing (SA) algorithm, a Shuffled Complex Evolution (SCE) algorithm as well as a Shuffled Frog-Leaping Algorithm (SFLA). These methods are combined with a hydraulic simulation to evaluate the solutions. The results of each method are compared to an exact solution method and discussed in this paper.

1 Introduction

In recent years, German municipal utilities responsible for the local water supplies are facing an increasing cost pressure. This is caused by several reasons [12]. One of the main reasons is the decreasing water consumption in Germany in the last two decades. When building water distribution systems, the planners forecasted an increasing demand of water in the future and planned the size of the components accordingly. Due to decreasing water consumption there are now many components that work inefficiently. For utilities there is great potential to increase the efficiency and therefore decrease the costs in a water distribution system. In this paper, the focus is on optimizing water tanks, which means deciding the optimal locations, dimensions and material properties of water tanks. As the utilities require an accurate solution in reasonable time for this problem, the challenge is to find a nearly optimal solution in a small amount of time.

Optimizing water tanks is a highly complex task due to their great influence on the hydraulics in the whole water distribution system. When modeling the hydraulics, it is necessary to use non-convex and nonlinear equations, which make the corresponding optimization models hard to solve. There are a

few approaches to tackle this problem in the literature. Hallmann and Suhl [9] present an optimization model which optimizes the planning of water tanks. The model is solved via a combination of network reduction, piecewise linearization and hydraulic simulation. Farmani et al. [8] propose a model which optimizes not only the locations and dimensions of tanks but also pump operation schedules. To identify the payoff characteristics between total cost and reliability of a water distribution system, they combine an Elitist Nondominated Sorting Genetic Algorithm method (NSGAI) with the simulation tool EPANET [14], that uses an extended time period simulation to compute the hydraulic properties. Vamvakeridou-Lyroudia [15] uses such a simulation to handle the problem of optimizing water tanks as well. The simulation is used to determine the inflow and outflow of water tanks. This information is combined with a Genetic Algorithm (GA) that chooses a new level for each tank. The optimization of size and location of water tanks is also considered in the work of Kurek and Ostfeld [11] who present two optimization models that additionally minimize the energy costs of pumps and optimize the water quality in the system. The two models differ in the objective function for the water quality. These models are solved via a combination of a Strength Pareto Evolutionary Algorithm II (SPEA2) and a simulation. In the work of Zheng et al. [16] an Ant Colony Optimization (ACO) algorithm is applied to several water distribution design problems. The authors introduce a novel method to control the convergence of the ACO and show the efficiency of this method.

Inspired by those ideas, we tackle the optimization model of [9], with a combination of heuristics and hydraulic simulation. In [9] the results indicate that for larger instances the solution time can be improved. Therefore, in this paper the optimization problem is treated with heuristics instead of exact optimization methods. The paper is organized as follows. Section 2 introduces shortly the fundamentals of water distribution systems. In Sect. 3 we give a short overview of the optimization model and then present the application of three different heuristics, namely a Simulated Annealing algorithm, a Shuffled Complex Evolution algorithm and a Shuffled Frog-Leaping Algorithm. After introducing these algorithms, Sect. 4 shows the computational results. The paper closes with a conclusion in Sect. 5.

2 Overview of Water Distribution Systems

In this section, the basic components and principles of water distribution system are described. Water distribution systems are built to transport water from several sources to different clients. Figure 1 gives an overview of the main components in a water distribution system.

The components of the system can be divided into nodes and links whose different meanings are described as follows. A node can be a reservoir, which is a natural source of water such as a lake or a river or it may represent a tank, which can store water and supply water into the system. Nodes can also represent junctions where some links join together. At junctions water can enter or leave

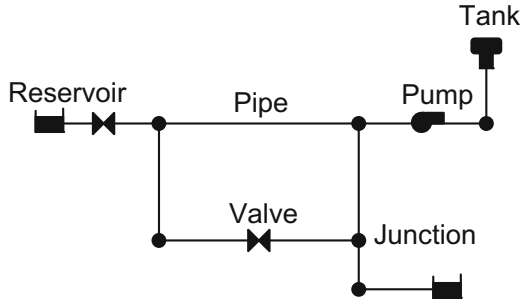


Fig. 1. Components in a water distribution system

the system. All nodes have in common that they have a hydraulic head, which is mainly determined by the elevation of the node. If a node represents a tank, the hydraulic head is also affected by the water level of the tank. The hydraulic head of a node is also influenced by the connected links and their properties. Most links represent pipes through which water flows from one node to another one. Links can also be valves, which can control the pressure in a pipe or the amount of water flowing through a pipe. A link may also represent a pump, which increases the pressure within a link, which is mainly used to transport water from nodes with a low elevation to nodes with a higher elevation. For further details of the main components in water distribution systems, we refer to [13, 14] or [10].

3 Solving an Optimization Model for Water Tanks

This section describes shortly the optimization model. The complete version of the model can be found in [9]. We consider a water distribution system that can be a completely new or an existing system. The optimization task is to minimize the costs of all water tanks in the system. Depending on the costs, the model can decide where to build new tanks, replace or scale up or down existing tanks. In addition, the dimensions and the material properties of a tank can be determined. This task is restricted by several constraints. Some of the constraints control the specific requirements of the tanks and their hydraulic behavior. There has to be a minimal amount of water in the system to satisfy the demand of each client at any time and provide the necessary amount of water for firefighting or any other kind of incident that can occur. Furthermore, the hydraulics in a water distribution system have to be considered, such as the volumetric flow through each pipe and the hydraulic head on each node. To model the hydraulics accurately, we add a nonlinear head loss equation to the model. This non-convex quadratic equation is introduced for each link to model the loss of the head within the link. Due to discrete decisions and this equation, the proposed optimization model becomes a NP-hard non-convex Mixed Integer Quadratically Constrained Program (MIQCP).

In [9] exact solution methods to solve the problem are used. When considering larger instances this may take quite a while. Therefore, we decided to solve the proposed optimization model with three different heuristic solution methods and combine them with a hydraulic simulation.

3.1 Simulated Annealing Algorithm

The first algorithm applied to this model is a Simulated Annealing (SA) algorithm. This algorithm imitates the cooling and annealing of metal. When metal cools, the molecules often align themselves into an energy advantageous structure. If the cooling process is very fast, the structure may not be optimal. To achieve a very good structure, a controlled annealing and cooling process can be used. This process is represented in a Simulated Annealing algorithm. As this algorithm is known as a very simple, efficient and robust method, it is applied to the model. More details of this algorithm can for example be read in the work of Barakat et al. [2], who applied this technique to an optimization model that determines the optimal structure of a single water tank. In our work the algorithm is used to find feasible values for the volume of stored water in a tank and the decision, if a tank should be built or change its size. In Table 1 all parameters of the SA algorithm are listed. The default values for the starting parameters were obtained in numerical experiments. Figure 2 shows an overview of the algorithm.

Table 1. Parameters of the Simulated Annealing algorithm

| Parameter | Meaning | Range | Default |
|-------------------|--|-----------------------|----------------|
| s_0 | Current feasible solution | - | - |
| s | Solution in the neighborhood of s_0 | - | - |
| t_0 | Initial temperature | 500–3000 | 1000 |
| t | Current temperature | - | - |
| $\alpha(t)$ | Function to model cooling process | $(0.7-0.999) \cdot t$ | $0.98 \cdot t$ |
| $maxIter$ | Maximal number of iterations within one temperature cycle | 5–30 | 10 |
| $f(s)$ | Objective value of solution s | - | - |
| Δ | Difference of the objective values $f(s_0)$ and $f(s)$ | - | - |
| $\exp(-\Delta/t)$ | Function to determine the probability of adopting a solution | - | - |
| t_{min} | Lower bound for temperature as stopping criterion | - | - |

The algorithm is initialized with a feasible solution s_0 that is provided by a hydraulic simulation described at the end of this section. We also determine an

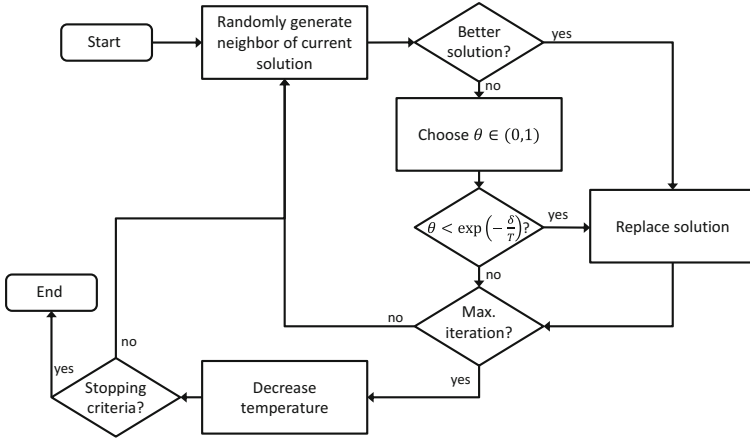


Fig. 2. Overview of the Simulated Annealing algorithm

initial temperature t_0 , a function $\alpha(t)$ to reduce the temperature within the solution process as well as a parameter $maxIter$ that defines the maximal number of iterations within one temperature cycle. In the next step we choose a solution s in the neighborhood of s_0 . To find this solution we choose one of the tanks in the system and vary the parameters of this tank. These are the decisions if the tank should exist or not, the volume of stored water and the dimensions. After that we calculate the objective function values $f(s_0)$ and $f(s)$ and compare them. If the objective value of s is better, we choose s as the best solution found so far. If not, the algorithm can decide to choose this solution nevertheless as the new best solution. This process is controlled via the function $\exp(-\Delta/t)$, where Δ is the difference of the two objective values. If $\Delta < \exp(-\Delta/t)$ holds, the new solution will replace the old one. This process is used to prevent the algorithm to remain at a local optimum. Afterwards, we determine a new solution in the neighborhood of the current solution s_0 . This process is carried out until the maximal number of iterations within one temperature cycle is reached. After that the temperature is reduced via the function $\alpha(t)$ and the process is started anew. This is executed until the stopping criterion is fulfilled. In our implementation the algorithm stops, when the temperature is less than a predefined level t_{min} . The algorithm provides the solution s_0 , which is the best found approximation to the optimal solution.

3.2 Shuffled Complex Evolution Algorithm

The second technique is a Shuffled Complex Evolution (SCE) algorithm. The SCE starts with a set of points constituting a population that is separated into several equally large complexes. These complexes are allowed to evolve independently from each other during reflection, correction and contraction steps. Within these steps the worst point of a complex is reflected through the constructed centroid to find a new and better point. After every complex evolves

itself, the complexes are shuffled and divided into new complexes again. With that shuffling, information that was gained within a single complex can be shared with other complexes. This procedure is performed until some stopping criteria are achieved. Further details about the general idea of SCE can for example be found in [2]. The algorithm is based on different concepts, such as clustering, systematic evolution of a complex of points spanning the space, competitive evolution and a combination of probabilistic and deterministic approaches, cf. [6]. The SCE is known to be robust, effective and efficient for a broad class of problems, even for global optimization problems. Therefore, it is well suited for the optimization problem presented above.

In our implementation a point is a configuration for all possible tank locations. Besides the volume of each tank, this configuration defines, whether a tank should be built, replaced by a new one or change its properties. The parameters we used for our implementation of the SCE can be found in Table 2. To obtain default values for the starting parameters numerical experiments were carried out. An overview of the algorithm can be found in Fig. 3.

Table 2. Parameters of the Shuffled Complex Evolution algorithm

| Parameter | Meaning | Range | Default |
|-----------|---|----------|---------|
| s | Number of points | ≥ 1 | - |
| c | Number of complexes | ≥ 1 | 5 |
| m | Number of points in each complex | ≥ 2 | 7 |
| q | Number of points in each sub-complex | ≥ 1 | 3 |
| α | Offspring generated by each sub-complex | > 1 | 4 |
| β | Evolution steps allowed for each complex shuffling | > 1 | 4 |
| tf | Probability of removing tanks | 0–100 | 75 |
| imp | Required minimal improvement of objective per iteration | ≥ 0 | 0.001 |
| $maxKimp$ | Maximal number of iterations without improvement | > 0 | 12 |
| $maxK$ | Maximal number of overall iterations | ≥ 0 | 30 |

The SCE starts with a feasible initial solution and then generates $s = c \cdot m$ points. These points are generated by calculating different values for the volume of the tanks by

$$V_{new} = \lambda \cdot (maxV - minV) + minV, \quad \lambda \in [0, 1], \quad (1)$$

where $maxV$ and $minV$ are the maximal and minimal volume the tank can contain, respectively. This equation ensures that the volume lies between these bounds. By a probability of tf , this value is set to 0 to forbid the tank and remove it from the network.

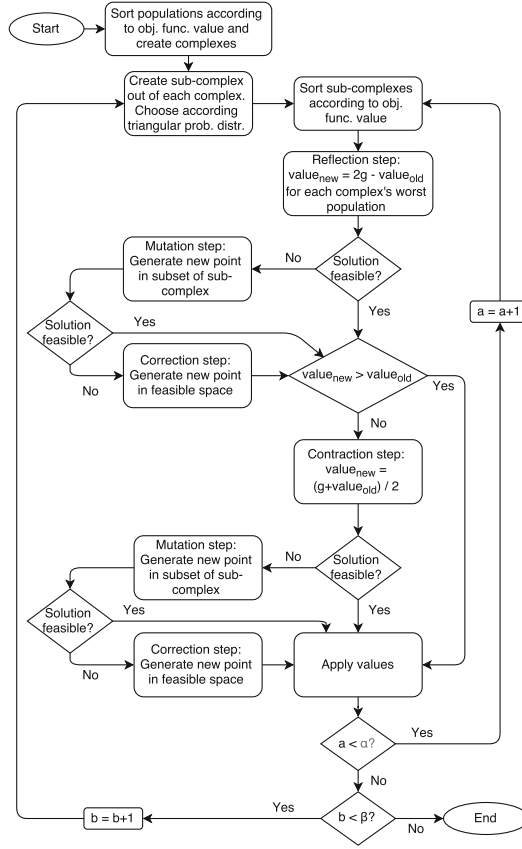


Fig. 3. Overview of the Shuffled Complex Evolution algorithm

After a point is created, the algorithm uses a hydraulic simulation described in Sect. 3.4 and simulates the configuration of the point to check the feasibility. If it is not feasible, the point is replaced by a new one. This process is carried out until all points are within the feasible space.

Now, the evolution process of the SCE starts. The SCE sorts the s points according to their objective function values and divides them into p complexes each with m points. Afterwards, q point of each complex are selected for a sub-complex according to a trapezoidal probability distribution with the probability of:

$$p_i = \frac{2(m+1-i)}{m(m+1)}, \quad \text{for } i = 1, \dots, m. \quad (2)$$

For each sub-complex sc the SCE calculates the centroid for each tank and carries out the *reflection step*, where it takes the worst point of the complex and adjusts each tank configuration by calculating the reflected volume with

$$V_{new} = 2 \cdot centroid_{sc} - V_{old}, \quad (3)$$

where $centroid_{sc}$ is the centroid of sub-complex sc . Now, the algorithm performs a hydraulic simulation for the new point. If it is not feasible, a *mutation step* applies and calculates a new point with new tank volumes by using

$$V_{new} = \lambda \cdot (maxV_{sc} - minV_{sc}) + maxV_{sc}, \quad \lambda \in [0, 1]. \quad (4)$$

If this procedure still produces an infeasible solution, the point is deleted and a *correction step* takes place. In this step a new point is created and simulated. The volumes in this step are calculated via

$$V_{new} = -\frac{h \cdot \lambda \cdot V_{init}}{2} + V_{init}, \quad \lambda \in [0, 1]. \quad (5)$$

The parameter h is introduced to control the feasibility of the new volume. At the beginning h has the value of 1 and decreases with each iteration by 0.005. With that, it can be ensured that the volume decreases step by step and will finally have the value of the original volume. If still no feasible solution can be found, the algorithm stops exploring the current complex.

After this process, the SCE compares the function value of the new point with the old one. If the new value is better, the new point is adopted. Otherwise a *contraction step* is performed that calculates new values for each tank with

$$V_{new} = \frac{centroid_{sc} + V_{old}}{2}. \quad (6)$$

Again, the SCE checks if the calculated solution is feasible and starts a *mutation step* and *correction step* as described before, if the new solution is infeasible or not good enough. Eventually, the new calculated point replaces the old one.

The next step of the SCE is to repeat the evolution steps. It returns to the step of sorting the sub-complexes according to the objective function values and repeats the evolution steps α times. Afterwards it shuffles the sub-complexes by creating new ones and starts the evolution process once more. This shuffling is performed β times.

After that, a whole iteration of the SCE is performed. Now, the algorithm checks if the results meet the stopping criteria. In this case there are two stopping criteria. The first uses the parameter *imp*, which is the required minimal improvement of the objective function value. If the function value does not improve by this amount after an iteration, there are just *maxKimp* more iterations until the algorithm terminates. If the improvement during these *maxKimp* iterations is better than *imp*, this stopping criterion does not longer hold. A second stopping criterion is the absolute maximum number of overall iterations *maxK*. When the algorithm reaches this number of iterations, it will stop in any case.

3.3 Shuffled Frog-Leaping Algorithm

As a third technique we use a Shuffled Frog-Leaping Algorithm (SFLA). Inspired by different evolutionary algorithms, which are based on the behavior of animals, Eusuff et al. [7] developed the SFLA. They use the advantages of group behavior, such as social communication and sharing information. Thus, the SFLA is based on a global exchange of information, which is carried by a virtual population of frogs. Eusuff et al. [7] make use of biological technical terms, when they talk about memes as the assigned information each frog contains. They also explain the difference between genes and memes: Whereas genetic evolution is mainly needed for a selection because of reproduction, memetic evolution is a much more faster, flexible and communicative way to spread information among individuals. Each frog carries its information as a meme, while the information contained in a whole population can be seen as a memetic vector. During the algorithm, virtual frogs are leaping through an area, searching for a stone with the most amount of food, which can be interpreted as searching for the optimal solution within the feasible space of the optimization problem. In this process, the frogs are allowed to communicate and making it possible to expand their knowledge to increase their opportunities for gaining more food. In the algorithm, this idea is implemented by creating an amount of virtual frogs within the feasible space, ranking them according to their objective function value and dividing them into so called memeplexes. The memeplexes are comparable to the complexes in the SCE. The memeplexes develop independently to ensure that they search in different locations in the feasible space. After a certain number of evolutionary steps, the frogs are mixed and ranked again to build new memeplexes and share their acquired knowledge with new communities in different regions. The algorithm ends, when one of the stopping criteria is met. Eusuff et al. [7] describe the general idea of this algorithm in more detail. Like the SCE, the SFLA is a robust and efficient algorithm, as it examines a broad area of the solution space. The SFLA has also the advantage that it can escape local optima and shares information gained in different areas of the solution space. Therefore, it is applied to the optimization model in this paper.

Similar to the SCE, we start our implementation of the SFLA by defining some points representing different tank configurations. The parameters that are used and the default values of the starting parameters can be found in Table 3. The last four parameters have the same meaning as in the SCE. The assignment of the default values were obtained in numerical experiments.

The start of the SFLA is similar to the one of the SCE. It begins with a feasible initial solution and creates $s = m \cdot n$ points and checks them for feasibility. The volume of each tank of a point is calculated via

$$V_{new} = \lambda \cdot (maxV - minV) + minV, \quad \lambda \in [0, 1]. \quad (7)$$

With a probability of tf , V_{new} is set to 0, to close the tank in the model. If a point is not feasible, it is replaced by a new, randomly generated one. When every point is feasible, the SFLA separates them into m memeplexes, ordered

Table 3. Parameters of the Shuffled Frog Leaping Algorithm

| Parameter | Meaning | Range | Default |
|-----------|---|----------|---------|
| m | Number of memplexes | ≥ 1 | 5 |
| n | Number of frogs per memplex | ≥ 2 | 5 |
| q | Number of frogs in each sub-complex | ≥ 1 | 3 |
| max_S | Maximal step size | ≥ 0 | 30 |
| tf | Probability of removing tanks | 0–100 | 75 |
| N | Number of evolutionary steps per memetic evolution | ≥ 1 | 4 |
| imp | Required minimal improvement of objective per iteration | ≥ 0 | 0.001 |
| $maxKimp$ | Maximal number of iterations without improvement | > 0 | 25 |
| $maxK$ | Maximal number of overall iterations | ≥ 0 | 50 |

by their objective function value. Now, the SFLA starts the memetic evolution, which is visualized in Fig. 4.

During the evolution, the memplexes are sorted in order to their objective function value. Afterwards, sub-memplexes are created out of each memplex, choosing q points according to the trapezoidal probability distribution with

$$p_i = \frac{2(q+1-i)}{q(q+1)}, \quad \text{for } i = 1, \dots, q. \quad (8)$$

Now, the SFLA improves every memplex's worst point by letting the point take a *positive reflection-step*. It updates the volume with

$$V_{new} = V_{old} + \min\{\lambda \cdot (V_b - V_w), max_S\}, \quad \lambda \in [0, 1], \quad (9)$$

where the value V_b represents the best point's volume inside the sub-memplex, V_w the volume of the worst point of the sub-memplex and max_S the maximal step size. The new values are applied, if there are no infeasibilities and the objective function value is better than the old one. Independent of the feasibility and objective function value, the SFLA performs a *negative reflection step*, calculating the volumes via

$$V_{new} = V_{old} + \min\{\lambda \cdot (V_b - V_w), -max_S\}, \quad \lambda \in [0, 1]. \quad (10)$$

If the negative reflection step produces a better objective function value, this values replace the values from the positive reflection step. If the original value of the point is still better than both new values, or if the reflection steps produced only infeasible solutions, the algorithm starts with contraction steps.

Similar to the reflection steps, the SFLA first performs a *positive contraction step*. This is then followed by a *negative contraction step*. The new volumes are

calculated by

$$V_{new} = V_{old} + \min\{\lambda \cdot (V_x - V_w), \max_S\}, \quad \lambda \in [0, 1], \quad (11)$$

for a positive step and

$$V_{new} = V_{old} + \min\{\lambda \cdot (V_x - V_w), -\max_S\}, \quad \lambda \in [0, 1], \quad (12)$$

for a negative, respectively. The value of V_x is the volume of the tank of the best point, regardless, whether it is within the sub-memplex or not. With this step it is possible to profit from information of other regions in the complex. After the contraction steps, the SFLA checks whether the memetic evolution can produce a better point. If all steps produce infeasible solutions, or if the steps cannot find a better solution, the SFLA starts a *censor step* and replaces the point with a new random point. The new point is created like the initial points, see formula (7).

Now, the algorithm picks the next worst point of the sub-memplex and repeats the reflection, contraction and censoring steps, until it performed a total amount of N iterations. After that, one iteration of memetic evolution is performed. Afterwards, the algorithm shuffles the memplexes, checks the stopping criteria and starts a new iteration of memetic evolution. Or it terminates and returns the best point that was found, depending on the stopping criteria.

3.4 Hydraulic Simulation

We use a hydraulic simulation tool to evaluate the solutions of the three algorithms presented above. This simulation tool was developed by our industry partner *Rechenzentrum für Versorgungsnetze Wehr GmbH* and is based on EPANET [14]. Like EPANET, the simulation is an extended period simulation and is able to calculate flows and heads of a water distribution system model during all time periods. The simulation considers all components that may appear in a water distribution system such as pipes, pumps, valves, reservoirs and tanks. It is based on the network structure of the system and divides the components into nodes and links. Compared to the MIQCP model, the simulation model considers more details, such as network based rules or water quality properties. However, the calculated values for the heads and the flows are the same in both models.

In order to calculate the heads and the flows, the mass balance equation and the head loss equation have to be considered for each node and link, respectively. There are many different algorithms that can be used to calculate the uprising system of nonlinear equations. There exist for example the *Hardy-Cross method*, the *Flow Adjustment algorithm*, the *Simultaneous Adjustment algorithm* and the *method of Todini and Pilati*. Details about those methods can be read in [4]. In our tool the method of Todini and Pilati is implemented. This method is based on the Newton's method and the solution time and the convergence rate are faster than those of the other methods.

After introducing all three heuristics and the hydraulic simulation, we will show in the next Section the results of the implemented algorithms.

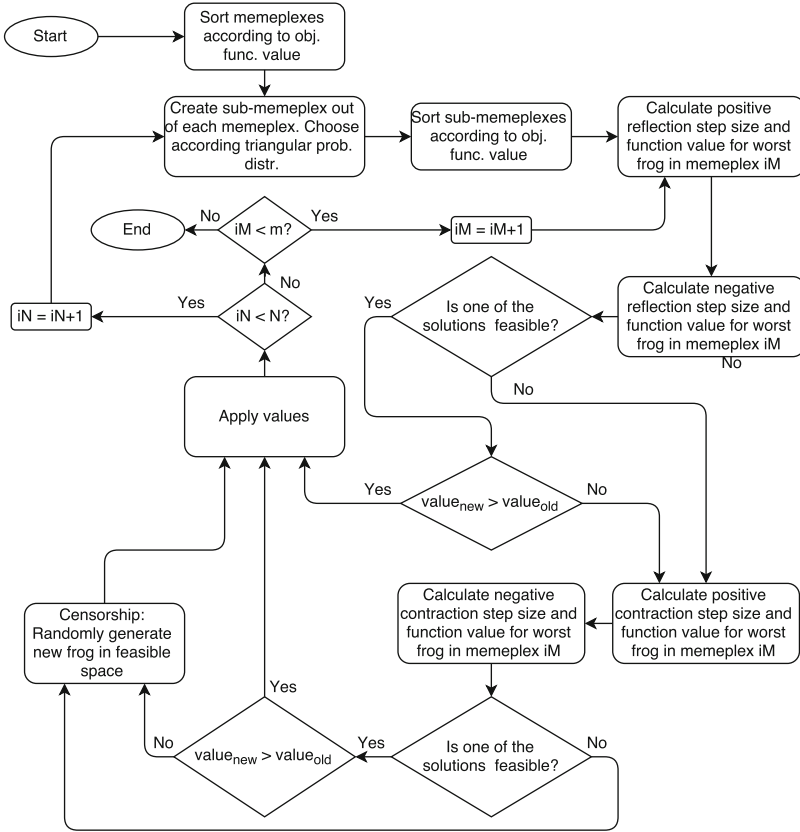


Fig. 4. Overview of the Shuffled Frog Leaping Algorithm

4 Computational Results

For the computational results we use 23 different network models. These models have a size of 8 nodes and 10 links up to a size of 932 nodes and 1014 links. Table 4 shows the properties of the models. The network models are divided into small networks, indicated with (*S*), medium sized networks, indicated with (*M*) and large networks, indicated with (*L*). Networks M06 to M08 and L01 to L04 are realistic networks based on German or Italian cities, while the others are fictitious. Most networks were provided by our industry partner, while others are taken from the literature, cf. [5].

Figure 5 shows the average of the computational times of the algorithms. The tests were performed on an Intel Core i7-3770 processor with 32 GB of RAM and the time limit was set to 300 min for each algorithm. We used SCIP, cf. [3], to solve the models in order to compare the solutions to the three algorithms presented above. All algorithms were started with the same initial feasible solution,

Table 4. Properties of network models

| Name | # Nodes | # Links | # Tanks | # Res. | Name in literature |
|------|---------|---------|---------|--------|--------------------|
| S01 | 8 | 10 | 3 | 1 | |
| S02 | 10 | 12 | 1 | 0 | |
| S03 | 11 | 12 | 4 | 1 | Shamir |
| S04 | 12 | 11 | 8 | 1 | |
| S05 | 12 | 11 | 5 | 1 | |
| S06 | 14 | 13 | 8 | 0 | |
| S07 | 14 | 15 | 7 | 1 | |
| S08 | 17 | 16 | 10 | 0 | |
| S09 | 18 | 17 | 6 | 1 | |
| S10 | 20 | 21 | 15 | 1 | |
| S11 | 21 | 20 | 15 | 0 | |
| M01 | 27 | 26 | 15 | 1 | |
| M02 | 30 | 31 | 10 | 1 | New York |
| M03 | 34 | 38 | 20 | 0 | |
| M04 | 42 | 44 | 10 | 1 | Hanoi |
| M05 | 46 | 50 | 15 | 1 | |
| M06 | 52 | 73 | 15 | 1 | Foss_Poly_0 |
| M07 | 56 | 77 | 20 | 0 | Foss_Iron |
| M08 | 86 | 114 | 15 | 3 | Pescara |
| L01 | 219 | 250 | 20 | 0 | |
| L02 | 300 | 345 | 30 | 2 | Modena |
| L03 | 916 | 973 | 25 | 1 | |
| L04 | 932 | 1014 | 30 | 2 | |

which is the given network model with predefined parameters for the components of the network.

It can be seen that the solution time of the SA grows with the size of the models and is acceptable also for the large models. The results of the SCE are similar, but solving the smaller network models takes a larger amount of time than solving medium sized network models. The SFLA shows a different behavior. Compared to the SA and SCE the SFLA was much faster in finding a solution for all instances. Taking a look at the results of SCIP it can be observed that SCIP is faster in solving the very small network models to optimality. For network model S09 and all following network models, the SFLA was faster in finding a solution than SCIP. The networks M06 to L04 could not be solved via SCIP to optimality within 24 h. It seems that the SFLA is superior to the SA, SCE and even SCIP. But to evaluate the overall performance of the algorithms we have to consider the quality of the solutions.

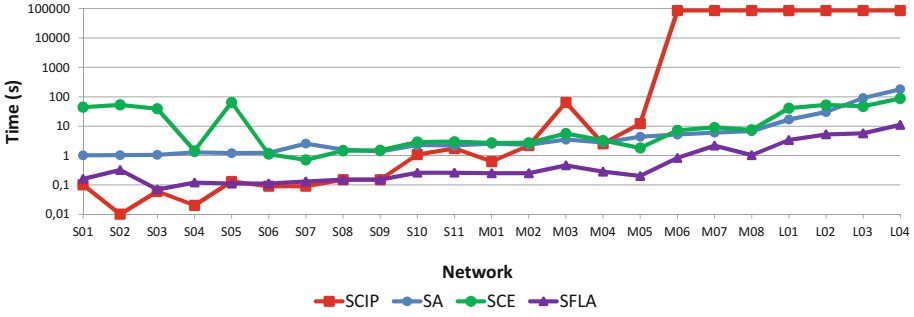


Fig. 5. Comparison of the average computational times of the algorithms

To evaluate the quality of the solutions we divide the network models into two groups. The first group are models that could be solved to optimality via SCIP, i.e. network models S01 to M05. The second group are models that could not be solved to optimality via SCIP, i.e. network models M06 to L04.

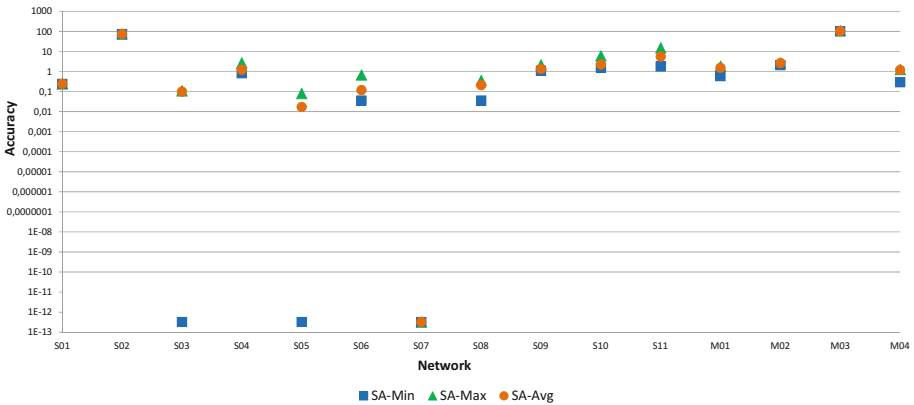


Fig. 6. Comparison of the accuracy of the SA, network models S01 to M05

Figures 6, 7 and 8 show the results of solving network models S01 to M05 with the SA, SCE and SFLA, respectively. The diagrams show the relative deviation of the objective values of the three algorithms compared to the optimal objective values calculated with SCIP. The minimal, maximal and average results are displayed for each network and each algorithm, which was carried out 50 times each. It can be seen, that the SA works robust on the optimization model as the minimal, maximal and average values are close to each other, except for network model S03 and S05. For most of the networks the accuracy of the SA is satisfying. However, there are also networks on which the behavior of the SA is bad. The SCE works also robust, but there are also a few networks where the

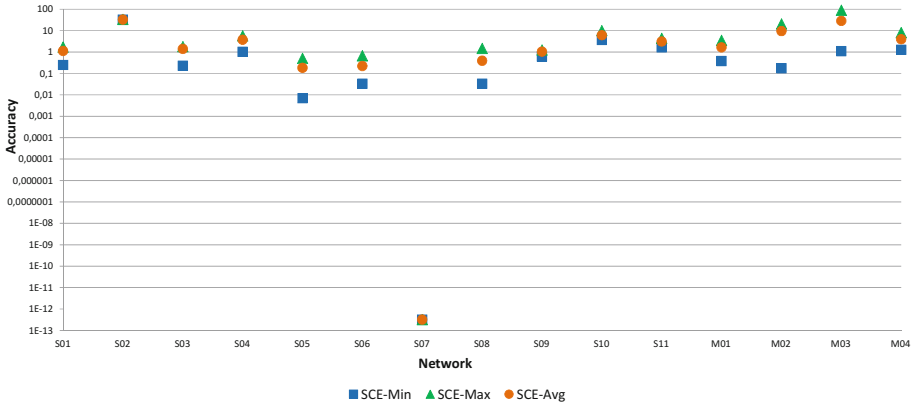


Fig. 7. Comparison of the accuracy of the SCE, network models S01 to M05

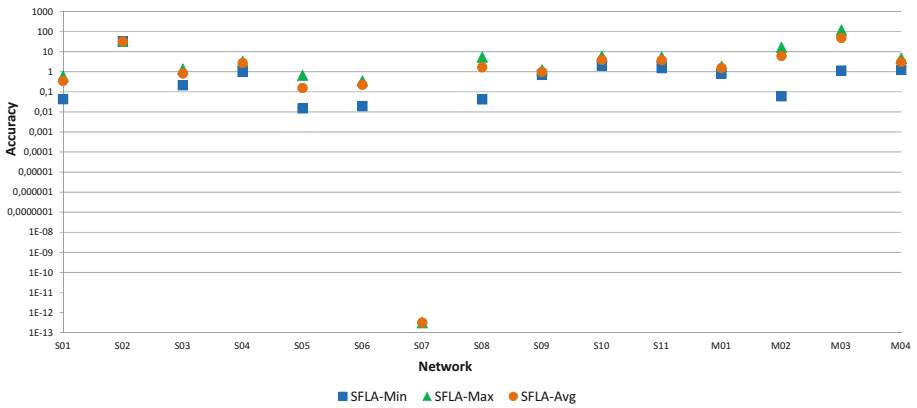


Fig. 8. Comparison of the accuracy of the SFLA, network models S01 to M05

deviation of the average and the minimal value is large. However, in most cases the results of the SCE are satisfying. The results of the SFLA are similar. The algorithm seems robust and the results are for most networks satisfactory.

Figure 9 shows the relative deviation of the average objective values of the three algorithms compared to the optimal objective values calculated with SCIP. It can be seen that all three algorithms worked quite similar and found in most cases good solutions. However, there are a few network models for which all the algorithms performed in a bad way, such as at network S02, S04, S10, S11, M02, M03 or M04. Comparing the three algorithms it can be seen that for each algorithm there is a network where the algorithm is superior to the other ones. In summary, it can be observed that for this group all three algorithms can be used to find good solutions in a small amount of time.

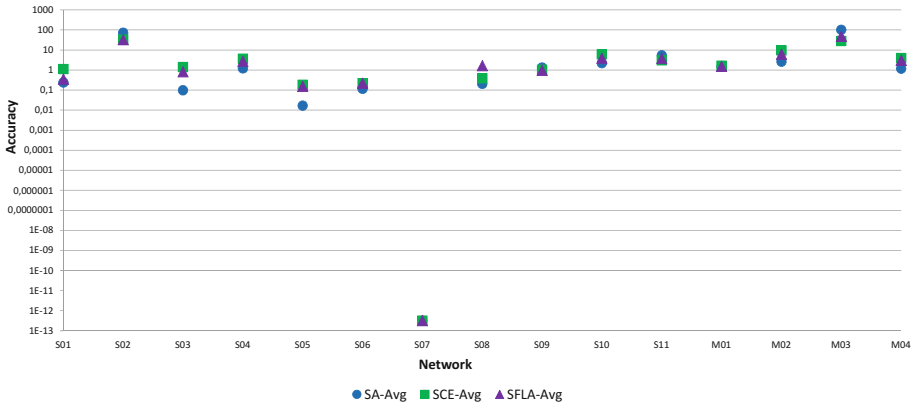


Fig. 9. Comparison of the average accuracy of the three algorithms, network models S01 to M05

In Table 5 the results of network models M06 to L04 are displayed. The table shows the average of the objective values found by the different algorithms and highlights the best average solution for each network model. It can be seen that the SA found in most cases the best average solution value whereas for four networks SCIP could not find any feasible solution at all after 24 h. The objective values of the three network models for which SCIP could find a feasible solution are also far away from the best solution values of the three heuristics. That shows that it is necessary to use the three heuristic algorithms for larger network models.

Table 5. Comparison of the accuracy of the algorithms, network models M06 to L04

| Net | SCIP | SA-avg | SCE-avg | SFLA-avg |
|-----|--------------|--------------------|--------------------|--------------|
| M06 | 10,234,205.9 | 5,680,080.8 | 4,850,122.4 | 7,359,551.7 |
| M07 | 3,240,196.4 | 1,132,296.4 | 7,799,312.4 | 7,615,791.4 |
| M08 | - | 1,097,918.0 | 7,123,101.5 | 3,395,830.6 |
| L01 | - | 4,268,256.5 | 2,124,536.8 | 2,419,478.2 |
| L02 | 41,203,988.3 | 7,982,303.1 | 16,817,403.7 | 14,847,193.4 |
| L03 | - | 1,196,237.6 | 7,696,012.5 | 3,844,213.2 |
| L04 | - | 803,147.5 | 19,127,777.3 | 16,538,324.0 |

In summary, it can be said that the heuristics are needed to solve the problem as SCIP is not able to solve the models for medium sized or larger instances to optimality. As the SA is robust, has reasonable solution times and finds satisfying solutions, the SA can be seen as the best of the three algorithms presented in this paper.

5 Conclusion and Future Work

In this work we presented an optimization model to determine optimal locations and dimensions of water tanks in a water distribution system. As the model is a non-convex MIQCP and therefore hard to solve, we implemented three different heuristic solution methods, namely a Simulated Annealing algorithm, a Shuffled Complex Evolution algorithm and a Shuffled Frog-Leaping Algorithm. All three algorithms have in common that they use a hydraulic simulation to evaluate the different solutions within the three algorithms. The computational results show that all heuristics were able to find good solutions. The comparison with SCIP shows that it is necessary to use the heuristics as SCIP was not able to find the optimal solution or even a feasible solution for some of the medium sized or large networks. The SA and the SCE were the best solution methods regarding the quality of the objective function values. When considering the amount of time the algorithms used, it can be seen that the SFLA was the fastest in finding the solutions. However, we recommend to use the SA as it has reasonable solution times and satisfying objective values.

In order to improve the performance of the three heuristics we would like to improve the evolution steps of the SCE and the SFLA in future work. We found that due to the mutations the solutions were often not in the feasible solution space. As the SCE and SFLA have several different parameters, the performance is influenced by the choice of values for these parameters. Even though, we obtained good parameter settings for the algorithms by numerical results, we would like to address finding better parameter settings for the three heuristics in future work. Therefore, we would like to apply a parameter tuner that is able to find a well suited setting for the parameters by using a genetic algorithm, cf. [1].

References

1. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 142–157. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04244-7_14
2. Barakat, S.A., Altoubat, S.: Application of evolutionary global optimization techniques in the design of RC water tanks. *Eng. Struct.* **31**(2), 332–344 (2009)
3. Berthold, T., Heinz, S., Vigerske, S.: Extending a CIP framework to solve MIQCPs. In: Lee, J., Leyffer, S. (eds.) *Mixed Integer Nonlinear Programming*. Springer, New York (2012). https://doi.org/10.1007/978-1-4614-1927-3_15
4. Boulos, P.F., Lansley, K.E., Karney, B.W.: *Comprehensive Water Distribution Systems Analysis Handbook for Engineers and Planners*. MWH Soft, Pasadena (2006)
5. Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P.: On the optimal design of water distribution networks: a practical MINLP approach. *Optim. Eng.* **13**(2), 219–246 (2012)
6. Duan, Q., Sorooshian, S., Gupta, V.: Shuffled complex evolution approach for effective and efficient global minimization. *J. Optim. Theor. Appl.* **76**(3), 501–521 (1993)

7. Eusuff, M., Lansey, K., Pasha, F.: Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Eng. Opt.* **38**(2), 129–154 (2006)
8. Farmani, R., Walters, G.A., Savic, D.A.: Trade-off between total cost and reliability for Anytown water distribution network. *J. Water Resour. Plan. Manag.* **131**(3), 161–171 (2005)
9. Hallmann, C., Suhl, L.: Optimizing water tanks in water distribution systems by combining network reduction, mathematical optimization and hydraulic simulation. *OR Spectr.* **38**(3), 577–595 (2016)
10. Karger, R., Cord-Landwehr, K., Hoffmann, F.: *Wasserversorgung*. Vieweg Verlag, Wiesbaden (2008)
11. Kurek, W., Ostfeld, A.: Multi-objective optimization of water quality, pumps operation, and storage sizing of water distribution systems. *J. Environ. Manag.* **115**, 189–197 (2013)
12. Nickel, D., Lange, M.A., Ayres, A., Schielein, J., Oelmann, M.: *Ökologische und hygienische Kennzahlen im Benchmarking der Wasserversorgung: Empfehlungen aus Sicht des Gewässer- und Gesundheitsschutzes*. Text 16/2013 des Umweltbundesamtes (2013)
13. Rautenberg, J., Fritsch, P., Hoch, W., Merkl, G., Otilinger, F., Weiß, M., Wricke, B.: *Mutschmann/Stimmelmayer Taschenbuch der Wasserversorgung*. Vieweg Verlag, Wiesbaden (2014)
14. Rossman, L.A.: *EPANET 2 - Users Manual*. Water Supply and Water Resources Division, National Risk Management Research Laboratory, Cincinnati (2000)
15. Vamvakieridou-Lyroudia, L.: Tank simulation for the optimization of water distribution networks. *J. Hydraul. Eng.* **133**(6), 625–636 (2007)
16. Zheng, F., Zecchin, A., Newman, J., Maier, H., Dandy, G.: An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems. *IEEE Trans. Evol. Comput.* **21**, 773–791 (2017)

Simulation of Materials



Accelerated Simulation of Sphere Packings Using Parallel Hardware

Zhixing Yang, Feng Gu, Thorsten Grosch, and Michael Kolonko^(✉)

Institute of Applied Stochastics and Operations Research and Institute of Computer Science, University of Technology Clausthal, Clausthal-Zellerfeld, Germany
{Zhixing.Yang, feng.gu, Thorsten.Grosch}@tu-clausthal.de,
kolonko@math.tu-clausthal.de

Abstract. The simulation of dry particle packings and their geometrical properties is of great importance to material sciences. Substantial acceleration of the simulation can be obtained using parallel hardware (GPU), but this requires specialized data structures and algorithms. We present a parallel version of the so-called collective rearrangement algorithm that allows to simulate random close packings of up to several million spherical particles from an arbitrary particle size distribution. The empirical time complexity of our implementation is almost linear in the number of spheres.

1 Introduction

Dry particle mixtures are the basis of materials used in many different fields, e.g. for concrete, for pills and tablets in the pharmaceutical industry, for casting moulds in foundries, or powders for 3D printing. Properties of the particles like shape and size distributions are often crucial for properties of the final material. Therefore material scientists are highly interested in simulating different aspects of particle packings before they are tested in the laboratory, see e.g. [12] for an overview.

Particularly important is the *space filling* Φ of a close random packing of a particle mixture, i.e. the ratio of the space occupied by the particles and the space of a surrounding container needed for the packing of the particles. Particle mixtures with a high space filling are e.g. used for high performance concrete (see e.g. [10]), lower space fillings may be desirable to obtain a particular porosity for filter material.

In this paper we investigate how the space filling $\Phi(f)$ of a close random packing for a given *particle size distribution* (PSD) f can be determined at least approximately. Here, the PSD f gives the relative frequency $f(r)$ of particles with radius r in the mixture for any $r > 0$. Note that this may easily be transformed into a PSD that gives the share $f(m)$ with respect to mass m (or volume) as it is common in material sciences. We restrict ourselves to *spherical* particles, such that the mixture is completely determined by its PSD f .

The run-time of the simulation plays an important role in particular if the overall aim is to find a mixture (i.e. a PSD f^*) with a high space filling $\Phi(f^*)$ or with a space filling close to a given value. Then the determination of $\Phi(f)$ is just the evaluation step in a more complex optimization problem. If we use some heuristic search procedure, like genetic algorithms or ant-algorithms to improve $\Phi(f)$, we typically have to evaluate $\Phi(f)$ for a large number of randomly produced PSD f . Therefore it is crucial that the simulation tool for $\Phi(f)$ is very fast.

There are some analytical (or semi-analytical) approaches to determine $\Phi(f)$, see e.g. [15], or [7]. However, if we consider a polydisperse mixture with a PSD f that covers a broad range of particle radii as it is often needed in applications, then these models seem to be inferior to a computer simulation approach, see [6] and the discussion therein. Here, simulation means to produce a random close packing of a sample of particles on a computer where the sample is drawn according to the given PSD.

Representative samples, e.g. for concrete mixtures, need at least 10^{15} particles with radii ranging from $0.01 \mu\text{m}$ up to 2 mm or more. In [6] a hierarchical simulation system is introduced that allows to break the PSD into fractions which can be simulated by smaller samples. The overall space filling of the mixture is then reconstructed from these partial simulations, including the interaction between fractions. The bottleneck of this approach is the time needed for a valid simulation of a single fraction of the PSD.

Our research therefore concentrates on the speed-up of the simulation of a given sample of spheres. In this paper we report on the implementation of a packing simulation on the GPU (graphical processing unit) that is nowadays available as a general purpose processor for standard desktop computers. Due to the large number of small processors working in parallel, the GPU is able to solve complex problems in very short time, provided the algorithms are adapted to the particular structure of the GPU. One important requirement is that all tasks started in parallel should take approximately the same time as all tasks have to wait for the slowest one. Also, they should only access data in a contiguous part of the memory to enable fast loading.

There are different ways to simulate the packing of spheres. If one aims at simulating a million or more spheres, then more complex approaches that take into account physical properties of the mixture (see e.g. [2]) are not suited because of prohibitive run-times. Instead, very simple models of the particles and their interaction have to be used, the simplest case being spheres with no other than their geometrical properties. In [1] two basic approaches to a geometrical simulation of sphere packings are discussed. The first approach adds the spheres sequentially into a container thus simulating the physical process of sedimentation. The disadvantage of this approach is that the resulting packing is not very dense and may even not be mechanically stable [1, 11].

The second approach called *collective rearrangement* (CR) starts with a random placement of the sample in a container allowing spheres to overlap. Then a mutual repulsion of the spheres is simulated that reduces the overlap. The size of the container is adapted until a valid packing is obtained. This way much

denser packings can be simulated. There are different variants of this idea [1, 5], the difference concerning the initial size of the container and/or of the particles, see details below.

In [9] a simulation based on the CR approach of [5] was examined. The most time consuming step here is to detect overlapping neighbors of a sphere during the repulsion step. On the sequentially working standard computer (CPU), sophisticated data structures had to be used to speed up the process, in particular a combination of *loose octrees* and so-called Verlet lists. It turned out that this structure is not suited for parallel computing as it uses lists with dynamically changing lengths. We therefore had to develop a completely different approach for the iterative reduction of the overlaps which is described in detail below.

With this new algorithm, we are now able to simulate a particle packing on the GPU much faster than on a standard CPU with a quality that is comparable to the sequential version. E.g., the simulation of 1 million spheres from a typical PSD that took about 30 h on an up-to-date sequential CPU, now takes less than 1.5 min on the GPU.

We describe our core data structure used to keep the positions of spheres in Sect. 2. In Sect. 3 we discuss the initial overlapping placement of the sample in a container and in Sect. 4 we give more details about the collision detection and collective rearrangement process. In Sect. 5 some experimental results are reported along with pictures of the packings and their interior produced by visualization tools developed in our group in close cooperation with the simulation. The final Sect. 6 contains some concluding remarks and an outlook on future research.

2 An Adaptive Grid as Main Data Structure

We start with an abstract description of the data structure we use to keep the spheres and their present location in the container. We assume that we are given a sample of N spheres which we identify with the numbers $\{0, \dots, N - 1\}$. For $i = 0, \dots, N - 1$, let $r_i > 0$ be the radius and $d_i = 2r_i$ the diameter of the i -th sphere. We are using a cubic container with side length L_0 into which the spheres are to be packed, for a generalization see below. We assume that the origin lies in a corner of the container and that $\mathbf{P}_i := (x_i, y_i, z_i)$ is the present position of the center of the sphere i with respect to this origin.

The data structure for the container and the spheres must allow to identify all possibly overlapping neighboring spheres, and the number of steps necessary should be about the same for all spheres in order to run efficiently on the GPU. We therefore use a recursive grid similar to a loose octree (see [9, 14]) for the sphere locations.

We iteratively divide the initial cubic container with side length L_0 into eight sub-containers of equal size as in an octree. On the l -th level, we then have 8^l sub-cells each with side length $L_l := L_0/2^l$ where $l = 0, 1, \dots$. We insert sphere i with diameter d_i into level l iff

$$\frac{1}{2}L_l \leq d_i < L_l, \quad (1)$$

thus sphere i is inserted on the first level (starting with the top level 0) where its size would fit into a cell. Sphere i is inserted into that cell that contains its present center \mathbf{P}_i . Note that we do not require that the sphere fits completely into one of the cells as it would be the case in a strict octree, in this respect the structure resembles the loose octree.

The index l^* of the lowest level needed is determined by the minimal diameter d_{min} occurring in the mixture, we must have

$$\frac{L_0}{2^{l^*+1}} \leq d_{min} < \frac{L_0}{2^{l^*}},$$

therefore $l^* = \lceil \log_2 \frac{L_0}{d_{min}} - 1 \rceil$ resulting in a total number of

$$C(l^*) := (8^{l^*+1} - 1)/7 \quad (2)$$

cells in the grid. If the memory requirement for these cells is too large, l^* is set to a reasonable value and all spheres with $d_i < L_0/2^{l^*}$ are added to the lowest level l^* .

Though logically the grid data structure is defined in a recursive way, we use a non-recursive implementation suited for the GPU. Its central part is a sphere-cell-list that gives all spheres belonging to one cell. On the CPU, this could be accomplished by a linked list, but the necessary dynamic memory allocation can cause massive performance degradation on parallel hardware. There are two approaches to handle such a list on the GPU. One is to use a bin with a predefined size for each cell. In order to make this work, there has to be a strict upper bound for the number of spheres belonging to one cell. When there is no overlap between spheres, this bound can be derived from the so-called kissing number, see [14]. But in our case, spheres can heavily overlap with each other during the collective rearrangement. Hence, no reasonable upper bound for the number of spheres in one cell is available, therefore the bin approach cannot be applied without dynamic memory allocation.

The second approach, which we use here, is to build the hierarchical grid using sorting as described in [3]. First, we keep the N spheres with all their data in different arrays of fixed length N . The i -th element of the r -array contains the radius r_i (as a number of type `float`), the x -, y -, and z -arrays contain in their i -th elements the present position x_i, y_i, z_i of the i -th sphere center and the c -array contains in its i -th element the number c_i of the grid cell the i -th sphere is presently located in. So the data object ‘sphere’ is represented by a cut through all these arrays, see Fig. 1 where the data values of sphere no. 6 are marked.

To enable fast access to all spheres in one grid cell, we then add a copy of the radius- and the x -, y -, z -arrays which are *sorted* according to their cell numbers, such that all spheres residing in one cell fill a contiguous part in each array. We call these the *sorted- r* , *sorted- x* etc. array. We have to add two arrays of length equal to the number of cells $C(l^*)$. Element k of the first array gives the index in the sorted arrays where cell k starts and the second gives the index of the last sphere in cell k . Finally we need an *idx*-array of length N that for

each index of the sorted arrays gives the index of the original unsorted sphere arrays, see Fig. 1 for an example. The hierarchy of the grid cells is also stored in a linear fashion, so that the grid cell numbers of one level occupy a contiguous part in memory.

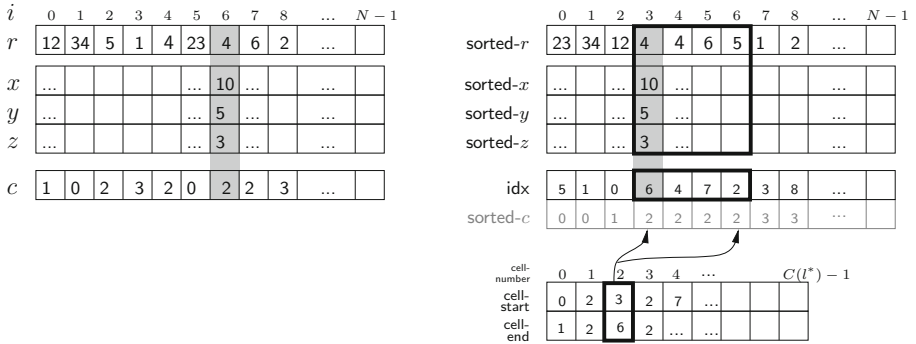


Fig. 1. A grid data structure for the GPU. Sphere $i = 6$ has radius $r_6 = 4$, its center is located at $\mathbf{P}_6 = (x_6, y_6, z_6) = (10, 5, 3)$ in grid cell no. $c_6 = 2$. In the arrays sorted according to grid cells, cell no. 2 starts with the sphere now residing in the sorted entry no. 3 and ends with the sphere in entry 6. From the idx-array we see that these are the spheres no. 6, 4, 7, 2. The sorted c -array is not needed in the implementation.

Note that with this data structure we may implement different types of containers. If the container has a cubic shape its outer walls will coincide with the boundaries of the hierarchical grid. For non-cubic containers, the grid size L_0 must be chosen such that the container fits into the grid, e.g. L_0 could be chosen such that the grid is a bounding box for the container. If we use hard walls, then all spheres have to be placed completely inside the container, thus placement in the grid cells at the outer border of the container has to be treated differently from the ‘inner’ cells, see below. In the case of *periodic* walls, each container wall is identified with the opposite wall, such that a sphere that juts out at one side occupies the corresponding space at the opposite side of the container, particular care has to be taken with spheres in the corners of the container, see Fig. 6 below. A periodic container models a spatial unit of a larger space that is tiled with identical replications of the container. In this case, spheres in the border grid cells are treated as in all other cells. Note that we may also use a combination of hard and periodic walls and rather arbitrary shapes of containers (e.g. cylindrical). The only restriction to the shape of the container is that there must be an efficient way to determine whether a sphere intersects the wall of the container and how the sphere must be moved to reduce the intersection to 0. Note that this may rule out some irregular shapes, e.g. with small niches in the boundary.

3 Initial Placement

Initially, the spheres are placed randomly in the container. The subsequent collective rearrangement steps need overlapping spheres in order to obtain a dense packing in the end. Therefore, the initial container size L_0 is chosen so small that most spheres will overlap at least with one of their neighbors. Though it might happen that there are isolated spheres without any overlap after the initial placement, they will be overlapped by other spheres after a few steps of the collective rearrangement as spheres will be pushed into the empty neighborhood of an isolated sphere.

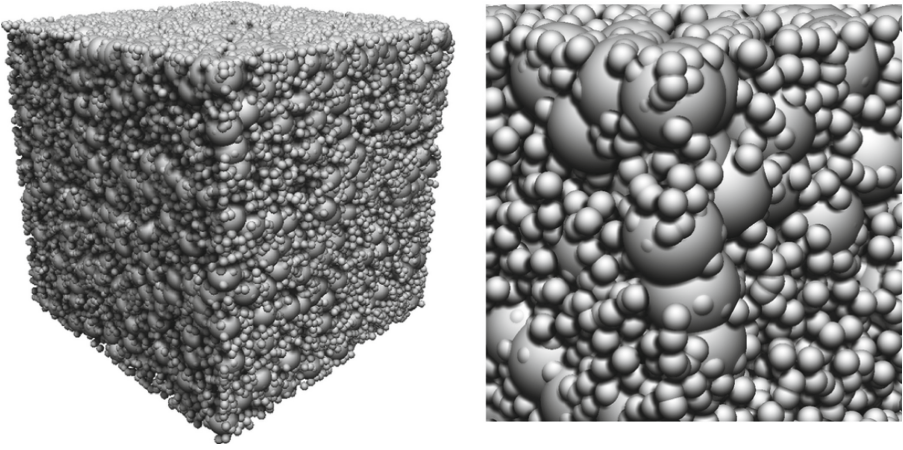


Fig. 2. The initial placement of a bidisperse mixture in a container with hard walls, the detail on the right hand side shows that the spheres are strongly overlapping.

We start with an initial container volume L_0^3 equal to the net volume of the spheres

$$V_s := \sum_{i=1}^N \frac{1}{6} \pi d_i^3. \quad (3)$$

Technically, we use a container of fixed side length $L_0 = 1$ and scale the radii of the spheres by a factor δ_0 such that

$$V_s(\delta_0) := \sum_{i=1}^N \frac{4}{3} \pi (\delta_0 r_i)^3 = \delta_0^3 V_s = L_0^3 = 1, \quad (4)$$

i.e. $\delta_0 := \sqrt[3]{V_s}$.

For a random placement of the spheres, the x -, y - and z - arrays are filled with independent random numbers distributed uniformly over the interval $[0, L_0] = [0, 1]$. If we use hard walls, we have to check whether the spheres are within

the walls of the container, otherwise we have to correct the placement. For a rectangular container, we restrict the random center for sphere i to $[r_i, 1 - r_i]$ in each dimension, then i lies within the walls. Similarly, for non-rectangular containers like cylinders we have to adapt the uniform distribution to the shape. The only requirement is, that the container fits into the grid.

4 Collective Rearrangement

The collective rearrangement (CR) is the central part of the algorithm. It imitates a repulsion among the spheres and gradually reduces the total overlap in the packing, but, as long as the container is too small, some overlap will remain. We determine the average overlap per sphere after each round of repulsion as a measure for the quality of the packing. If it has not improved for several rounds, the placement of the spheres has attained some kind of equilibrium for the present container and repulsion is stopped. Typically, the remaining overlaps (as well as the remaining free space) are then spread more evenly over the container than before. Then the container is slightly enlarged and the CR phase with repeated rounds of repulsion is started again. CR phase and subsequent container enlargement are repeated until finally the remaining overlap is negligible. This way, the packing is always a random dense arrangement and is slowly transformed into a physically valid one. We shall now describe the technical details of the collective rearrangement on the GPU.

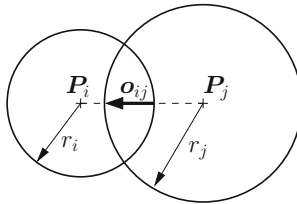


Fig. 3. The overlap of the active sphere i with sphere j .

During a repulsion step a sphere i is taken as *active* sphere and the *displacement vector* $\mathbf{O}(i)$ is determined for this sphere, i.e. the vector by which i should be moved in order to reduce its overlap. Let $D_{ij} = \|\mathbf{P}_i - \mathbf{P}_j\|$ be the distance of the two sphere centers at \mathbf{P}_i and \mathbf{P}_j . The spheres i and j overlap whenever $D_{ij} < r_i + r_j$. We define the (raw) *overlap vector* \mathbf{o}_{ij} of sphere i with sphere j as

$$\mathbf{o}_{ij} := (\mathbf{P}_i - \mathbf{P}_j) \left[\frac{r_i + r_j}{D_{ij}} - 1 \right]^+, \quad (5)$$

where $[a]^+ = \max\{a, 0\}$, see Fig. 3. \mathbf{o}_{ij} is the displacement vector that, if added to \mathbf{P}_i , would reduce the overlap with sphere j to zero. Note that $\mathbf{o}_{ij} = -\mathbf{o}_{ji}$. If the two spheres do not overlap, we see from (5) that $\mathbf{o}_{ij} = 0$. However, this overlap

is not well defined if the two sphere centers coincide as then the denominator in (5) is equal to 0. Though theoretically this will only happen with probability 0, due to the limited numerical precision of our calculations, we have to check the centers before evaluating (5) and, in case they are identical within the precision used, add a small random displacement to one of the spheres.

As sphere i will, in general, overlap with more than one neighbor, we add a weight to this raw overlap vector \mathbf{o}_{ij} that reflects the relative importance the overlap with sphere j has for sphere i . More precisely, let I_i denote the set of spheres j that have a positive overlap $\|\mathbf{o}_{ij}\| > 0$ with sphere i . Let v_{ij} be the volume of the overlap of the two spheres that is given by (see e.g. [13])

$$v_{ij} = a \cdot (r_i + r_j - D_{ij})^2 (D_{ij}^2 + 2D_{ij}(r_i + r_j) - 3(r_i - r_j)^2) / D_{ij}$$

if $|r_i - r_j| < D_{ij} < r_i + r_j$, i.e. if there is an overlap but neither of the spheres is completely contained in the other. Here, $a = \pi/12$ is a constant that may be dropped below. If $D_{ij} < |r_i - r_j|$, then $v_{ij} = a \cdot 16 \min(r_i, r_j)^3$ is the volume of the smaller sphere. Let

$$V_{i\bullet} = \sum_{j \in I_i} v_{ij}$$

be the total volume of pairwise overlaps that sphere i has with other spheres. Note that this need not be the exact total overlap as we do not take into account the overlap of more than two spheres. Then $v_{ij}/V_{i\bullet}$ reflects the relative importance the overlap with sphere j has for sphere i . As we want to move both overlapping spheres away from each other by half of their overlap, we use as final displacement vector $\mathbf{O}(i)$ for sphere i the sum of all weighted overlap vectors divided by 2:

$$\mathbf{O}(i) = \frac{1}{2} \sum_{j \in I_i} \frac{v_{ij}}{V_{i\bullet}} \mathbf{o}_{ij}, \quad i = 0, \dots, N-1. \quad (6)$$

Technically, all spheres are becoming active spheres in parallel and therefore also the calculation of the vectors $\mathbf{O}(i)$ must be done in parallel. Each sphere i has its own GPU thread that compares i to all spheres j with *larger or equal* radius $r_j \geq r_i$ and collects the overlaps.

The thread of sphere i first compares i to spheres j with *equal* radius and adds the vector $\frac{1}{2}v_{ij}\mathbf{o}_{ij}$ to the i -th element of a *displacement array* \mathbf{O}^{\geq} , see Fig. 4. \mathbf{O}^{\geq} consists of three arrays of length N that hold the x -, y - and z -components of the vector. Also, the volume v_{ij} of the overlap is added to the i -th component of a *volume array* \mathbf{V}^{\geq} . Sphere i is then compared to spheres k with *larger* radius $r_k > r_i$. The resulting weighted overlap $\frac{1}{2}v_{ik}\mathbf{o}_{ik}$ and v_{ik} are also added to the i -th components of \mathbf{O}^{\geq} and \mathbf{V}^{\geq} , respectively.

The displacement $\frac{1}{2}v_{ki}\mathbf{o}_{ki} = -\frac{1}{2}v_{ik}\mathbf{o}_{ik}$ should also be given to the larger partner k of this overlap, but if we try to write this to the k -th components of \mathbf{O}^{\geq} and \mathbf{V}^{\geq} , this may collide with other threads from spheres m with $r_m < r_k$ that also have some overlap with k and are active at the same time. These operations therefore have to be sequentialized by making them *atomic* which

slows down the process as many threads might have to wait. In order not to degrade the overall efficiency of the repulsion, these write operations do not access the arrays \mathbf{O}^{\geq} and \mathbf{V}^{\geq} . Instead we use additional arrays $\mathbf{O}^>$, $\mathbf{V}^>$ and add $-\frac{1}{2}v_{ik}\mathbf{o}_{ik}$ to the k -th component of $\mathbf{O}^>$ and v_{ik} to $\mathbf{V}^>$. Though some of these write operation may have to wait, the access to \mathbf{O}^{\geq} and \mathbf{V}^{\geq} is not affected, as there the i -th component may only be accessed by the thread of sphere i , see Fig. 4 for an illustration.

After all spheres i have been compared in this way to all other spheres j with $r_j \geq r_i$, we add component-wise

$$\mathbf{O}^{\geq} = \mathbf{O}^{\geq} + \mathbf{O}^>, \quad \text{and} \quad \mathbf{V}^{\geq} = \mathbf{V}^{\geq} + \mathbf{V}^> ,$$

then the i -th component of \mathbf{V}^{\geq} is equal to $V_{i\bullet}$, by which we then divide the i -th entry of \mathbf{O}^{\geq} to obtain $\mathbf{O}(i)$ as defined in (6).

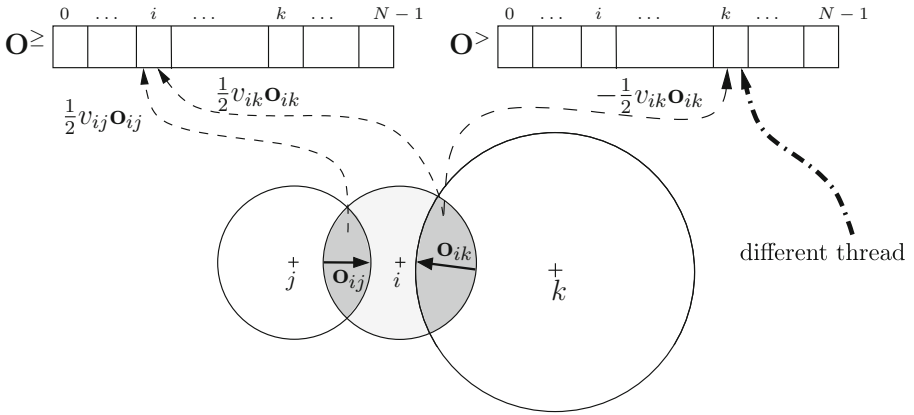


Fig. 4. Sphere i overlaps sphere j with $r_i = r_j$ and sphere k with $r_i < r_k$. The (weighted) overlap $v_{ik}\mathbf{o}_{ik}$ of the active sphere i with a larger sphere k is shared equally between the two spheres i and k . The overlap $v_{ij}\mathbf{o}_{ij}$ with a sphere j of equal size is only registered for sphere i . Access to array $\mathbf{O}^>$ is shared by several threads and must therefore be sequentialized by so-called atomic operations.

Finally, the total displacement vectors $\mathbf{O}(i)$ are added to the positions \mathbf{P}_i in the original unsorted vectors (using the idx -arrays introduced above). The repulsion step is completed by correcting the new positions $\mathbf{P}_i + \mathbf{O}(i)$ in case wall restrictions are violated. Then a new repulsion step can start with the new sphere positions.

Note that we compare each sphere only with spheres of the same or larger radius as suggested in [14]. This prevents visiting pairs of spheres with different radii twice. Only those pairs with equal radii will be visited twice, once when i is the active sphere and once when j is active. This is the reason why in this case

the overlap $\frac{1}{2}v_{ij}\mathbf{o}_{ij}$ is written only once to the i -th components of the overlap array \mathbf{O}^{\geq} .

We do not need to check spheres for collision that are too far away from the active sphere i . Therefore we may restrict the search to certain cells of the grid. We first check the spheres on the grid level $l(i)$ on which i is stored. From the definition of the grid in (1) we see, that on this level only spheres from cell c_i , in which sphere i resides, and the $3 \times 3 \times 3 - 1 = 26$ cells surrounding c_i can have a positive overlap with i and from these only the ones with radius at least as large as r_i are considered. Then, we only have to check spheres from cells on the next upper level $l(i) - 1$ as spheres on levels $> l(i)$ have a radius smaller than r_i . Again, only spheres from the cell on that level that contains c_i and its 26 neighbors have to be searched and the same applies to all levels up to the uppermost level $l = 0$. Note that there is a slight deviation from this for the cells on levels 0 and 1 and for cells at the boundary of the container in case we have hard bounds, as these cells have less than 26 neighbors. The important fact here is that we move through the tree upwards which results in a very limited number of cells that have to be searched to detect all overlaps. Using the sorted arrays introduced above, all spheres in a cell (i.e. their present position and radius) can be easily accessed in parallel.

As mentioned above, for non-overlapping spheres as they were used in [14], the so-called kissing number bounds the number of possible collisions of a sphere with spheres of equal or larger radius. In our case, where spheres may overlap, there is no strict limit to the number of spheres to be checked. In our experiments however (see Sect. 5), we typically found 30–40 spheres in each cell. Therefore, the number of neighbors that actually had to be checked for overlap was less than this number times the number of neighboring cells, i.e. less than $40 \times 27 = 1080$ in most cases, as it is desirable for parallel processing.

During the collection of the overlap vectors, we also sum up the pairwise overlaps $\|\mathbf{o}_{ij}\|$ to determine the average overlap per sphere as a measure for the quality of the present packing.

The repulsion steps are repeated until the average overlap stops improving, then a collective rearrangement phase is considered complete and the (relative) size of the container is enlarged by a scaling of the sphere radii. The present scaling factor $\delta_k, k \geq 0$, (see (4) for δ_0) is changed from δ_k to $\delta_{k+1} := (1 - \alpha)\delta_k$, i.e. the spheres are shrunked by 100α percent, resulting in new empty space between the spheres which is then filled during the next repulsion step. α is depending on the present average overlap and is reduced during the process to avoid spheres without any overlap.

CR phase and shrinking of spheres are repeated until the remaining average overlap seems negligible. Then the spheres may be enlarged again to produce new overlap and the whole process may start from the beginning taking the result of the preceding stage as a starting placement. This may be repeated several times resulting in a very dense packing that is in good accordance to laboratory results with real mixtures, see Sect. 5 below.

Let us close this section by noting that the sequential version of the rearrangement seems physically more correct than the parallel one. In the sequential procedure, for each single active sphere i the overlap with its neighbors at their present position is determined and i is then moved to its new position. After that, the next sphere becomes active and is moved using the possibly updated, valid position of its neighbors. The parallel rearrangement step, however, takes into account the position of all neighboring spheres *before* the joint rearrangement takes place. Therefore, a sphere may move away from a neighbor which itself moves to another direction due to its overlap with other spheres. This ‘error’ is made up for by sharing the displacement between the pairs of overlapping spheres and by the large number of repetitions we can afford with parallel computations.

It might though happen that spheres start to oscillate during the iterations. Such situations have to be handled similarly to constellations where sphere centers get too close together.

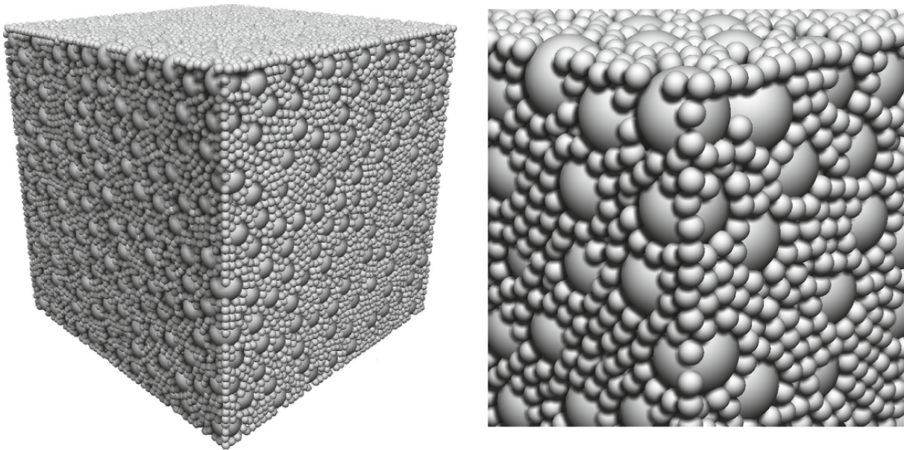


Fig. 5. The final packing of the mixture Bidisperse2 of Fig. 2, the detail on the right hand side shows that overlaps have vanished during CR.

5 Experimental Results

First we checked the quality of the simulated packings. We compared the simulated space fillings with laboratory results for 6 different mixtures of glass spheres, Table 1 shows that except for the mixture Bidisperse1, simulation was very close to the real values.

Let us look more closely at mixture Bidisperse2, it consists of spheres with two different radii: 40.36% of the mass of the mixture have radius 0.940 mm and 59.64% have radius 6.003 mm, see [6] for details of the other mixtures. Figure 2 shows an initial placement of a sample of 50,000 spheres of Bidisperse2 in a

Table 1. Simulated and real space filling of six different mixtures named “Fuller1”, “Fuller2” etc. Laboratory results are means from seven repetitions, the relative error is defined as (laboratory - simulation)/laboratory.

| Mixture | Fuller1 | Fuller2 | Fuller3 | Bidisperse1 | Bidisperse2 | Tridisperse |
|------------|----------|----------|----------|-------------|-------------|-------------|
| Laboratory | 0.7158 | 0.7530 | 0.7899 | 0.6950 | 0.7591 | 0.6953 |
| Simulation | 0.71722 | 0.7652 | 0.7901 | 0.7231 | 0.7541 | 0.6955 |
| Rel. error | -0.00198 | -0.01620 | -0.00025 | -0.04043 | 0.00658 | -0.00028 |

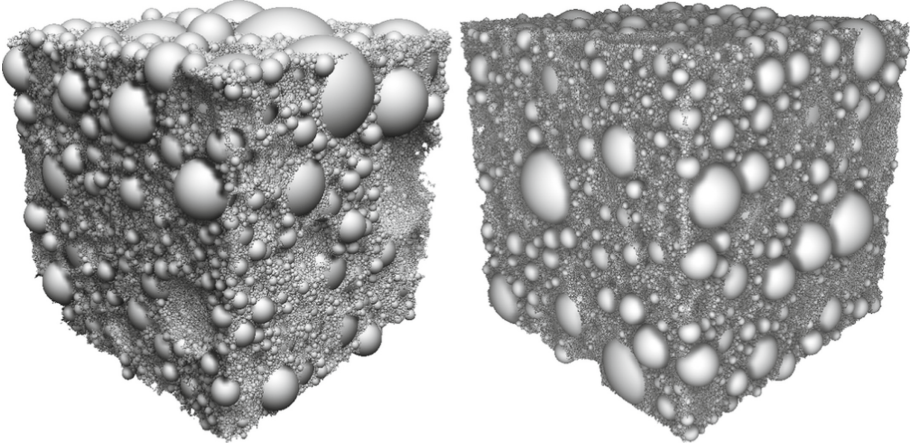


Fig. 6. The final packings of samples from a cement mixture with periodic boundaries, the left hand picture shows a 1 million sample, the right hand side a 10 million sample.

container with hard walls. The overlap, that can be seen in this figure, is reduced by the parallel CR algorithm until a valid packing is obtained as shown in Fig. 5.

For this mixture, the former sequential version of the CR algorithm resulted in a space filling of 73.25% (relative error -0.03504 compared with the laboratory result) after a run-time of 560.16 s on a Intel Core i7-6700 CPU. The run-time for the parallel version was 26.27 s on a GEFORCE® GTX 1070 GPU with a relative error of only 0.0065 (see Table 1). So the parallel version in this case is more than 20 times faster and more precise.

The next example is a cement mixture as it is used in real concrete with radii ranging from $0.04 \mu\text{m}$ to $83.89 \mu\text{m}$. In this case the simulated space filling of 86.25% is much higher than the real space filling (around 73%), as our simulation at present does not take into account the agglomeration of small particles that tend to form ill-shaped agglomerates decreasing the overall space filling of the mixture. A way to simulate this on the sequential CPU is described in [8].

Nevertheless, we use this mixture to demonstrate the speed-up of the simulation with the GPU for samples of larger size. The left hand picture in Fig. 6 shows the final packing of 1 million spheres drawn from the cement PSD in a

container with periodic boundaries. The space filling was 86.4178%. The periodicity of the container walls is clear to see, spheres that jut out from the container on the rear sides take up space on the opposite front sides. The right hand side of Fig. 6 shows a packing with 10 million spheres from the same PSD with a space filling of 85.7916%. The different simulation results for 1 and 10 million indicate that a sample of 1 million is probably not representative for that highly polydisperse PSD.

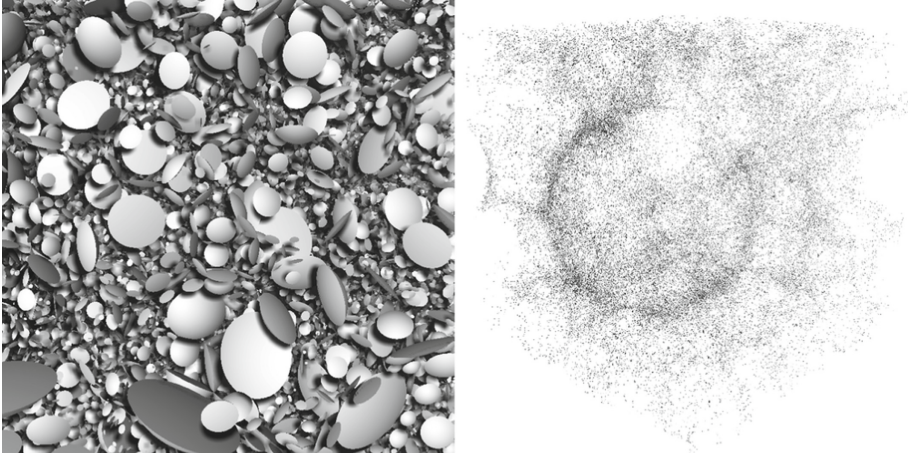


Fig. 7. A visualization of the internal overlaps for the 1 million cement sample. The left hand picture shows a detail of the lens-shaped overlaps in the initial placement, the right hand picture shows all remaining overlaps near the end of the CR process. This picture also reveals that the sample contains a huge sphere in the middle with a large number of small spheres still having small overlaps with the big one.

The CR process for these samples can be visualized with the tools that are developed in a companion project to our simulation (‘Virtual Microscope’, see [4]). Figure 7 shows how the overlaps vanish during the CR process.

The run-time for the simulation of the 1 million sample with the sequential version of the program took 134 131 s which is more than 37 h. The parallel version on the GPU yielded the same space filling in 83 s, i.e. the parallel version is 1616 times faster than the sequential version. The run-time of the sequential version increases rapidly with an increasing number of spheres, whereas for the parallel version the increase is quite moderate. Figure 8 shows the average run-times for the first repulsion step for different number of spheres. Though the time complexity of this step is inherently quadratic (each sphere has to be compared to every other sphere), our algorithm allows an almost linear complexity on the GPU.

Sample sizes for the GPU are therefore limited not so much by the run-time of the simulation but by its memory requirements. Simulation (and visualization) on our GPU GEFORCE® GTX 1070 (8 GB memory) can handle up to 10 million spheres.

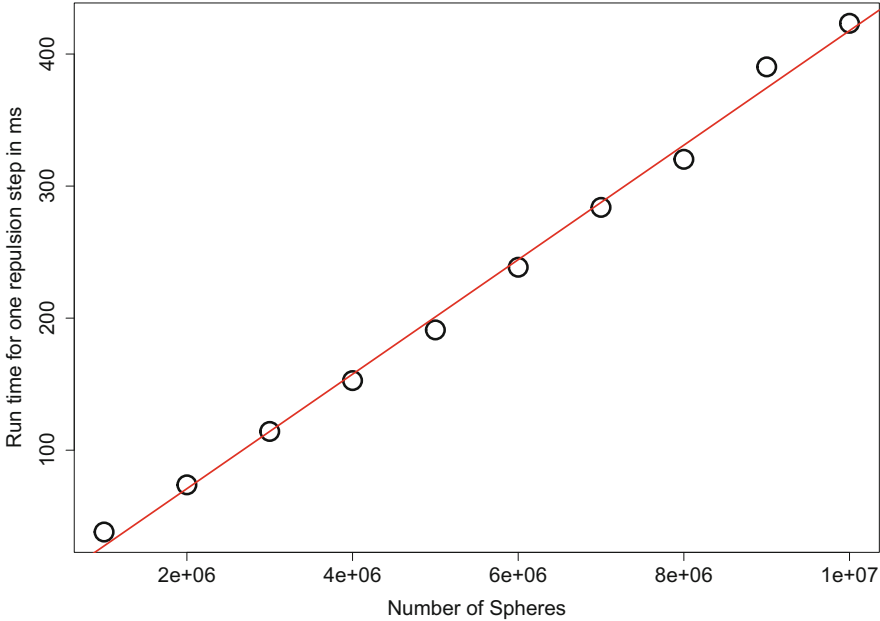


Fig. 8. The run-times of a repulsion step are almost linear in the number of spheres.

6 Conclusion

The GPU can be used to speed-up simulation of large particle samples considerably. For the collective rearrangement algorithm that generates particularly dense packings, data structures and displacement of the spheres had to be redesigned in order to make full use of the parallel capacity of the GPU. Though the parallel version of the collective rearrangement is even less similar to the physical process of particle packing than the sequential version, the results are at least as good in much shorter time.

Future research will concentrate on non-spherical particles that are constructed from a rigid collection of smaller spheres. The GPU will allow to increase the number of such particles in a simulation and to obtain more realistic results.

Also we will extend the cooperation with the visualization co-project (see [4]) to get more insight into internal processes during rearrangement.

Acknowledgments. Zhixing Yang is supported by the Dres. Edith und Klaus Dyckerhoff-Stiftung, grant number T218/26441 /2015. Feng Gu receives a scholarship of the Simulation Science Center Clausthal-Göttingen within the project ‘Virtual Microscope’.

References

1. Bezrukov, A., Bargiel, M., Stoyan, D.: Statistical analysis of simulated random packings of spheres. *Part. Part. Syst. Charact.* **19**(2), 111–118 (2002)
2. Cundall, P.A., Strack, O.D.L.: A discrete numerical model for granular assemblies. *Géotechnique* **29**(1), 47–65 (1979)
3. Green, S.: Particle simulation using CUDA. *NVIDIA Whitepaper* **6**, 121–128 (2010)
4. Gu, F., Yang, Z., Kolonko, M., Grosch, T.: Interactive visualization of gaps and overlaps for large and dynamic sphere packings. In: Hullin, M., Klein, R., Schultz, T., Yao, A. (eds.) *Vision, Modeling & Visualization*. The Eurographics Association (2017)
5. He, D., Ekere, N.N., Cai, L.: Computer simulation of random packing of unequal particles. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* **60**(6), 7098–7104 (1999)
6. Kolonko, M., Raschdorf, S., Wäsch, D.: A hierarchical approach to simulate the packing density of particle mixtures on a computer. *Granular Matter* **12**(6), 629–643 (2010)
7. de Larrard, F., Sedran, T.: Optimization of ultra-high-performance concrete by the use of a packing model. *Cem. Concr. Res.* **24**(6), 997–1009 (1994)
8. Mock, S.: Simulation von hoch polydispersen zufällig dichten Partikelpackungen unter Berücksichtigung der Agglomeration im Feinstkornbereich. Ph.D. thesis, Institute for Applied Stochastics and OR, University of Technology Clausthal, Germany (2015)
9. Raschdorf, S., Kolonko, M.: A comparison of data structures for the simulation of polydisperse particle packings. *Int. J. Numer. Methods Eng.* **85**(5), 625–639 (2011)
10. Schmidt, M., Fehling, E., Geisenhanslüke, C. (eds.): *Ultra High Performance Concrete (UHPC) - Proceedings of the International Symposium on UHPC*, Schriftenreihe Baustoffe und Massivbau - Structural Materials and Engineering Series, vol. 3. Kassel University Press, Kassel (2004)
11. Topic, N., Pöschel, T.: Steepest descent ballistic deposition of complex shaped particles. *J. Comput. Phys.* **308**, 421–437 (2016)
12. Torquato, S.: *Random Heterogenous Materials: Microstructure and Macroscopic Properties*, Interdisciplinary Applied Mathematics, 2nd edn., vol. 16. Springer, New York (2006)
13. Weisstein, E.W.: Sphere-sphere intersection. In: *MathWorld*. Wolfram Research, Inc. (2007) <http://mathworld.wolfram.com/Sphere-SphereIntersection.html>
14. Weller, R., Frese, U., Zachmann, G.: Parallel collision detection in constant time. In: Bender, J., Dequidt, J., Duriez, C., Zachmann, G. (eds.) *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association (2013)
15. Yu, A.B., Standish, N.: Estimation of the porosity of particle mixtures by a linear-mixture packing model. *Ind. Eng. Chem. Res.* **30**(6), 1372–1385 (1991)



MC/MD Coupling for Scale Bridging Simulations of Solute Segregation in Solids: An Application Study

Hariprasath Ganesan¹, Christoph Begau¹, and Godehard Sutmann^{1,2}(✉)

¹ Interdisciplinary Centre for Advanced Materials Simulation (ICAMS),
Ruhr-University Bochum, 44801 Bochum, Germany

² Jülich Supercomputing Centre,
Forschungszentrum Jülich, 52425 Jülich, Germany
g.sutmann@fz-juelich.de

Abstract. A parallel hybrid Monte Carlo/Molecular Dynamics coupled framework has been developed to overcome the time scale limitation in simulations of segregation of interstitial atoms in solids. Simulations were performed using the proposed coupling approach to demonstrate its potential to model carbon segregation in ferritic steels with a single dislocation. Many simulations were carried out for different background carbon concentrations. This paper is a first step towards understanding the effect of segregation of interstitial atoms in solids and its influence on dislocation mobility in external fields. To this end, we carried out MD simulations, where shear forces were applied to mechanically drive screw dislocation on configurations with segregated carbon atoms. The results are compared with a reference system containing homogeneously distributed carbon atoms where the influence of segregated carbon on dislocation mobility could be observed. Simulation results gave qualitative evidence that the local concentration of interstitial solutes like carbon provides a significant pinning effect for the dislocation.

Keywords: Solute segregation · Parallelization · Cottrell atmospheres

1 Introduction

Macroscopic quantities like yield strength in crystalline metallic materials are governed by mechanisms occurring on atomistic length scales, e.g., solute segregation. Increased yield strength due to retardation of dislocation mobility in strain aged steel samples is attributed to the hindrance induced by segregated interstitial solutes like carbon (C) [1], which is an important alloying element in ferritic steels and which form a so-called Cottrell atmosphere around dislocations. Wilde and co-workers provided the proof of Cottrell atmospheres from an experimental standpoint using a 3D reconstruction technique [2]. Some of the earlier theoretical models proposed by Zhao et al. [3] extending the Cottrell theory towards carbon atmosphere formation in ultra-low carbon steels during

strain aging. The theoretical model further included fluctuations in carbon concentration, saturation at dislocations, and carbon segregation to grain boundaries. Later, it was compared with strain ageing experiments and an increase in carbon segregation to grain boundaries with decreasing grain size could be observed. In one of their later works [4], two types of carbon atoms were mentioned participating in the Cottrell cloud-defect interaction. The first type refers to atoms located below the dislocation line and which are tied to the dislocation having lost the ability to freely move, whereas the second type of atoms refers to atoms approaching the dislocation with the capability to pin it. Waterschoot et al. [5] performed experimental studies to observe strain ageing in the dual-phase steel and different classified stages such as Cottrell atmosphere formation, precipitation of existing solutes and the contribution from the martensite phase. Berbenni et al. [6] performed static ageing uniaxial strain tension experiments, and simulated bake hardened and homogeneous strain-hardening curves using the micromechanical approach for aluminum-killed bake hardened steel samples. However, the representative volume element (RVE) model used in the simulations consider linear elastic and isotropic assumptions for every single crystal in the model. Further, the authors included various physical parameters such as dislocation density, initial carbon content, the volume fraction of clusters in their micro-mechanical model to predict the macroscopic behaviour of different pre-strained samples.

To observe the elementary mechanism in great detail, we need atomistic simulations, while it is tedious to track the contribution of a single Carbon atom or clusters of carbon in pinning the individual dislocations using experimental studies. For atomistic modeling, we need more insight into the single crystal before reaching polycrystals with the higher dimensional defects. Khater et al. [7] performed MD simulations to estimate the local stress exerted on the dislocation due to the carbon atoms. A single carbon atom is placed at different octahedral sites and its interaction with the edge dislocation gliding in the $\{1\ 1\ 0\}$ and $\{1\ 1\ 2\}$ slip planes is calculated. Introducing carbon atoms in the iron matrix causes tetragonal lattice distortion and certain tetragonal distortion axes which show a strong modification in the local lattice friction. The computed stresses are later used to describe the interaction strength on the continuum level. Some of the earlier works in the past [8–10] developed an MD-MC coupling approach to overcome the time scale limitation of the rare event physics, i.e., segregation. Those approaches, however, considered several assumptions such as biased sampling, localised trial move, pre-computation of binding energies or saturation-limit assumption and therefore they are problem specific. In the present work, we overcome those limitations to model interstitial solute segregation in single crystalline metallic materials. We present an application example of carbon Cottrell atmosphere formation in ferritic iron with the major emphasis on the investigation of the influence of Cottrell cloud on dislocation mobility. Also, the role of carbon local concentration on dislocation pinning is investigated using MD simulations.

The modeling of the segregation process by atomistic simulation methods, e.g., molecular dynamics (MD), in large systems is a particularly challenging problem. Since the diffusion activation energy for C in Fe is larger than 100 kJ/mol, diffusion at room temperature and below is a rare event compared to the timescale of a typical MD simulation, which is governed by time steps in the order of femtoseconds. Therefore a large number of atom oscillations is needed in a local environment until an energy barrier crossing takes place. Considering the complexity of a system including its chemical environment and strain field due to crystal defects these processes cannot be explored with traditional deterministic methods. In principle, the same applies to local Monte Carlo methods, which might also lead to confinement of solute atoms to a local energy basin for a large number of trial moves. As a solution, we propose a coupled framework of MD and Monte Carlo using virtual atoms (V) to model interstitial solute segregation in a crystal with arbitrary defects. Non-local trial moves are performed which sample configuration space more efficiently and which can overcome local energy barriers. The approach is valid for interstitial atoms in solids, and it is demonstrated to work for carbon segregation in ferritic iron including screw dislocations. Upon randomly sampling the simulation domain without introducing any bias, the results for $T = 0\text{ K}$ simulations show a high carbon concentration near the dislocation core, i.e., the formation of a Cottrell atmosphere, which is observed in experiments [2].

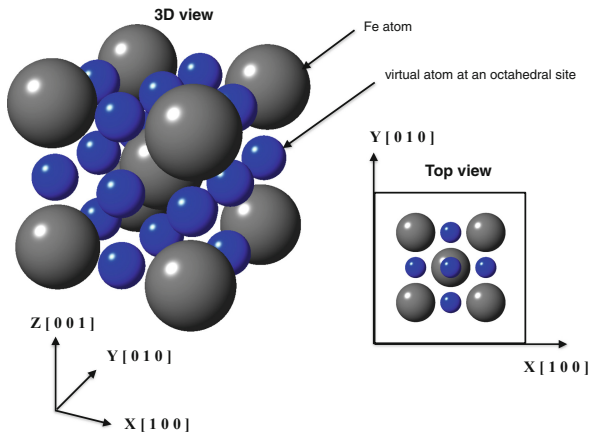


Fig. 1. A 3D perspective projection of a bcc unit cell showing ‘Fe’ atom and ‘virtual atom’ (blue) introduced at all the octahedral site. Also, the XY plane (top view) is shown further for clarity. (Color figure online)

2 Method

We apply a hybrid formulation of Molecular Dynamics (MD) or Molecular Statics (MS) for $T = 0\text{ K}$ and Monte Carlo (MC) simulation protocols. The link

between MD and MC is achieved using *virtual atoms*. These correspond technically to pseudo-atoms located at all octahedral sites in the host crystalline matrix, i.e., α -Fe. First principles studies show that interstitial solute elements, e.g., carbon, diffuse to or prefer to stay at octahedral sites, as they are energetically favorable to achieve a minimum energy configuration. The idea of virtual atoms is illustrated in Fig. 1), where iron atoms (*grey*) are located at lattice sites, whereas virtual atoms (*blue*) and carbon atoms (*green*) are located at octahedral sites. The sampling of configuration space is performed by interchanging an interstitial atom with a virtual atom in the system via stochastic selection rules. Therefore, one has to consider the coupled events in two sub-volumes, Ω_1 and Ω_2 . In general, this exchange results in a non-equilibrium configuration which, for 0 K simulations, is relaxed in a subsequent step via molecular statics. After having relaxed the structure, the new configuration is accepted, if the energy difference between the two states is negative, i.e.

$$\delta E = E\{\Omega_1, 1 \rightarrow 2\} + E\{\Omega_2, 2 \rightarrow 1\} - E\{\Omega_1, 1\} - E\{\Omega_2, 2\} < 0 \quad (1)$$

where $E\{\Omega_1, 1 \rightarrow 2\}$ means the energy in sub-volume Ω_1 after relaxing to a new minimum energy state if the center atom in Ω_1 , e.g., a solute atom, has been replaced by the center atom of Ω_2 , e.g., a virtual atom (consequently, $E\{\Omega_2, 2 \rightarrow 1\}$ is the opposite process in sub-volume Ω_2). Ω_1 and Ω_2 are chosen randomly for a trial exchange of an atom. In principle, the sub-volumes are also allowed to overlap and consequently, the energy difference, Eq. 1, has to be computed for a connected region. This trial exchange can be considered as a 1-step limiting case of a switching process, which is performed in discrete steps [11–13]. According to a $T = 0\text{ K}$ energy minimization, the trial state is accepted if the energy in the new combined state is lower than in the old state. Accordingly, the configuration is explored towards the minimum energetic state of the system, although it is quite probable that the absolute minimum is not reached and the system is confined to a local minimum. This behavior is quite tolerable as a physical system will most probably also not reach the absolute minimum, due to the diffusive process of atoms which will also be captured in local minima. Therefore the method can reproduce real trends of system behavior and compute statistically significant density distributions of diffusive atoms.

Since local trial moves of atoms will most probably lead to high rejection rates due to large energy barriers between neighbored energy basins, the idea is to propagate the system via non-local moves, which explores configuration space more efficiently. However, since completely random trial moves of atoms will most likely result in unfavorable locations and therefore are most likely to be rejected, the present approach relies on so-called virtual atoms. These are placeholders, which are located at interstitial positions in the lattice and which have an asymmetric interaction, i.e., they are influenced by the interaction with the material atoms, e.g., Fe atoms, but in turn, they do not influence the material atoms, i.e., there is no force from the virtual atoms to the material. Therefore, they follow any lattice configuration changes, e.g., deformation, consistently, but do not affect the configuration by their presence. In this way, when interstitial

atoms are exchanged with a virtual atom, they are placed in a position, which is already near to local equilibrium, and the number of relaxation steps after an atom swap gets minimal.

3 Parallelization

To deduce some direct relationship between the atomic scale mechanisms such as carbon-segregation, carbon-dislocation pinning, retarded dislocation mobility which influence macroscopic material behavior (e.g., improved yield strength) large scale (multi-million atoms) atomistic simulations including defects and interstitial solutes are required. Large-scale simulations are necessary for this type of systems to reduce artifacts, originating from finite system size and boundary conditions to a minimum. In fact, every swap trial move of C-V atom pairs are followed by a relaxation procedure of all atoms in the sub-volumes Ω_1, Ω_2 for about 10^3 MS steps. For the present case, the sub-volumes have a radius of 2 nm, corresponding to about 2×10^4 atoms in each $\Omega_{1,2}$. For the Monte Carlo scheme, $\mathcal{O}(10^6)$ trial moves are to be performed to reach a steady state configuration and therefore, requirements in CPU time for this type of simulation are rather high. Consequently, parallelization is one efficient way to reduce the time-to-solution considerably. The main approach for the parallelization mentioned in this article refers to the coupling between a local MD and MC (*as highlighted as a blue region*) in Fig. 2 which takes a configuration from, e.g., a global MD simulation, as a starting configuration, performs an MC/MD coupled minimization of the solute atoms and finishes with a configuration, which might be used for post-processing or as a new starting configuration for a global MD.

For the parallelization, a manager-worker approach is chosen considering the performance and the complexity of physics to be simulated. A graphical illustration is shown in Fig. 3, explaining the overall work-flow of the implementation. For simplicity, one manager and four worker processors are considered here for schematic illustration. A work/job-queue (J1–J4) as shown in Fig. 3, is created by the manager and each item in the queue is related to a processor, associated with a pair of atoms, i.e., a virtual atom (V) and an interstitial carbon atom (I) that are chosen randomly for the corresponding MC step. Each job-ID in the job-queue maps to the MC step number and the worker processors flag a request-message to the manager. The manager owns a copy of all the coordinates of the system and sends appropriate information to the processors (Fig. 3 a: job allocation), i.e., transfer of information about which atom pair (I, V) to exchange as well as transfer of all coordinates $\{\mathbf{x}(\Omega_I)\}$ and $\{\mathbf{x}(\Omega_V)\}$ of atoms which belong to a spherical environment of radius R_c with a total volume $\Omega_{I,V}(p) = \Omega_I \cup \Omega_V$ on a processor p . Energy computations are performed for the constructed spherical sub-domains using local-MD at the worker, i.e., each worker transfers the coordinates of atoms to an MD routine, which is executed locally on the same core, which is associated to the worker (no further communication is invoked). Worker processors post a finish-message asynchronously to the manager processor immediately upon completion of the task. Completed job information is

updated in the job-queue (Fig. 3b: job retrieval) and where it is stored, until it can be decided if there were any spatial conflicts with other sub-domains, which were overlapping during the computation, resulting in a possible rejection of configurations. A new job is assigned to the worker processor instantaneously after retrieving the earlier finished job. After a certain time, the processors will run completely asynchronously due to runtime differences on the processors and differences in work allocation so that the manager is continuously retrieving information from and sending information to the workers.

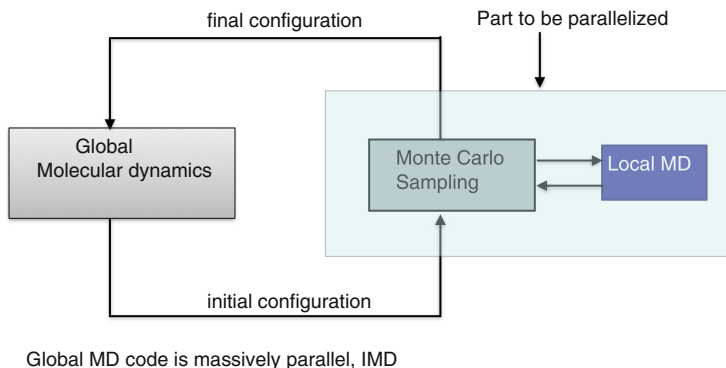


Fig. 2. Schematic showing the interface between global MD and MC-local MD coupling schemes. Modules to be parallelized are highlighted in a blue window. (Color figure online)

The atom swap together with the molecular statics calculation is performed on a given processor ‘p’ and the evaluation of the energy difference, Eq. 1, is sent back to the manager. In case that $\delta E < 0$, also the coordinates are sent back to the manager. Since many processors are working asynchronously in parallel and independently from each other, false overlap between interaction ranges can occur, i.e., $\Omega_{I,V}(p_i) \cap \Omega_{I,V}(p_j) \neq \emptyset$, which risks that the outcome of the processor which is at a later position in the work queue would depend on the outcome of the other processor earlier in the queue. Such possible scenario is illustrated in the schematics (Fig. 3), where MC step number ‘1’ and ‘3’ showing a spatial overlap, whose spherical domains are executed concurrently on the worker processor ‘W-3’ and ‘W-2’ respectively. Therefore, the manager has to analyze any possible overlap between regions, administrated on different processors and has to decide whether a trial configuration, which has been accepted on the worker level, has to be rejected on the manager level. The convention which we follow here is the following: if two processors which have been assigned a job ID ‘i’ and ‘j’, such that ($i < j$) show an overlap between their sub-volumes, and if on both processors the trial configurations would have been accepted (i.e., $\delta E_i < 0$ and $\delta E_j < 0$), then the configuration corresponding to the larger job-ID (in this case j) would be discarded. If we denote with $a(i)$ an acceptance and with $r(i)$

rejection of the trial configuration on processor i for the case of two processors, there are four different possible scenarios: (i) $\{r(i), r(j)\}$; (ii) $\{a(i), r(j)\}$; (iii) $\{r(i), a(j)\}$; (iv) $\{a(i), a(j)\}$. It is only in case (iv) that configuration j has to be discarded, although having reached a lower energy state locally on the worker processor.

It is evident that the higher I -concentration or, the larger the processor count, the more likely false overlap takes place [15]. This situation has been analyzed experimentally and analytically, and it was found that for homogeneous distributions of carbon interstitials with a concentration of $n_C = 0.01 \text{ wt}\%$ in a system of $N_{Fe} = 10^6$ iron atoms, the scalability gets up to ≈ 60 processors. The realization of larger processor counts will be possible when introducing hybrid schemes, where the relaxation for each Ω is done on sub-partitions of the computing system.

4 Application, Results and Discussions

A defect-free ferritic single crystal box is constructed with Fe atoms. Virtual atoms are introduced at all octahedral sites, and the box measures $600 \text{ \AA} \times 300 \text{ \AA} \times 69.25 \text{ \AA}$ along X, Y and Z direction respectively. A single screw dislocation is introduced using an analytical equation, such that the dislocation core locates at $X = 150 \text{ \AA}$ and $Y = 150 \text{ \AA}$ and the dislocation line runs parallel to the Z direction. (see Fig. 4). An in-house code is used to randomly introduce carbon atoms in the system to achieve the required background concentration. This code chooses random virtual atoms and turns them into C, as long as the desired concentration is achieved. Such random selection of virtual atom is necessary to assure that there is no systematic bias included in the initial configurations. And the configuration is relaxed using Molecular Statics (MS) to achieve a minimum energy configuration. IMD [16], a highly parallel and scalable classical MD package is used in this work for all Molecular dynamics (MD) and MS simulations.

The simulation box is completely periodic along the Z direction, as the dislocation line is parallel along Z direction and fixed along X and Y direction. Therefore, boundary walls of thickness 10 \AA are constructed along X, Y direction and the mobility of atoms belonging to these boundary zones are completely restricted. And the crystal orientation reads $X = [-1 \ 2 \ -1]$, $Y = [-1 \ 0 \ 1]$ and $Z = [1 \ 1 \ 1]$. To demonstrate the applicability of the method, a hybrid MD-MC scheme has been applied to the above described Fe-C system, consisting of $N_{Fe} = 1.07 \times 10^6$, $N_V = 2.88 \times 10^6$ virtual atoms and different number of carbon atoms for varied background concentrations namely $0.0025 \text{ wt}\%$, $0.005 \text{ wt}\%$, $0.01 \text{ wt}\%$, $0.015 \text{ wt}\%$, and $0.02 \text{ wt}\%$. Starting from a homogeneous distribution of carbon atoms (for example, see Fig. 4), a parallel hybrid MD-MC scheme is applied to model C segregation process in the presence of crystal defect like a dislocation. We used an embedded atom method (EAM) [17] to define the atomic interactions and a recently improvised FeC potential from Becquart group [18] by Veiga et al. [19] used in this work.

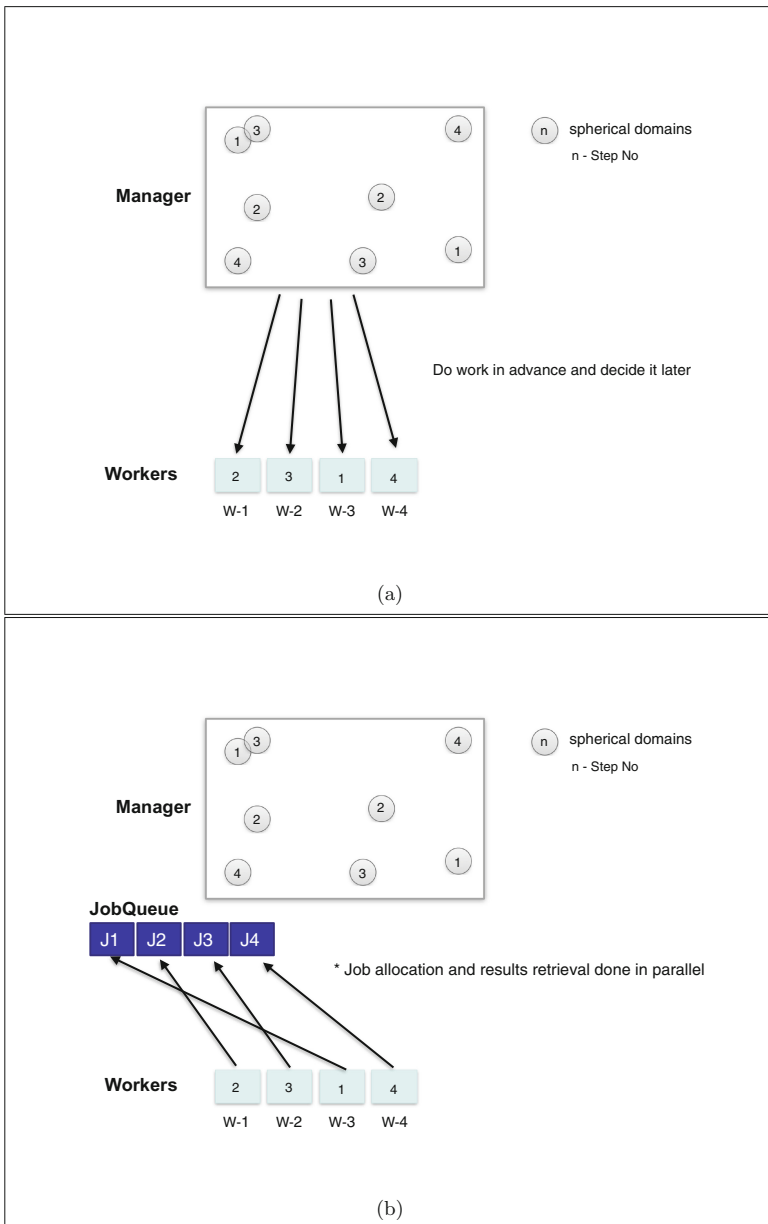


Fig. 3. Schematic illustrating how manager processor sample the whole computational domain through sphere constructions of randomly chosen carbon and virtual atom for a swap type trial move. During each MC step two spheres are constructed and shown with the corresponding step number. Corresponding jobs are assigned to free workers who post a message to the Manager during a `MPI_Probe` [14] call. Workers list containing different job ID's is shown as an example. (3a) Job allocations from manager to worker processors. (3b) Job retrievals from worker processors to manager.

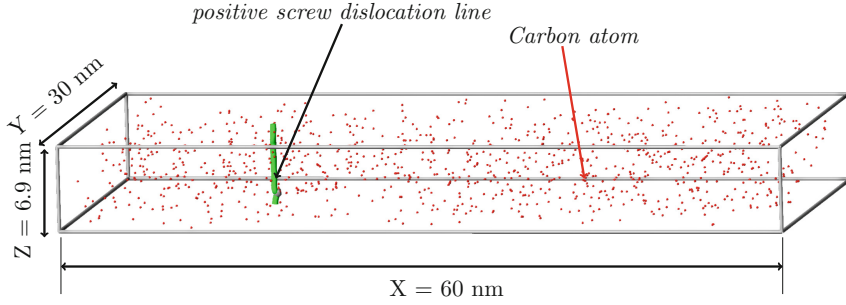


Fig. 4. An example simulation box set up used for modeling C segregation using the proposed hybrid MD/MC coupling approach linked via virtual atoms. The orthogonal simulation box spans $60 \text{ nm} \times 30 \text{ nm} \times 6.9 \text{ nm}$ along X, Y and Z. A positive screw dislocation is introduced at the coordinates ($X = 15 \text{ nm}$, $Y = 15 \text{ nm}$). Carbon atoms are randomly introduced to achieve some required background concentration and the distribution is qualitatively homogeneous.

To demonstrate the capability of this scheme, 5×10^5 MC steps are performed on different start configurations with varying background concentration of C atoms at $T = 0.5 \text{ K}$. At each MC step, efficient energy computation is performed in a MD external library, used for fast computation of energies of given configurations. The MD library was initially made for finite temperature simulations but was applied in the present work for micro-mechanics calculations at the ground state. To exploit the advantage of fast energy computations in the external library, the simulations are performed at $T = 0.5 \text{ K}$ with negligible thermal contributions and therefore needing no further adjustments of the algorithms. A systematic trend in the increase of C concentration is observed near the dislocation core for various configurations with a different global concentration of carbon. Atomic weight concentrations of carbon atoms are computed for different output configurations after 5×10^5 MC steps using an in-house visualization tool ‘AtomViewer’. C atoms within the radius of 15 \AA are considered for calculating the concentrations. Such concentration profiles of C atoms shown are shown in Fig. 5 (Left), along with the carbon Cottrell atmospheres Fig. 5 (Right), for simulation boxes with different background concentration of C atoms in the order 0.02 wt.%, 0.015 wt.%, 0.01 wt.% and 0.005 wt.%. (Top to Bottom in Fig. 5) respectively. It is observed that higher C concentration near the dislocation core with an increase in background C concentration. Configurations achieved after 0.5 Million MC steps are shown for all cases, and the segregated carbon atoms are forming three-fold symmetry pattern around the screw dislocation core are well seen in the cases of high concentration (0.02 wt.%, 0.015 wt.% and 0.01 wt.%). More lattice distortions and strain field are introduced in high concentration cases, due to carbon-carbon interactions segregated near the dislocation core. Therefore, more carbon atoms find these sites near the dislocation core to be energetically favorable during the trial move. Hence, non-local trial moves (i.e., displacing a C atom from far region towards dislocation core) proves to be

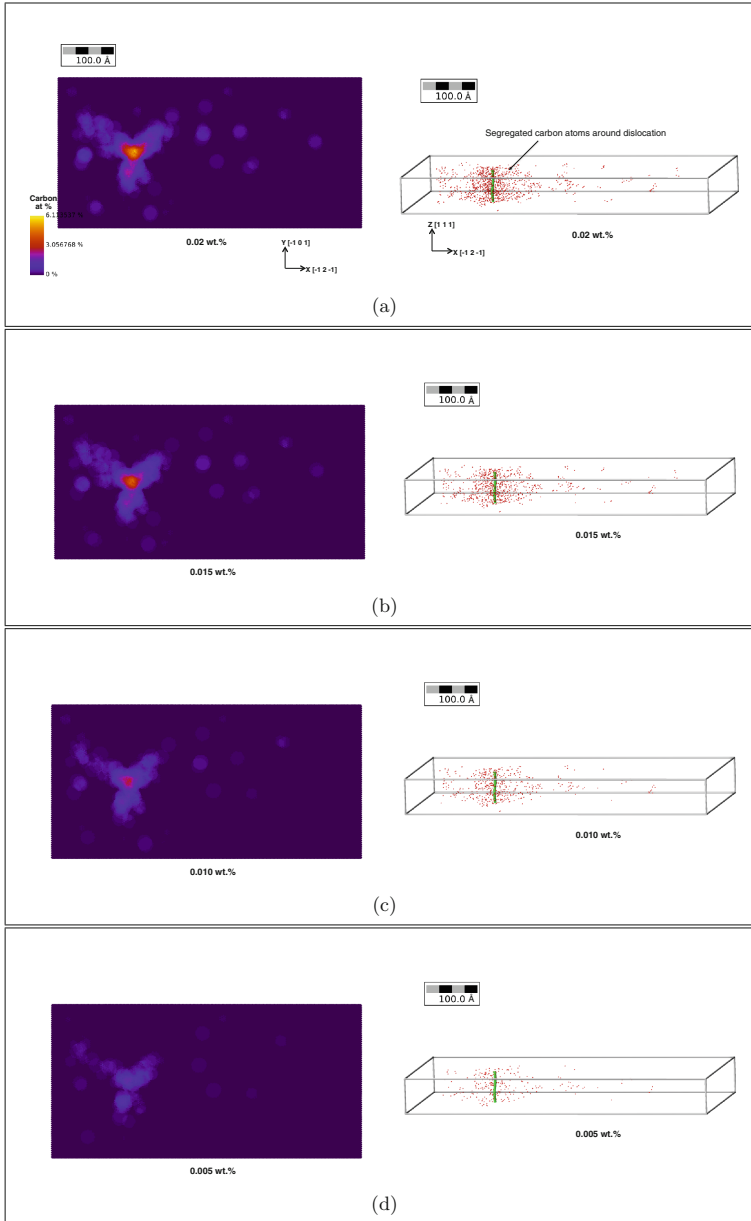


Fig. 5. Carbon segregated configurations achieved after 0.5 Million MC steps using the hybrid approach for simulation domain with different background concentrations (Top-Bottom: 0.02 wt.%, 0.015 wt.%, 0.01 wt.%, 0.005 wt.%). In 5(a-d), atomic weight concentrations (Left) and carbon Cottrell atmospheres (Right) formed near the dislocation core are shown. Atomic concentrations are computed within the sphere radius of 1.5 nm for these computations and with increase in carbon concentration, Cottrell cloud exhibit a three-fold symmetry pattern around the screw dislocation with higher concentration at the core.

more successful till the segregated C atoms near the core achieve some saturation or more stable clusters. However, for configurations with low global carbon concentrations, the segregated carbon atoms are not showing a clear three-fold pattern. Because, in these cases, C atom once finding a stable site near the core or any other distortion site (partially away) shall not prefer some large non-local trial move anymore. Also, the lattice distortions caused by carbon-carbon interactions due to local C concentration is less. Carbon Cottrell atmospheres showing such three-fold pattern are in qualitative agreement with some earlier published works [9, 10, 20].

The earlier obtained configurations with partially segregated C atoms using a parallel hybrid MD/MC scheme is used for performing some shear simulations. Boundary walls of the simulation domains are reconstructed such that, the atomic layers spanning $Y = 0-1$ nm and $Y = 29-30$ nm are fixed. Virtual atoms belonging to the fixed zones are eliminated to exploit some computational performance. Now the configurations are completely relaxed using Molecular Statics (MS) to minimize the system energy due to segregated carbon atoms. Shear forces applied to the fixed walls corresponding to the plane containing the normal direction along $[1\ 0\ 1]$.

Shear simulations are performed using IMD [16], using an NVT ensemble computed at $T = 0.5$ K to be thermally consistent with the segregation simulations using the parallel MD-MC coupling. Force controlled shear simulations are performed, where a pre-computed measure of force is applied to the fixed layer of atoms in the outer x, z -surface layers. Externally applied shear forces on the fixed walls as rigid motion act as the necessary driving force to dislocation motion.

The force computation is based on the critical stress ($\tau = 1.2$ GPa) of a screw dislocation in bcc Fe, from the earlier published works of our group member [21, 22]. Computation is based on the relation $\tau = F/A$, where $F = f \times n$, f is the force per atom, n the total number of atoms in the fixed wall and A the cross-sectional area ($X \times Z$). We applied a maximum stress of 1.08 GPa during the first 10 ps of the simulation, followed by a second stage with a maximum stress of 1.44 GPa for 40 ps. The shear simulations were performed with a ‘NVT’ ensemble and output files were written for every 1 ps. According to the target shear stress, the external force is computed to be around $f = 0.008629$ nN/atom and applied along $+Z$ and $-Z$ direction on the opposite walls (see Fig. 4). Also, the ‘RIGID’ feature of IMD [16] is used for these boundary atoms, meaning that they will displace only along Z direction as rigid bodies. Shear simulations are partitioned into two sub-simulation, where the first sub-simulation runs for $T_1 = 10$ ps during which 0–90% of the estimated force is applied in an incremental fashion. The second sub-simulation part runs for $T_2 = 40$ ps during which 90–120% of the estimated force is applied in an incremental fashion. Simulation snapshots for every $T = 10$ ps are obtained to observe the changes in dislocation mobility in the configurations with segregated carbon atoms and homogeneous distribution of C atoms with the different background global concentration. Some example snapshots (zoomed-in view) for different configurations (0.02 wt.% and

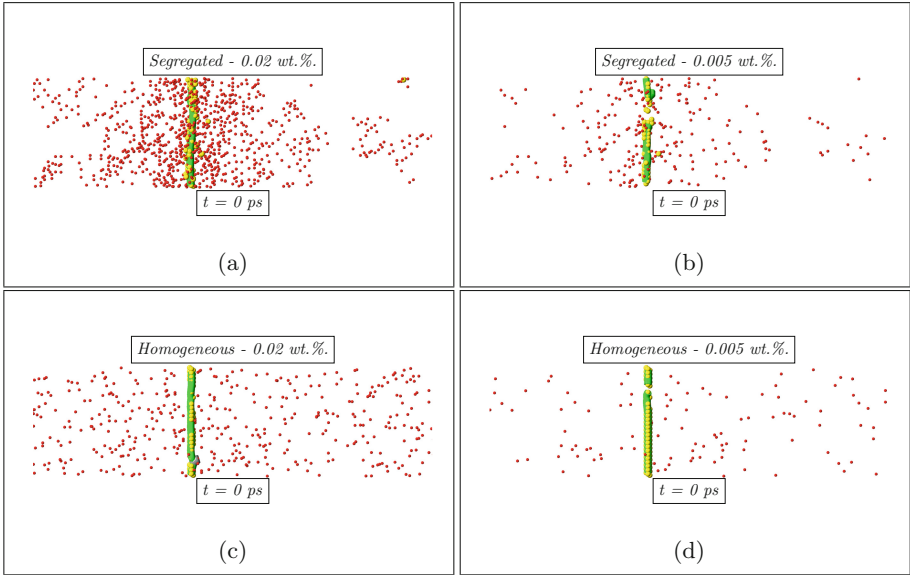


Fig. 6. Initial configurations with segregated (6a, 6b) and homogeneous (6c, 6d) distribution of carbon atoms used as input for shear simulations for different global concentrations of 0.02 wt.% (Left) and 0.005 wt.% (Right) respectively.

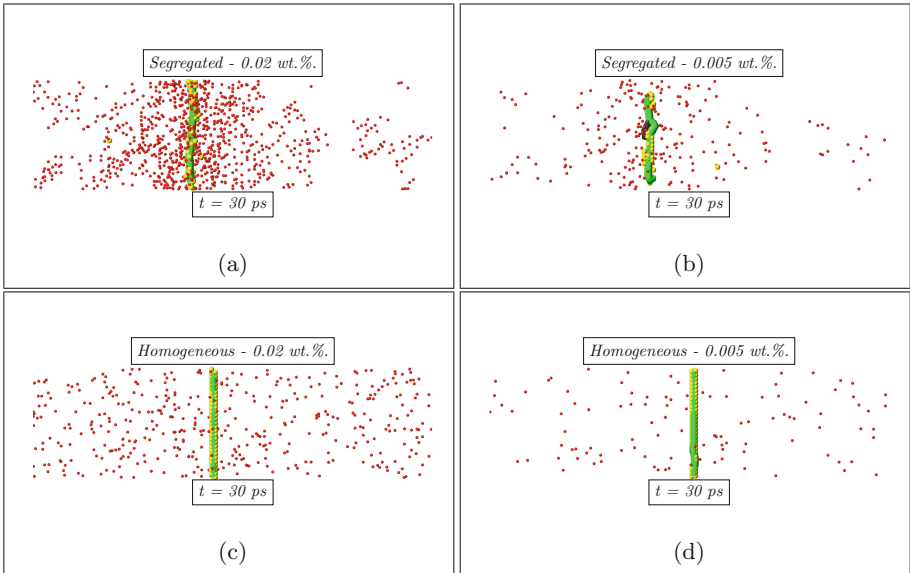


Fig. 7. Sheared configurations achieved after $t = 30$ ps for segregated (7a, 7b) and homogeneous (7c, 7d) distribution of carbon atoms with different global concentrations of 0.02 wt.% (Left) and 0.005 wt.% (Right) respectively.

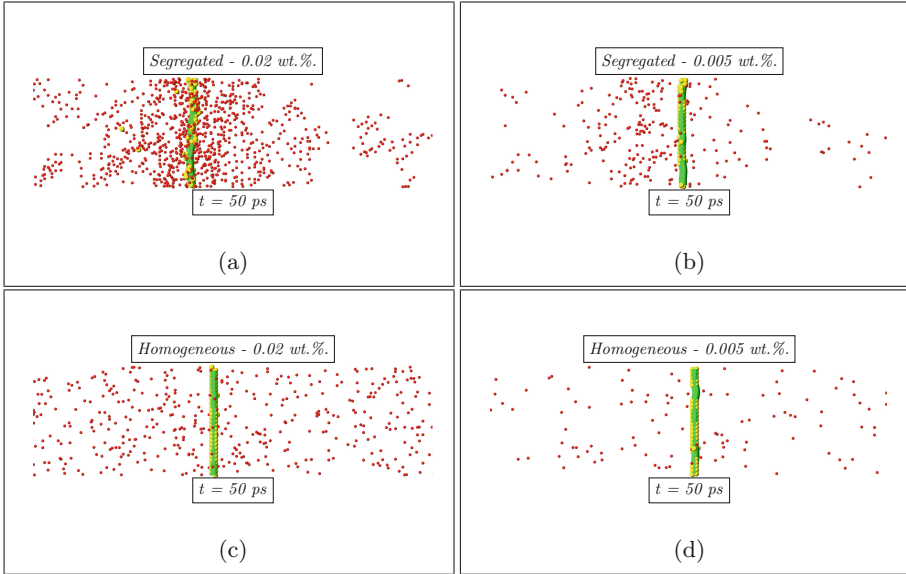


Fig. 8. Sheared configurations achieved after $t = 50$ ps for segregated (8a, 8b) and homogeneous (8c, 8d) distribution of carbon atoms with different global concentrations of 0.02 wt.% (Left) and 0.005 wt.% (Right) respectively.

0.005 wt.%) are shown (refer Fig. 8) at different time steps $t = 0$ ps (see 6a, b, c, d), $t = 30$ ps (see 7a, b, c, d), and $t = 50$ ps (see 8a, b, c, d) for segregated and homogeneously distributed C environment.

It is observed that the screw dislocation is almost completely locked inside the Cottrell atmospheres (*segregated C atoms*) with a minimum displacement of 1.9 \AA inside the configuration with 0.02 wt.% C (see Figs. 6a and 8a), for the applied shear force for a period, $t_{total} = 50$ ps. In comparison to a configuration with a homogeneous distribution of C with concentration 0.02 wt.%, the displacement measures to be 15.94 \AA . Also the dislocation mobility in configurations with low C concentrations (0.005 wt.%) are measured, compared for segregated and homogeneous distribution of C atoms. Displacements measured after $t_{total} = 50$ ps read 32.718 \AA (*segregated*) and 73.337 \AA (*homogeneous*) respectively (see Fig. 8b, d). Further measurements of dislocation displacement, for other configuration with concentrations (0.015 wt.%, 0.010 wt.%, 0.0025 wt.%, and 0.0 wt.%) showed a trend that the difference in displacement between the segregated and homogeneous distribution of C slowly decreased for the lowering global C concentrations. Hence, it is qualitatively concluded that the local concentration of C atoms forming the Cottrell atmospheres near the dislocation core proves to effectively contribute to the hindrance in dislocation mobility. Because it is evident from the configuration with higher global C concentration (0.02 wt.%), where the influence of local C concentration in the vicinity of dislocation core showed to be a factor of 10. To make some quantitative estimate, sufficient statistics

shall be performed for a finite number of start-up configurations with equal concentrations.

In the work of Veiga et al. [9,10], where an MD and kinetic MC are coupled to model C Cottrell atmospheres and study its influence on dislocation mobility. However, there exist several key differences in comparison to this work. The concept of ‘virtual atoms’ used in this work to link MD and MC is unique to capture any distortion in the ferritic host environment, to include chemical and mechanical contributions consistently and to efficiently sample the configurational space. Dimensions of the computational domain are extensively large, and the number of atoms used in this work is a factor of 10 more (i.e., in Million) with the aim to simulate a system with complex dislocation networks. A uniform and unbiased MC sampling approach is used in this work to probe the configurational space, whereas Veiga et al. [10] used a biased sampling approach that the probability of choosing an octahedral site within 5 nm from the dislocation line is about 50%. Most importantly, Veiga et al. [10] performed trial moves only by swapping random octahedral sites. However, virtual atoms used in this work though introduced at standard octahedral sites initially, have the potential to rearrange to some off-lattice sites considering the distortions in the environment. The parallel MD-MC coupling framework applying the concept of *virtual atoms*, introduced in this work has a generic character, such that it is possible to extend the present study to other systems with interstitial solutes, e.g., Mn or H in Fe or other systems. In principle, the requirement is a suitable interaction potential plus an extension of the model towards a one-way interaction model between the virtual atoms and the underlying atomic lattice.

5 Conclusion

It could be shown that the segregation process in the presence of the dislocation can be simulated very efficiently using the MD-MC coupling approach. Shear simulation results using MD gave qualitative evidence that the local concentration of interstitial solutes, like carbon, provides a significant pinning effect for the dislocation. The parallel scalability of the scheme is reduced due to false overlap when the concentration of C-atoms increases around the dislocation core during segregation. This efficiency degradation will be compensated by an additional parallel-replica approach in the near future. For a homogeneous defect free Fe-C-system with $N_{Fe} \approx 10^6$ particles, scalability could be obtained up to 32–64 processors. Work is in progress towards an efficiency control scheme which decides on-the-fly for a given number of processors upon an extension towards a replica system method. Further work on the hybrid MS/MC scheme using virtual atoms and a parallelization using a manager-worker approach is ongoing and will be published in future.

Acknowledgements. The authors gratefully acknowledge the funding from Deutsche Forschungsgemeinschaft (DFG) - BE 5360/1-1.

References

1. Cottrell, A.H., Bilby, B.A.: Dislocation theory of yielding and strain ageing of iron. *Proc. Phys. Soc. London Sect. A* **62**(1), 49 (1949)
2. Wilde, J., Cerezo, A., Smith, G.D.W.: Three-dimensional atomic-scale mapping of a Cottrell atmosphere around a dislocation in iron. *Scripta Mater.* **43**(1), 39–48 (2000)
3. Zhao, J., De, A., Cooman, B.D.: A model for the cottrell atmosphere formation during aging of ultra low carbon bake hardening steels. *ISIJ Int.* **40**(7), 725–730 (2000)
4. Zhao, J., De, A., De Cooman, B.: Kinetics of cottrell atmosphere formation during strain aging of ultra-low carbon steels. *Mater. Lett.* **44**(6), 374–378 (2000)
5. Waterschoot, T., De Cooman, B., De, A., Vandeputte, S.: Static strain aging phenomena in cold-rolled dual-phase steels. *Metall. Mater. Trans. A* **34**(3), 781–791 (2003)
6. Berbenni, S., Favier, V., Lemoine, X., Berveiller, M.: A micromechanical approach to model the bake hardening effect for low carbon steels. *Scripta Mater.* **51**(4), 303–308 (2004)
7. Khater, H., Monnet, G., Terentyev, D., Serra, A.: Dislocation glide in fe-carbon solid solution: from atomistic to continuum level description. *Int. J. Plast.* **62**, 34–49 (2014)
8. Sadigh, B., Erhart, P., Stukowski, A., Caro, A., Martinez, E., Zepeda-Ruiz, L.: Scalable parallel Monte Carlo algorithm for atomistic simulations of precipitation in alloys. *Phys. Rev. B* **85**(18), 184,203 (2012)
9. Veiga, R.G.A., Perez, M., Becquart, C.S., Domain, C.: Atomistic modeling of carbon Cottrell atmospheres in bcc iron. *J. Phys. Condens. Matter* **25**(2), 025,401 (2012)
10. Veiga, R.G.A., Goldenstein, H., Perez, M., Becquart, C.S.: Monte Carlo and molecular dynamics simulations of screw dislocation locking by Cottrell atmospheres in low carbon Fe-C alloys. *Scripta Mater.* **108**, 19–22 (2015)
11. Frenkel, D: Lecture notes on: free energy calculations. In: Meyer, M., Pontikis, V. (eds.) *Computer Simulation in Materials Science*, p. 85. Kluwer Academic Publishers, Amsterdam (1991)
12. Jarzynski, C.: Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.* **78**, 2690 (1997)
13. Crooks, G.: Nonequilibrium measurements of free energy differences for microscopically reversible Markovian systems. *J. Stat. Phys.* **90**, 1481 (1998)
14. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard Version 3.0 (2012). <http://mpi-forum.org/mpi-30/>
15. Sutmann, G., Ganesan, H., Begau, C.: Cluster formation in stochastic disk systems. In: *AIP Conference Proceedings*, vol. 1863, p. 560089 (2017). <https://doi.org/10.1063/1.4992772>
16. Stadler, J., Mikulla, R., Trebin, H.-R.: IMD: a software package for molecular dynamics studies on parallel computers. *Int. J. Mod. Phys. C* **8**(05), 1131–1140 (1997)
17. Daw, M.S., Baskes, M.I.: Embedded-atom method: derivation and application to impurities, surfaces, and other defects in metals. *Phys. Rev. B* **29**(12), 6443 (1984)
18. Becquart, C.S.: Atomistic modeling of an Fe system with a small concentration of C. *Comput. Mater. Sci.* **40**(1), 119–129 (2007)

19. Veiga, R.G.A., Becquart, C.S., Perez, M.: Comments on Atomistic modeling of an Fe system with a small concentration of C. *Comput. Mater. Sci.* **82**, 118–121 (2014)
20. Cochardt, A.W., Schoek, G., Wiedersich, H.: Interaction between dislocations and interstitial atoms in body-centered cubic metals. *Acta Metall.* **3**(6), 533–537 (1955)
21. Koester, A., Ma, A., Hartmaier, A.: Atomistically informed crystal plasticity model for body-centered cubic iron. *Acta Mater.* **60**(9), 3894–3901 (2012)
22. Chockalingam, K., Janisch, R., Hartmaier, A.: Coupled atomistic-continuum study of the effects of C atoms at α -Fe dislocation cores. *Modell. Simul. Mater. Sci. Eng.* **22**(7), 075007 (2014)



3D Microstructure Modeling and Simulation of Materials in Lithium-ion Battery Cells

Julian Feinauer^(✉), Daniel Westhoff, Klaus Kuchler, and Volker Schmidt

Institute of Stochastics, Ulm University, Helmholtzstraße 18, 89069 Ulm, Germany
julian.feinauer@uni-ulm.de

Abstract. The microstructure of lithium-ion battery electrodes has a major influence on the performance and durability of lithium-ion batteries. In this paper, an overview of a general framework for the simulation of battery electrode microstructures is presented. A multistep approach is used for the generation of such particle-based materials. First, a ‘host lattice’ for the coarse structure of the material and the placement of particles is generated. Then, several application-specific rules, which, e.g., influence connectivity are implemented. Finally, the particles are simulated using Gaussian random fields on the sphere. To show the broad applicability of this approach, three different applications of the general framework are discussed, which allow to model the microstructure of anodes of energy and power cells as well as of cathodes of energy cells. Finally, the validation of such models as well as applications together with electrochemical transport simulation are presented.

Keywords: Stochastic 3D microstructure modeling
Lithium-ion cell anodes · Lithium-ion cell cathodes
Gaussian random fields on the sphere

1 Introduction

The usage of lithium-ion battery cells is steadily growing in various fields of daily life. This implies the necessity for further improvement of battery materials. As laboratory experiments are expensive in time and costs, model-based simulations of electrochemical processes in battery cells have become an important part of battery research. Simulations are also necessary for the optimization of charging strategies and control systems [24].

Electrochemical simulation models go back to the famous work of Newman and co-workers [21]. However, these models neglect detailed geometry information of the cell and its 3D microstructure by using averaged characteristics. Recently, spatially resolved transport models have been developed to simulate the charge transport in lithium-ion battery cells [16, 17].

In this work, the focus is on stochastic 3D microstructure models. Such models can be used as spatial input information for spatially resolved transport models, where highly resolved image data of 3D microstructures of battery electrodes are necessary for model calibration. The major advantage of the combination of stochastic microstructure models with spatially resolved transport models is that this approach can be used for realistic simulations of local effects. This includes lithium-plating and further aging mechanisms which are influenced by local potentials. Furthermore, the computational complexity of this approach can be minimized for specific applications like cycle-life simulations by reduced basis methods [8]. An example of a structure generated by such a 3D microstructure model is shown in Fig. 1.

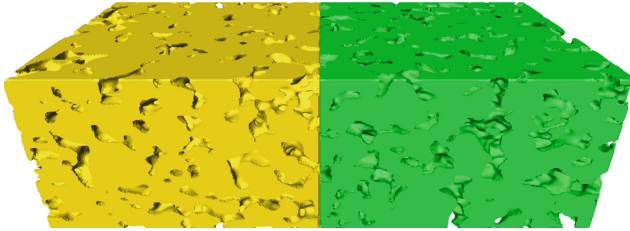


Fig. 1. Comparison of experimental and simulated anode structures. Experimental data (left), simulated data (right).

2 Simulation of Lithium-ion Cell Electrodes

2.1 General Approach

Methods of stochastic geometry and spatial statistics have been proven to be a viable tool for the simulation of 3D microstructures of energy materials like those used in fuel cells or solar cells [9, 10, 25]. The general idea of these methods is to provide microstructure models based on a few parameters that are able to generate 3D microstructures which are similar in a statistical sense to those observed in experimental data. This means that averaged characteristics like volume fractions or specific surface areas but also more refined descriptors of microstructures like tortuosity of transportation paths or the pore size distribution are in good agreement, see [20, 26] for more details.

Furthermore, these models and the corresponding simulation algorithms are off-grid meaning that they can be used to generate structures on arbitrary length scales. The computation time it takes to generate a realization of such a microstructure model is usually very small. Thus, the usual problems of tomographic imaging related with the generation of a sufficiently large amount of highly resolved 3D images in sufficiently large regions of interest can be solved once such a model is developed and fitted to experimental data.

2.2 Modeling Framework

The microstructures of different types of lithium-ion battery electrodes vary from each other, however, they share some common structural properties. In general, electrodes consist of systems of connected particles that can have complex shapes. For the electrode models it is very important to fit fundamental microstructure characteristics like porosity and specific surface area exactly [23]. Another important characteristic of the electrodes is the high connectivity of particles. This means that each particle is connected to many neighboring particles. Thus, the model of individual particle sizes and shapes has to be flexible enough to meet all these conditions.

A general framework based on a stochastic particle model which possesses these properties has been developed. It is based on the representation of particles as linear combinations of spherical harmonics which can then be seen as realizations of Gaussian random fields (GRFs) on the sphere. Particles with random sizes and shapes can be simulated using GRFs on the sphere whose parameters are fitted to the coefficients of the spherical harmonics expansions of particles extracted from experimental data [7]. The whole framework consists of several steps, which are described in more detail in the following subsections:

- approximation of particles
- particle model
- host lattice, particle arrangement
- connectivity of particles
- particle placement and boundary conditions
- simulation of individual particles

Approximation of Particles. As our aim is to develop a stochastic model for irregularly shaped particles of electrode materials, a first step is to find a suitable (analytical) representation of the particles observed in tomographic images. Thus, each particle is approximated by an analytical function using an expansion in spherical harmonic functions.

The spherical harmonic functions form a basis of the family of square-integrable functions defined on the unit sphere. This means that every square-integrable function can be written as an expansion in spherical harmonics. This is pretty similar to the well-known Fourier series expansion that is frequently used in signal processing.

In such an expansion the spherical harmonics have a natural ordering by their degree l and their order m . Regarding our application this means that spherical harmonic functions with higher order l represent smaller features of the particle surfaces. This motivates the truncation of the series of spherical harmonic functions at a certain cutoff degree L which leads to a good approximation of the particles.

On the one hand, this is necessary for numerical calculations and it makes further modeling easier. But, on the other hand, this is a natural way to smooth the particles, e.g., to minimize artifacts resulting from measurements, binarization or (particle) segmentation. This smoothing is similar to the theory used in

signal processing with Fourier methods. Of course, a good choice of the parameter L is crucial and thus has been investigated in detail in [7].

Images showing the experimental microstructure before and after the smoothing of particles by the usage of spherical harmonics are given in Fig. 2. A more detailed comparison for an individual particle can be found in Fig. 3.

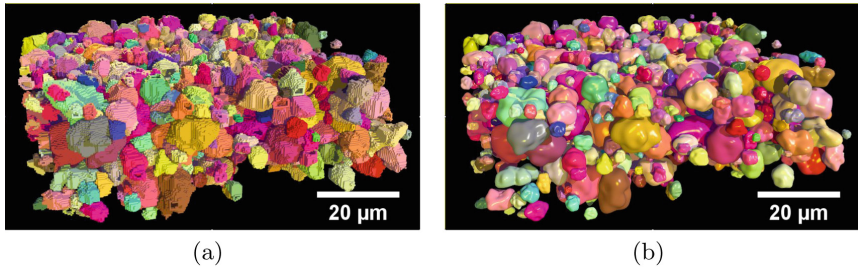


Fig. 2. Comparison of the result of structural segmentation and approximation of particles by spherical harmonics. Particle system before (a), and after approximation by spherical harmonics (b). Reprinted from [7] with permission from Elsevier.

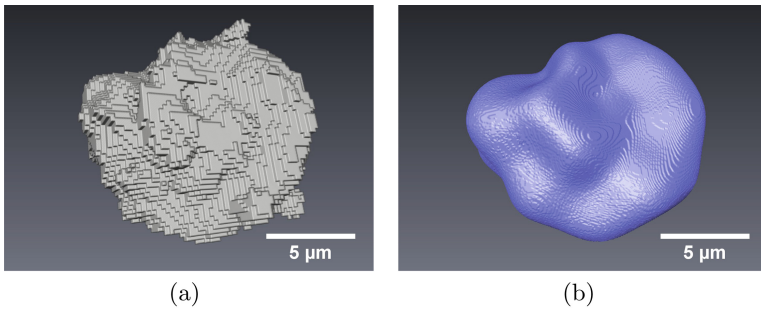


Fig. 3. Example of a particle from an energy cell anode. Segmented particle from tomographic image (a) and approximation of the particle with a truncated series of spherical harmonics for $L = 10$ (b). Reprinted from [7] with permission from Elsevier.

Particle Model. As described above, particles in electrode materials are often complex shaped and no perfect spheres, see Fig. 3(a). Thus, simple approaches where particle models are based on spheres, like, e.g., in [11] are not sufficiently precise, especially since the volume fraction as well as the surface area of particles plays an important role for the functionality of the material, see, e.g., [2, 4]. Thus, an alternative method which allows great flexibility, namely Gaussian random fields on the sphere, lead to a better accuracy of the model.

Gaussian random fields on the sphere are a model to describe the surface of randomly shaped objects. The idea is to generate a random radius function

(in spherical coordinates) that assigns a radius value to each point (or angle) on the sphere. Of course, these values are not totally independent but have a spatial correlation. The exact spatial correlation can be controlled by the so-called angular power spectrum (APS) which is related to the Fourier spectrum in the 1D analogy. Thus, for a given material the APS has to be calculated for particles from experimental data and can then be used to generate random particles. An example of the APS for random particles from an energy cell is shown in Fig. 4.

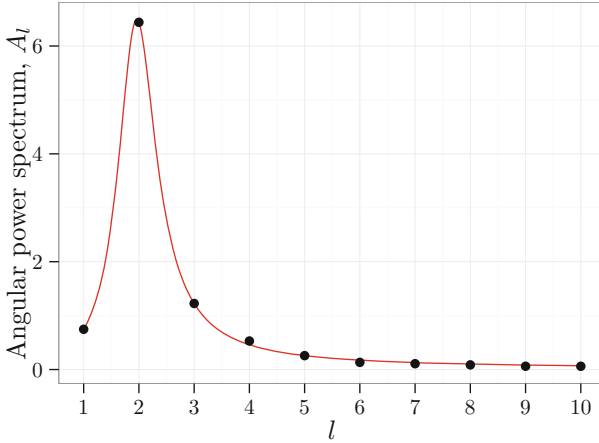


Fig. 4. Example of an angular power spectrum. Reprinted from [6] with permission from Elsevier.

It turns out that this approach works well and that the simulated particles are in good accordance with the particles from the experimental samples, even for very differently shaped particles. Formally, this can be seen by comparing shape characteristics like volume, surface area or sphericity for two large sets of simulated particles and particles from experimental data, see [6, 7] for more details.

Host Lattice, Particle Arrangement. As electrode materials can have high volume fractions of active materials (up to 73%, see e.g., [7]) one major challenge is to create a dense packing of irregularly shaped particles. There exist several approaches to generate dense particle structures directly from a set of particles, e.g., by using packing algorithms like the force-biased algorithm [19]. However, a huge drawback is that this approach is very time-consuming and for large observation windows it takes up to days on modern hardware. Furthermore, dense packing algorithms are usually applied to packings of spheres. In a situation where particles are not spherical or even not convex they often cannot be applied or at least are way more complex.

Moreover, the particles (that will be generated later with the model described above) have high connectivity. Thus, an indirect approach for the placement of particles is used. First, a ‘host lattice’ is generated consisting of single polytopes where particles will be placed inside in the next step.

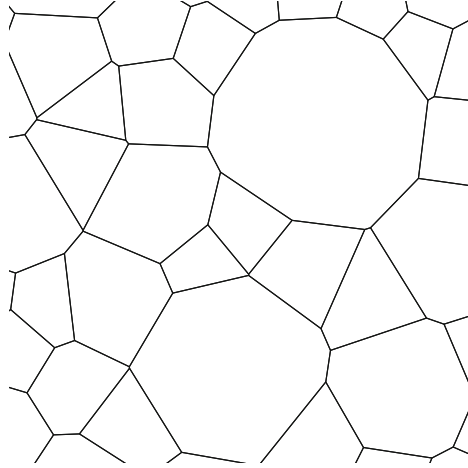


Fig. 5. 2D intersection of a Laguerre tessellation. Reprinted from [6] with permission from Elsevier.

A Laguerre tessellation is used as ‘host lattice’, which is a division of the space into disjoint convex polytopes. In more detail, a random tessellation [18] is used as this gives great flexibility but can also be generated efficiently as Laguerre diagram of a random marked point pattern, see Fig. 5. The generation of a ‘host lattice’ for particle placement has to be adopted for different materials. This means that the random point pattern for the generation of the Laguerre tessellation has to be fitted to each case, and in some applications not all polytopes are used as particle hosts but some are left empty.

Connectivity of the Particles. As emphasized before, the high connectivity of particles is very important for electrode materials. Thus, this property has to be incorporated in the model. This is done by using a random graph which contains the corresponding information, the so-called connectivity graph. The nodes of this graph correspond to the polytopes of the tessellation and an edge in the graph indicates that the particles, that will be placed in two polytopes later on, have to be in contact with each other.

Two polytopes are said to be neighboring if they share a common facet. Thus, facets of the tessellation can be seen as edges of the connectivity graph. An example of such a graph is shown in Fig. 6.

One important side condition is, as stated above, that there are no particles or clusters of particles that have no connection to the rest of the system. This means that for each particle there has to be a path of connections to every other particle

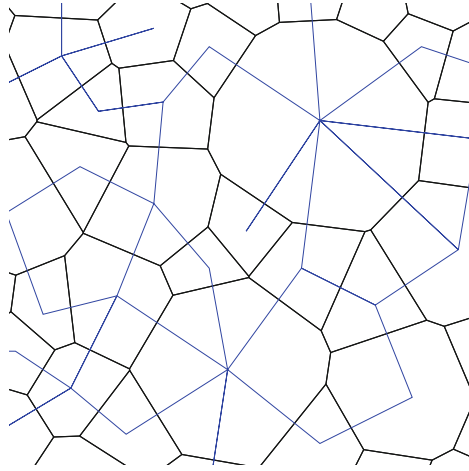


Fig. 6. Example of a connectivity graph (blue) and the underlying tessellation (black). Reprinted from [6] with permission from Elsevier. (Color figure online)

in the bulk. This can be achieved by a special construction algorithm of the graph as follows. After the generation of the tessellation, first the graph of full connectivity is generated. This means the polytopes in the observation window are taken as vertices and for the edges, every facet is turned into an edge that connects the two polytopes that share this facet. Thus, in this graph each node is connected to all its neighbors. Depending on the application a weighting of the edges can be applied.

Then, the minimum spanning tree (MST) of the graph of full connectivity is calculated. The MST is the minimal subgraph which fulfills the condition that there exists a path from every node to every other node in the graph [13].

The MST forms the basis for the connectivity graph. This final connectivity graph is then generated by adding edges between neighboring nodes based on a Bernoulli experiment, where the edge-putting probability is given by variations of the following different model types:

- The probability is fixed and independent of the polytope/particle.
- The probability is given as a function of characteristics (like the surface area) of the common facet.
- The probability depends on the angle of the connection of the centroids of the two neighboring polytopes.
- The probability depends on the distance between the centroids of the two polytopes.

The latter option is especially necessary if the electrode microstructure is anisotropic. More details can be found in [28].

Particle Placement and Boundary Conditions. Given the model for the particle shapes based on the GRFs on the sphere, particles are placed inside polytopes of the tessellation.

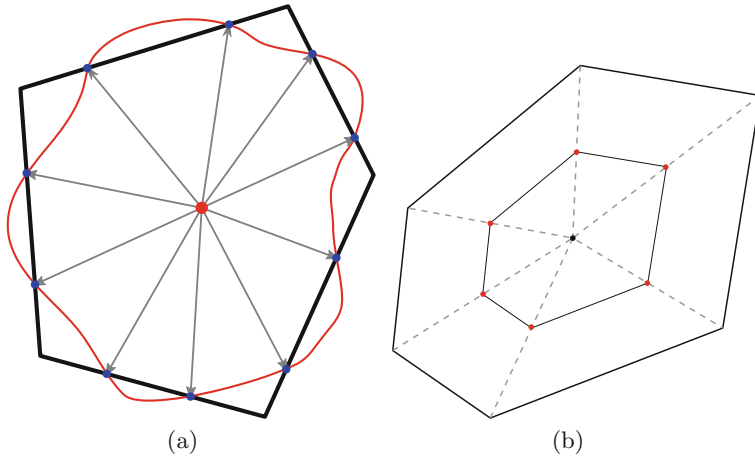


Fig. 7. (a) 2D schema of the boundary conditions. Particle (red) and surrounding polytope (black). The boundary conditions are that the particle touches the facet at specific points (blue). These conditions are also generated for the neighboring particles, thus it is ensured that both particles touch each other. (b) Boundary conditions on the facet (red dots). Both neighboring particles have to touch several points on their common facet and thus are connected. Reprinted from [6] with permission from Elsevier. (Color figure online)

Furthermore, the following two rules are used for the particle placement:

- A particle’s shape should roughly follow the shape of the polytope.
- A particle has to touch its neighbors according to the edges of the connectivity graph.

Generally, the particles should have shapes that are characteristic for the material. This is ensured by using realizations of the particle model explained above. As the particles are placed inside the Laguerre polytopes, their barycenters are used as origins of spherical coordinate systems. The particles are then represented as an expansion in spherical harmonic functions whose coefficients are generated from the GRF model described above.

Simulation of Individual Particles. The last step for the generation of a simulated microstructure is the generation of particles. On the one hand, they need to be generated from the (stochastic) model described above and on the other hand they need to fulfill the boundary conditions explained previously.

This is indeed possible due to the special nature of the particle model. To be more precise, because the particle model is based on an expansion in spherical harmonics, the rules stated above translate into linear boundary conditions. As the coefficients are then sampled from a GRF model, the problem can be translated to drawing coefficients from a multivariate Gaussian distribution with boundary conditions. To make the simulation more efficient the distribution

is transformed to a distribution whose realizations always fulfill the boundary conditions. Further details are given in [6].

3 Simulation of Particulate Materials

Using the method for modeling and simulation of individual particles described above, it is possible to build models for different kinds of electrode materials. Electrodes in so-called energy cells like the ones used in electric vehicles have high particle densities and relatively small porosities.

The simulated 3D microstructures are then used for spatially resolved transport simulations [14] where the focus is put on the electrochemical validation of the simulated microstructures. Moreover, in [8], the electrochemical simulations performed on simulated 3D microstructures are combined with model order reduction methods to accelerate the whole procedure which enables the simulation of multiple cycles, e.g., for virtual aging tests.

In lithium-ion cells designed for power applications like the ones used in plugin hybrid vehicles, the volume fraction of active material is lower but the specific surface area is higher to allow for higher charge and discharge currents. For a detailed comparison for different electrode and cell types see [5, Table 2]. Such 3D microstructures have been simulated with an extended version of the modeling approach described above for energy cells, where a refined tessellation model is used and some polytopes are left empty when placing the particles [28].

The counterpart of the anode in a lithium-ion battery cell is the positive electrode, also called cathode. The considered cathodes of plugin hybrid energy cells exhibit low volume fractions of active material, a different particle connectivity and especially nearly spherical particle shapes. By adaptations and enhancements of the above two modeling approaches for anodes, it is possible to use the general framework also for simulation of 3D cathode microstructures.

In the following the different applications of the general framework described above and the adaptations are discussed in more detail.

3.1 Anode of an Energy Cell

The main difficulty for the simulation of anodes that are optimized for a large energy density is the high volume fraction of active material. In our case the volume fraction of active material is about 73%. Therefore, one of the main challenges is to generate realistic structures with such a high volume fraction but still realistic particles and the high connectivity present in the material. Common approaches for the generation of dense packings like, e.g., force-biased algorithms usually require simple particle shapes like balls [1].

Host Lattice, Particle Arrangement. The generation of the host lattice is based on a point pattern generated by a so-called random sequential adsorption process (RSA) [3, 27]. This leads to point patterns that have some ‘regularity’. This means that, with a high probability, there will be no isolated points as well as no clustering of points. The ‘host lattice’ is then obtained by calculating the Laguerre diagram of the realization of the RSA process.

Connectivity of the Particles. Based on the minimum spanning tree of the graph of full connectivity, edges are added with a probability proportional to the area of the facet between two neighboring polytopes. Figure 7 shows the boundary conditions that are applied to the particles in detail.

Simulation of Individual Particles. The particles are simulated using the GRF model described above with an angular power spectrum that has been fitted to the particles extracted from tomographic images. The angular power spectrum is shown in Fig. 4. A comparison of the experimental structure and the result of model-based simulation is shown in Fig. 1.

3.2 Anode of a Power Cell

The microstructure of power cell anodes strongly differs from the one in energy cell anodes. While both consist of a completely connected system of particles (the so-called active material), the volume fraction of the active material is much lower for power cell anodes. This improves transport properties in the pore phase. Thus, the stochastic microstructure model for energy cell anodes has to be adapted to capture this property. Besides some smaller modifications, which include a dependence of the connectivity graph on the spatial orientation of pairs of particles to each other in order to include the anisotropy that was observed in tomographic image data, the main difference is the inclusion of empty polytopes in the host lattice, where no particles are created in. This allows to account for the low volume fraction of the active material, while preserving realistic shapes of particles.

Host Lattice, Particle Arrangement. The generation of the host lattice is again based on a system of spheres, which is simulated using a modification of the so-called force-biased algorithm [1]. Based on an initial configuration of spheres, an iterative rearrangement is performed until the overlap of spheres falls below a certain threshold. This procedure results in a system of spheres which resembles the properties of the particle system observed in tomographic image data, represented as spheres with volume-equivalent radii. Based on this, the corresponding Laguerre tessellation can be computed, which results in a space-filling system of polytopes.

Connectivity of the Particles. Given the Laguerre tessellation, a graph indicating connectivity between particles is created. To account for the anisotropic shape of particles observed in tomographic image data, the graph is created such that particles are rather connected in horizontal direction than in vertical direction. This means that, besides the size of the Laguerre facet between two points and the distance of those points, the angle with respect to horizontal direction between those points is computed. Starting with a minimum spanning tree to ensure complete connectivity, edges are added based on those three characteristics with a probability such that the mean coordination number of the graph extracted from tomographic image data is matched.

Modification of Host Lattice to Include Empty Polytopes. Particles need to fulfill the connectivity conditions induced by the graph on the one hand, but on the other hand they have to preserve the desired size. This is important to match the volume fraction of active material in the electrode. Therefore, the polytopes where particles are placed in are made smaller by including empty polytopes into the space-filling system. These empty polytopes are found by adding points to the generators of the Laguerre tessellation at the center of those facets where no connectivity is induced by the graph. Moreover, it is ensured that the resulting empty polytopes do not remove facets of the tessellation where two particles are supposed to be connected.

Simulation of Individual Particles. Finally, the particles are modeled using spherical harmonics in the polytopes that have been made smaller in the preceding step. Because of the smaller polytopes, most particles can fulfill their connectivity conditions and required size together with a reasonable shape. In case this is problematic, a more flexible way of setting the boundary conditions to account for the connectivity is used. For details, we refer to [28]. A comparison of a model output to tomographic image data can be found in Fig. 8.

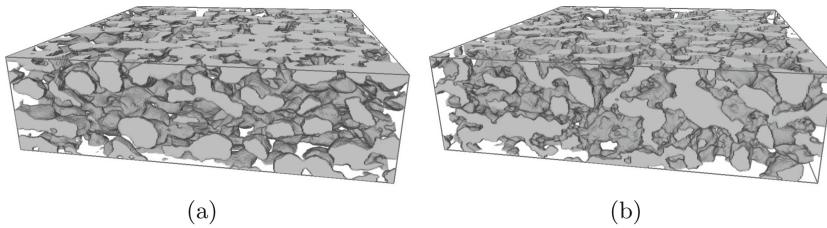


Fig. 8. Comparison of cutout of tomographic image data for a power cell anode (a) and corresponding model realization (b).

3.3 Cathode of an Energy Cell

After having applied the general framework to model and simulate the 3D microstructure of two kinds of anodes, namely the ones in lithium-ion energy and power cells, a third application of the framework concerns modeling and simulation of cathode microstructures (pristine as well as cyclically aged structures). Cathode microstructures also exhibit distinct structural characteristics which made some adaptations necessary. There are three main structural differences observed in cathodes.

Host Lattice, Particle Arrangement. First, there are locally occurring large pores, especially in the case of the pristine cathode. Therefore, when arranging the placement of particles in a similar way as done for power cell anodes, a random marked point pattern is simulated which explicitly generates large polytopes in the host lattice into which no particles are placed later on. These large and empty polytopes mimic large pores in the simulated microstructure.

Connectivity of Particles. Further, cathodes for lithium-ion batteries exhibit another particle connectivity than anodes. That means, the particle system forming the cathode microstructure is not necessarily fully connected anymore as observed in the anode cases. This characteristic feature is captured by omitting the previously used tool of a minimum spanning tree when creating the connectivity graph. Instead, the probability that a pair of particles is connected depends on just two criteria which are determined from the host lattice, see [15] for details. The result is a suitable particle connectivity graph.

Particle Placement, Boundary Conditions and Simulation of Individual Particles. The third and most obvious microstructural difference are the nearly spherical-shaped particles in the cathodes, see Fig. 9. To achieve simulated particles of such shapes, we proceed in three steps:

- The polytopes in the host lattice into which particles will be placed are shrunk to nearly spherical shapes by adding further polytopes which remain empty.
- The number of boundary (contact) conditions per particle is reduced, i.e., now there is only one boundary condition (point) per facet, compare Fig. 7, which guarantees connectivity as claimed by the graph.
- The parameter L at which the series of spherical harmonic functions is truncated is no longer fixed for each particle but dynamically chosen depending on the number of connected neighboring particles. This number is also called the coordination number of a particle and it is known for each particle from the connectivity graph. At the end, a higher number of connected neighboring particles means a larger L and, vice versa, a smaller coordination number means a smaller L .

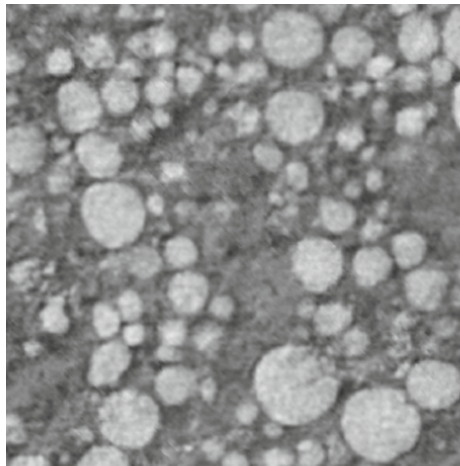


Fig. 9. Tomographic grayscale image - 2D slice of a cutout - showing the microstructure of a pristine cathode.

All three steps lead to less restricted and less degenerated particles and allow them to have nearly spherical shapes as desired.

To give a short overview, the basic ideas of the application of the general framework to cathodes of an energy cell are summarized and illustrated by 2D sketches in Fig. 10.

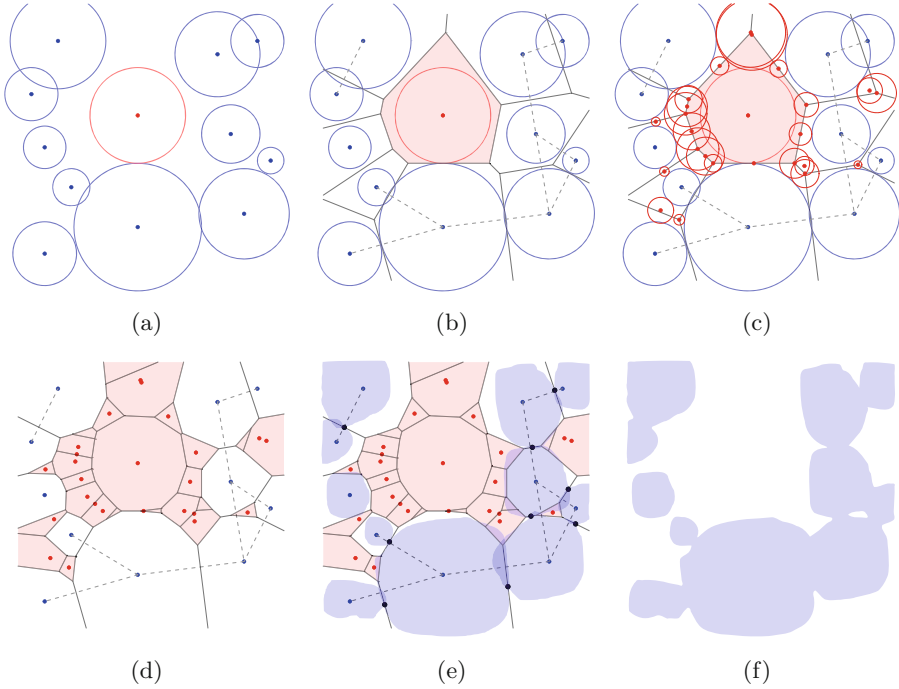


Fig. 10. Overview of the basic cathode modeling ideas. (a) Two random marked point patterns are realized, where the blue dots and circles induce particles and the red ones induce large pores; (b) A connectivity graph (dashed gray lines) based on the random marked point patterns and the corresponding Laguerre tessellation (black lines) is simulated, where the red shaded polytope indicates an (empty) pore polytope (i.e., no particle is placed into); (c) Additional marked points (further red dots and circles) are determined that induce further pore polytopes; (d) Final arrangement of particle polytopes (i.e., a particle is placed into) and pore polytopes (red shaded) is computed, where the initial connectivity is still retained; (e) Particles fulfilling boundary conditions (black dots) are created in the corresponding polytopes using spherical harmonics; (f) Only the particles are kept and morphological smoothing operations lead to the final particle system. (Color figure online)

3.4 Applications of Simulated Structures

The main fields of application for microstructure models of energy materials is the design of new structures with better functionality due to structural improvements. This is done with a method which we call virtual materials testing.

The idea is to generate virtual structures from the models where the parameters are varied in a certain range around the values determined for real materials. This gives us microstructures that are realistic in the sense that they can be manufactured with known production processes on the one hand. On the other hand, these microstructures differ from the known ones and can possibly have more preferable properties. A similar procedure is used in [12, 26] to investigate the relation between microstructure and charge transport properties.

As the models introduced above reconstruct microstructures of lithium-ion cell electrodes, the functionality or performance of the material has to be determined by spatially resolved electrochemical simulations. The theoretical background is described in [16, 17].

For anodes of energy cells the microstructure model has been validated using the electrochemical simulation model [14] where it has been shown that there is a good agreement of the functional performance of virtual and experimental microstructures, see Fig. 11.

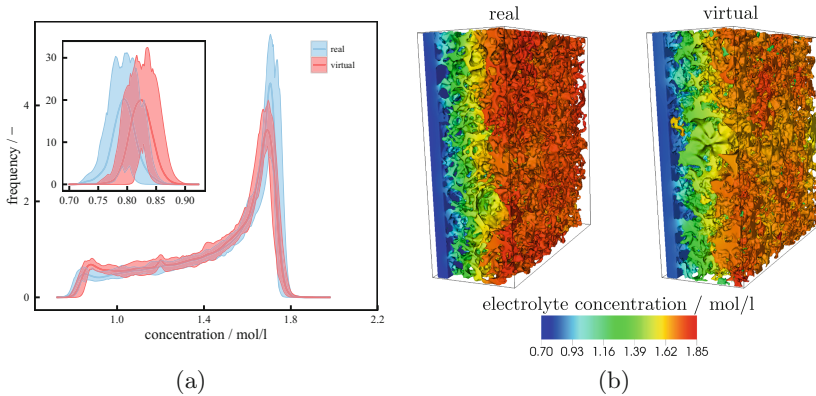


Fig. 11. (a) The distribution of the electrolyte concentration for the electrode pore space and the separator (inset). The density is normed to unity area. The small peak at the initial concentration indicates unconnected pore volume. (b) Spatial distribution of electrolyte concentration for two cut-outs: real (left) and virtual (right). Both have the same color scale (shown below). Larger particles can be seen in both structures. Also both cut-outs show electrolyte pores, which are less connected to the main pore space: (virtual) orange part close to the blue and (real) dark red at the upper corner. Reprinted from [14] with permission from Elsevier. (Color figure online)

Furthermore, a workflow was developed to automatize the whole process and also to speed up the simulation of many cycles which is necessary for studies on the aging of the cells. This workflow is described in detail in [8].

4 Summary, Related Work and Outlook

A general framework for the simulation of battery electrode microstructures is presented. The framework consists of multiple steps that can be adopted for a wide range of electrode materials. Three microstructure models for different electrodes of lithium-ion cells based on this framework are considered. Finally, a validation of the model using spatially resolved electrochemical simulations is mentioned and further applications of such microstructure models are discussed.

Currently, the modeling tools are extended to incorporate cracked particles. In [22], an application of machine learning for the detection of broken particles in tomographic images for negative electrode materials of lithium-ion batteries has been proposed. In [29], an approach for structural segmentation of defective particles has been developed, which also accounts for cracks and holes in particles. The automated detection of cracked particles combined with the parametric representation of individual particles described above enables statistical investigations of the relationship between the morphology of particles and their cracking behavior.

Furthermore, investigations of the influence of different degrees of compression or multi-layered constructions of positive electrode materials on the properties of lithium-ion batteries, e.g., on their ion transport behavior, are possible using the tools described in the present paper.

References

1. Bezrukov, A., Bargie, M., Stoyan, D.: Statistical analysis of simulated random packings of spheres. *Particle Particle Syst. Charact.* **19**(2), 111–118 (2002)
2. Chen, C.F., Mukherjee, P.P.: Probing the morphological influence on solid electrolyte interphase and impedance response in intercalation electrodes. *Phys. Chem. Chem. Phys.* **17**(15), 9812–9827 (2015)
3. Chiu, S.N., Stoyan, D., Kendall, W.S., Mecke, J.: *Stochastic Geometry and Its Applications*, 3rd edn. Wiley, Chichester (2013)
4. Cho, S., Chen, C.F., Mukherjee, P.P.: Influence of microstructure on impedance response in intercalation electrodes. *J. Electrochem. Soc.* **162**(7), A1202–A1214 (2015)
5. Dunn, J.B., Gaines, L., Barnes, M., Wang, M., Sullivan, J.: Material and energy flows in the materials production, assembly, and end-of-life stages of the automotive lithium-ion battery life cycle. Technical report, Argonne National Laboratory (ANL) (2012)
6. Feinauer, J., Brereton, T., Spetl, A., Weber, M., Manke, I., Schmidt, V.: Stochastic 3D modeling of the microstructure of lithium-ion battery anodes via Gaussian random fields on the sphere. *Comput. Mater. Sci.* **109**, 137–146 (2015)
7. Feinauer, J., Spetl, A., Manke, I., Strege, S., Kwade, A., Pott, A., Schmidt, V.: Structural characterization of particle systems using spherical harmonics. *Mater. Charact.* **106**, 123–133 (2015)
8. Feinauer, J., Hein, S., Rave, S., Schmidt, S., Westhoff, D., Zausch, J., Iliev, O., Latz, A., Ohlberger, M., Schmidt, V.: MULTIBAT: unified workflow for fast electrochemical 3D simulations of lithium-ion cells combining virtual stochastic microstructures, electrochemical degradation models and model order reduction. *J. Comput. Sci.* (2018, in print)

9. Gaiselmann, G., Neumann, M., Holzer, L., Hocker, T., Prestat, M., Schmidt, V.: Stochastic 3D modeling of LSC cathodes based on structural segmentation of FIB-SEM images. *Comput. Mater. Sci.* **67**, 48–62 (2013)
10. Gaiselmann, G., Thiedmann, R., Manke, I., Lehnert, W., Schmidt, V.: Stochastic 3D modeling of fiber-based materials. *Comput. Mater. Sci.* **59**, 75–86 (2012)
11. Gaiselmann, G., Neumann, M., Holzer, L., Hocker, T., Prestat, M.R., Schmidt, V.: Stochastic 3D modeling of $\text{La}_{0.6}\text{Sr}_{0.4}\text{CoO}_{3-\delta}$ cathodes based on structural segmentation of FIB-SEM images. *Comput. Mater. Sci.* **67**, 48–62 (2013)
12. Gaiselmann, G., Neumann, M., Schmidt, V., Pecho, O., Hocker, T., Holzer, L.: Quantitative relationships between microstructure and effective transport properties based on virtual materials testing. *AIChE J.* **60**(6), 1983–1999 (2014)
13. Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. *Ann. Hist. Comput.* **7**(1), 43–57 (1985)
14. Hein, S., Feinauer, J., Westhoff, D., Manke, I., Schmidt, V., Latz, A.: Stochastic microstructure modeling and electrochemical simulation of lithium-ion cell anodes in 3D. *J. Power Sour.* **336**, 161–171 (2016)
15. Kuchler, K., Westhoff, D., Feinauer, J., Mitsch, T., Manke, I., Schmidt, V.: Stochastic model of the 3D microstructure of Li-ion battery cathodes under various cyclical aging scenarios. *Model. Simul. Mater. Sci. Eng.* **26**, 035005 (2018)
16. Latz, A., Zausch, J.: Thermodynamic consistent transport theory of Li-ion batteries. *J. Power Sour.* **196**, 3296–3302 (2011)
17. Latz, A., Zausch, J.: Thermodynamic derivation of a Butler-Volmer model for intercalation in Li-ion batteries. *Electrochimica Acta* **110**, 358–362 (2013)
18. Lautensack, C., Zuyev, S.: Random Laguerre tessellations. *Adv. Appl. Probab.* **40**(3), 630–650 (2008)
19. Mościński, J., Bargiel, M., Rycerz, Z., Jacobs, P.: The force-biased algorithm for the irregular close packing of equal hard spheres. *Mol. Simul.* **3**(4), 201–212 (1989)
20. Münch, B., Holzer, L.: Contradicting geometrical concepts in pore size analysis attained with electron microscopy and mercury intrusion. *J. Am. Ceram. Soc.* **91**(12), 4059–4067 (2008)
21. Newman, J., Thomas, K., Hafezi, H., Wheeler, D.: Modeling of lithium-ion batteries. *J. Power Sour.* **119**, 838–843 (2003)
22. Petrich, L., Westhoff, D., Feinauer, J., Finegan, D.P., Daemi, S.R., Shearing, P.R., Schmidt, V.: Crack detection in lithium-ion cells using machine learning. *Comput. Mater. Sci.* **136**, 297–305 (2017)
23. Pfaffmann, L., Birkenmaier, C., Müller, M., Bauer, W., Mitsch, T., Feinauer, J., Scheiba, F., Hintennach, A., Schleid, T., Schmidt, V., Ehrenberg, H.: Investigation of the electrochemical active surface area and lithium diffusion in graphite anodes by a novel OsO₄ staining method. *J. Power Sour.* **307**, 762–771 (2016)
24. Remmlinger, J., Tippmann, S., Buchholz, M., Dietmayer, K.: Low-temperature charging of lithium-ion cells Part II: model reduction and application. *J. Power Sour.* **254**, 268–276 (2014)
25. Stenzel, O., Koster, L., Thiedmann, R., Oosterhout, S., Janssen, R., Schmidt, V.: A new approach to model-based simulation of disordered polymer blend solar cells. *Adv. Funct. Mater.* **22**, 1236–1244 (2012)
26. Stenzel, O., Pecho, O., Holzer, L., Neumann, M., Schmidt, V.: Predicting effective conductivities based on geometric microstructure characteristics. *AIChE J.* **62**(5), 1834–1843 (2016)
27. Torquato, S.: *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*. Springer, New York (2002)

28. Westhoff, D., Feinauer, J., Kuchler, K., Mitsch, T., Manke, I., Hein, S., Latz, A., Schmidt, V.: Parametric stochastic 3D model for the microstructure of anodes in lithium-ion power cells. *Comput. Mater. Sci.* **126**, 453–467 (2017)
29. Westhoff, D., Finegan, D.P., Shearing, P.R., Schmidt, V.: Algorithmic structural segmentation of defective particle systems: a lithium-ion battery study. *J. Microsc.* **270**, 71–82 (2018)



On Microstructure-Property Relationships Derived by Virtual Materials Testing with an Emphasis on Effective Conductivity

Matthias Neumann¹(✉), Orkun Furat¹, Dzmitry Hlushkou², Ulrich Tallarek², Lorenz Holzer³, and Volker Schmidt¹

¹ Institute of Stochastics, Ulm University,
Helmholtzstraße 18, 89069 Ulm, Germany
matthias.neumann@uni-ulm.de

² Department of Chemistry, Philipps-Universität Marburg,
Hans-Meerwein-Straße 4, 35032 Marburg, Germany

³ Institute of Computational Physics, ZHAW,
Wildbachstrasse 21, 8400 Winterthur, Switzerland

Abstract. Based on virtual materials testing, which combines image analysis, stochastic microstructure modeling and numerical simulations, quantitative relationships between microstructure characteristics and effective conductivity can be derived. The idea of virtual materials testing is to generate a large variety of stochastically simulated microstructures in short time. These virtual, but realistic microstructures are used as input for numerical transport simulations. Finally, a large data basis is available to study microstructure-property relationships quantitatively by classical regression analysis and tools from statistical learning. The microstructure-property relationships obtained for effective conductivity can also be applied to Fickian diffusion. For validation, we discuss an example of Fickian diffusion in porous silica monoliths on the basis of 3D image data.

1 Introduction

The functionality of many materials, like, e.g., solar cells [1], batteries [2], fuel cells [3] or silica monoliths used for catalysis [4], is strongly influenced by their microstructure. Thus an optimal design of the microstructure regarding effective macroscopic properties of these materials would lead to an improvement of their functionality. This kind of microstructure optimization, in turn, requires an

M. Neumann and V. Schmidt—The work of MN and VS has been partially funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) under grant 03ET6095E.

D. Hlushkou and U. Tallarek—The work of DH and UT has been supported by the Deutsche Forschungsgemeinschaft DFG (Bonn, Germany) under grant TA 268/9-1.

© Springer Nature Switzerland AG 2018

M. Baum et al. (Eds.): SimScience 2017, CCIS 889, pp. 145–158, 2018.

https://doi.org/10.1007/978-3-319-96271-9_9

understanding of the quantitative relationships between microstructure characteristics and effective macroscopic properties, which are – so far – only available for some special types of simple microstructures [5] like, e.g., the coated spheres model introduced in [6].

A direct approach to investigate relationships between microstructure and effective macroscopic properties is based on tomographic 3D imaging of microstructures. Among other methods, X-ray tomography [7], FIB-SEM tomography [8] and STEM tomography [9] provide highly resolved information about microstructures on different length scales. On the one hand, 3D image data can be analyzed by tools from spatial statistics [10] and mathematical morphology [11], which allows the computation of well-defined microstructure characteristics. On the other hand, 3D image data can be used as an input information for the numerical simulation of effective macroscopic properties, like, e.g., effective diffusivity in silica monoliths [12]. This combination of image analysis and numerical simulation enables a direct investigation of the relationship between well-defined microstructure characteristics and effective macroscopic properties [12–16]. However, this approach is limited as it is not possible to investigate a sufficiently large data set of different microstructures due to the high costs of 3D imaging.

Thus an alternative approach – called virtual materials testing – was suggested where image analysis and numerical simulations are combined with stochastic 3D microstructure modeling [17]. By the aid of stochastic modeling virtual, but realistic, microstructures can be generated in short time. So a large data set of virtual microstructures with a wide range of microstructure characteristics is generated in order to study the quantitative relationship between microstructure characteristics and effective macroscopic properties efficiently. In the present paper, we give an overview of the results having been obtained for the microstructure influence on electric conductivity by virtual materials testing [17–19]. Furthermore, we show how the results can be used to predict effective diffusivity in silica monoliths.

Before we give a detailed description on how to simulate the considered stochastic 3D microstructure model in Sect. 3, the microstructure characteristics which are related to effective conductivity are described in Sect. 2. The results of virtual materials testing with respect to electric conductivity are reviewed in Sect. 4. It is shown in Sect. 5 that the results can be used to predict effective diffusivity in silica monoliths the morphology of which has been analyzed in [20]. Conclusions are given in Sect. 6.

2 Microstructure Characteristics and M -factor

To investigate the microstructure influence on effective conductivity σ_{eff} we consider the microstructure characteristics volume fraction ε , mean geometric tortuosity τ_{geom} , mean geodesic tortuosity τ_{geod} and constrictivity β . These microstructure characteristics are computed based on voxelized 3D images

representing (virtual or real) microstructures. Moreover, using $\varepsilon, \tau_{\text{geom}}, \tau_{\text{geod}}, \beta$ the M -factor

$$M = \sigma_{\text{eff}}/\sigma_0 \quad (1)$$

is predicted, where σ_0 denotes the intrinsic conductivity of the considered material.

2.1 M -factor

As described in [18], we consider conductive transport processes within porous materials with one conducting phase, the intrinsic conductivity of which is σ_0 .

Let J denote the current density, σ the space-dependent conductivity function, U the electric potential and t the time. Then, electric charge transport is described by Ohm's law

$$J = -\sigma \frac{\partial U}{\partial x} \quad (2)$$

and

$$\frac{\partial U}{\partial t} = \sigma \frac{\partial^2 U}{\partial x^2}. \quad (3)$$

Assuming constant boundary conditions, such systems converge to an equilibrium which is described by the Laplace equation

$$\Delta U = 0. \quad (4)$$

By averaging over J and U as described in [5] we obtain the effective conductivity σ_{eff} and thus the M -factor of the microstructure. For the numerical simulation of M the software NM-SESES [21] has been used in [17], while GeoDict [22] has been used in [18, 19].

2.2 Tortuosity

Both characteristics, τ_{geom} and τ_{geod} describe the length of paths through the conducting phase relative to the thickness of the material. Note that – in contrast to concepts of effective tortuosity – the characteristics τ_{geom} and τ_{geod} depend only on the geometry of the microstructure. An overview on the available concepts of tortuosity in the literature is given in [23]. To determine τ_{geod} , we compute for each voxel of the conducting phase located at the start layer, i.e. the plane where the conduction process starts, the shortest path through the conducting phase to the goal layer in terms of geodesic distance [24]. Such a shortest path is visualized in Fig. 1(left). Mean geodesic tortuosity τ_{geod} is defined as the average of these path lengths divided by the thickness of the material. Mean geometric tortuosity τ_{geom} is also defined as an average of shortest path lengths, where shortest paths are computed on a skeleton of the conducting phase, see Fig. 1(right). The skeleton of a microstructure is a network of medial axes through the conducting phase [24]. In [17] and [18] the software Avizo [25] has been used to extract the skeleton from 3D image data.

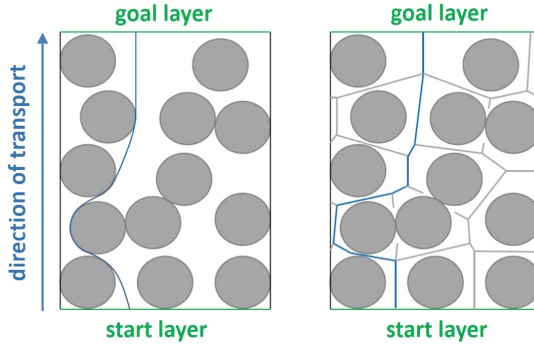


Fig. 1. Visualization of two different kinds of shortest path corresponding to τ_{geom} (center, right) and τ_{geod} (left) in 2D. The conducting phase is represented in white, while sphericle obstacles are represented in grey. The shortest paths from the start layer to the goal layer are visualized in blue. The skeleton on which the paths for τ_{geom} are computed is represented by a grey network (right). Reprinted from [18], Fig. 3, with permission of J. Wiley & Sons. (Color figure online)

2.3 Constrictivity

Constrictivity β quantifies bottleneck effects. It has been defined for a single tube with periodic constrictions in [26] by $\beta = (r_{\min}/r_{\max})^2$, where r_{\min} and r_{\max} are the radius of the minimum and maximum cross-section of the tube, respectively. This concept of constrictivity has been generalized defining r_{\min}, r_{\max} and thus β for complex microstructures [27]. As described in [19], r_{\max} is defined as the 50% quantile of the continuous pore size distribution, while r_{\min} is defined as the 50% quantile of the MIP pore size distribution, which is based on a geometrical simulation of mercury intrusion porosimetry, introduced in [28]. Constrictivity takes values between 0 and 1, where values close to 0 indicate strong bottleneck effects while values close to 1 indicate that there are no bottlenecks at all.

3 Stochastic 3D Microstructure Model

In [17], a parametric stochastic 3D microstructure model has been developed, which has been used to simulate virtual microstructures with many different constellations for the microstructure characteristics $\varepsilon, \tau_{\text{geom}}$ (or τ_{geod}) and β . We recall the definition of the model introduced in [17] and give a detailed description on how to simulate model realizations.

3.1 Model Definition

Using tools of stochastic geometry and graph theory, we define a random set Ξ representing the conducting phase of a microstructure. At first a random geometric graph $G = (V, E)$ is modeled consisting of a set of vertices V and a

set of edges E , which are connections between the vertices. In a second step the edges of the graph are randomly dilated to get a full-dimensional conducting phase.

The set of vertices V is modeled by a homogeneous Poisson point process with some intensity $\lambda > 0$, see e.g. [29]. This means that the vertices are distributed completely at random in the three-dimensional space and the expected number of points per unit volume is given by λ . Edges between pairs of vertices are put according to the rule of a generalized relative neighborhood graph (RNG) introduced in [17]. For a parameter $\alpha \in \mathbb{R}$, two vertices $v_1, v_2 \in V$ are connected by an edge if there is no other vertex $v_3 \in V \setminus \{v_1, v_2\}$ such that

$$d_\alpha(v_1, v_2) > \max\{d_\alpha(v_1, v_3), d_\alpha(v_2, v_3)\}, \tag{5}$$

where

$$d_\alpha(v_1, v_2) = d(v_1, v_2) \max \left\{ 0.01, \left(1 - \frac{2\varphi(v_1, v_2)}{\pi} \right) \right\}^\alpha. \tag{6}$$

Here $d(v_1, v_2)$ denotes the Euclidean distance from v_1 to v_2 , and $\varphi(v_1, v_2)$ denotes the acute angle between the line segments $v_2 - v_1$ and $(0, 0, 1)$. By the aid of the model parameter α , it is possible to control the directional distribution of edges in the graph G and thus one can control τ_{geom} and τ_{geod} in the virtual microstructure. Note that for $\alpha = 0$ we are in the situation of the classical RNG [30], i.e., there is no preferred orientation of the edges. For $\alpha > 0$ we obtain more edges oriented in direction $(0, 0, 1)$ than for $\alpha < 0$, see Fig. 2.

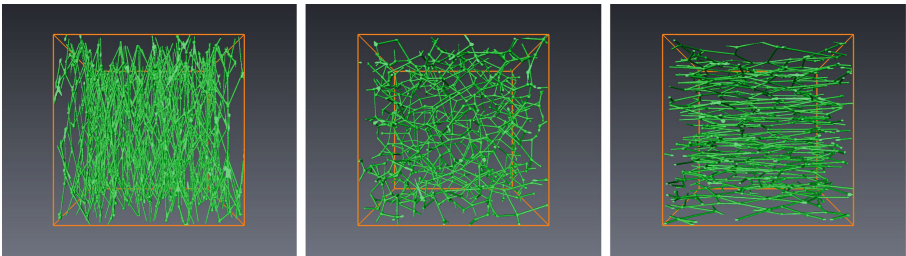


Fig. 2. Generalized RNG, where the set of vertices is given by a realization of a homogeneous Poisson point process. The parameter α is chosen as $\alpha = -5$ (left, vertically oriented edges), $\alpha = 0$ (center, no preferred orientation of edges), and $\alpha = 5$ (right, horizontally oriented edges). Direction $(0, 0, 1)$ is the direction from front to back. Reprinted from [17], Fig. 5, with permission of J. Wiley & Sons.

In order to result in a full-dimensional conducting phase each edge of G is dilated by a sphere with random radius. The dilation radii are independently and identically distributed (i.i.d) following a Gamma-distribution with mean value $g_1 > 0$ and variance $g_2 > 0$. The Gamma-distribution is shifted by 1 to ensure that each edge is at least dilated by a ball of radius 1. Formally, we result in a conducting phase

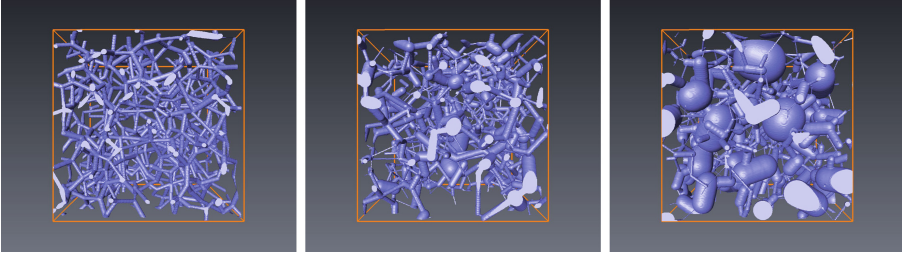


Fig. 3. Three realizations of the final conducting phase, where λ, α and g_1 are constant. Only the parameter g_2 is varied, that is $g_2 = 1$ (left), $g_2 = 4$ (center), and $g_2 = 7$ (right). Reprinted from [17], Fig. 6, with permission of J. Wiley & Sons.

$$\Xi = \bigcup_{e \in E} e \oplus B(o, R_e), \quad (7)$$

where \oplus denotes the Minkowski addition of sets, R_e denotes the random dilation radius corresponding to the edge $e \in E$, and $B(o, r)$ denotes the (closed) ball centered at the origin $o \in \mathbb{R}^3$ with radius $r > 0$. While the volume fraction of Ξ can be controlled by the intensity $\lambda > 0$ and g_1 , constrictivity can be controlled by g_2 . The larger the variance of the dilation radii is, the more bottlenecks are created in the conductive phase, see Fig. 3.

3.2 Simulation of Model Realizations

Having defined the stochastic 3D microstructure model, the simulation of model realizations is described. In the following we assume algorithms for the simulation of uniformly distributed, Poisson distributed and Gamma distributed random variables to be known. For simulation of random variables the reader is referred to [31]. To simulate a realization of Ξ in a cuboid $W = [0, w]^3$ for given model parameters $\lambda, \alpha, g_1, g_2$, we simulate the generalized RNG G at first. The graph G is simulated based on a realization of vertices in a larger observation window $W^+ = [-\delta, w + \delta]^3$ for some $\delta > 0$ in order to avoid edge effects. Note that this approach approximates a realization of Ξ since in general, we can not guarantee that vertices located outside of W^+ do not influence the conductive phase in W , i.e., the random set $\Xi \cap W$. Nevertheless, the approximation error becomes neglectable in practice for sufficiently large W^+ .

To simulate G we begin with the simulation of the set of vertices $V = \{v_1, \dots, v_N\}$ in W^+ , where N is the random number of vertices in W^+ . That means that a homogeneous Poisson process has to be simulated in W^+ .

Algorithm 1 (Simulation of vertices [29]). *Let $\lambda > 0$ be the intensity of the homogeneous Poisson point process. Then, the set of vertices is simulated in W^+ by the following two step approach*

1. Simulate a realization n of the random number N of vertices in W^+ , which is Poisson distributed with parameter $\lambda(w + 2\delta)^3$.
2. For each $i \in \{1, \dots, n\}$, simulate the position of the i -th vertex v_i , which is uniformly distributed in W^+ . That is, v_i is a three-dimensional vector each entry of which is uniformly distributed in the interval $[-\delta, w + \delta]$.

For a given set of vertices V , the edges are put and their random dilation radii are simulated to obtain the full-dimensional conductive phase.

Algorithm 2 (Simulation of the full-dimensional conductive phase). Consider the set of vertices $V = \{v_1, \dots, v_n\}$ simulated by Algorithm 1.

1. Check for all $1 \leq i < j \leq n$ whether v_i and v_j are connected by an edge in G according to Inequality (5).
2. Simulate the dilation radii for the edges, i.e., simulate a collection of i.i.d. Gamma distributed random variables with mean g_1 and variance g_2 resulting in a realization $\{r_e : e \in E\}$ of the random radii $\{R_e : e \in E\}$.
3. The set

$$W \cap \bigcup_{e \in E} e \oplus B(o, r_e) \quad (8)$$

is an approximation of a realization of $\Xi \cap W$.

4 Prediction of Effective Conductivity

A number of 43 virtual microstructures is generated with different constellations for the microstructure characteristics ε , $\tau_{\text{geom}}/\tau_{\text{geod}}$ and β . These virtual microstructures are used as an input for numerical simulations of the M -factor. As a result the prediction formulas

$$\widehat{M} = \min \left\{ 1, \max \left\{ 0, 2.03 \frac{\varepsilon^{1.57} \beta^{0.72}}{\tau_{\text{geom}}^2} \right\} \right\}, \quad (9)$$

and

$$\widehat{M} = \frac{\varepsilon^{1.15} \beta^{0.37}}{\tau_{\text{geod}}^{4.39}}. \quad (10)$$

for the M -factor have been obtained by regression analysis in [17, 18], respectively. The mean absolute percentage error (MAPE) is 16% when using Eq. (9) and 18% when using Eq. (10). A visualization of the goodness of fit is given in Fig. 4.

On the one hand, a prefactor of 2.03 is necessary in Eq. (9) to obtain an adequate prediction of M by ε , β and τ_{geom} . On the other hand $2.03 \varepsilon^{1.57} \beta^{0.72} \tau_{\text{geom}}^{-2}$ is in general not within the interval $[0, 1]$ and has thus to be artificially restricted to $[0, 1]$. Moreover, τ_{geom} depends on the specific choice of the skeletonization algorithm and skeletonization might be problematic for microstructures with a large volume fraction of the conductive phase. Even if the MAPE of Eq. (9) is smaller than the one of Eq. (10), the conclusion of [18] is to prefer τ_{geod} rather

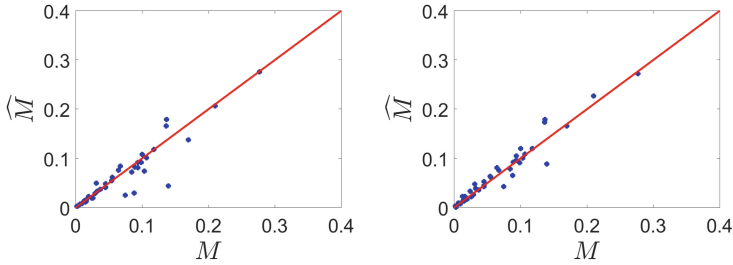


Fig. 4. Predicted M -factor \widehat{M} over numerically simulated M -factor M for the 43 virtual microstructures from [17,18]. Equation (9) (left) and Eq. (10) (right) are used to predict M .

than τ_{geom} as microstructure characteristic describing the length of transport paths, in particular for the prediction of the M -factor.

An extension of this simulation study has been performed in [19], where 8119 virtual microstructures have been generated and the microstructure characteristics $\varepsilon, \beta, \tau_{\text{geod}}$ as well as the M -factor have been computed for each of them. Aside from the model presented in Sect. 3, the stochastic microstructure model from [32] has been used to generate virtual microstructures, where the conducting phase is a union of spherical particles or its complement. Considering different types of stochastic microstructure models, we want to ensure that the virtual testing approach is influenced by the specific kind of generating virtual microstructures as little as possible. It has been shown that the prediction of the M -factor can be further improved using methods from statistical learning [33], i.e. neural networks (9% MAPE) and random forests (8% MAPE), see Fig. 5. The MAPE of Eq. (10) reduces to 13% when all 8119 virtual structures are taken into account because then, the extreme microstructures play a less important role [17,18]. Furthermore, the extension of the simulation study has shown that Eq. (10) slightly underestimates the M -factors close to 1. Validation with experimental image data representing microstructures in SOFC anodes and

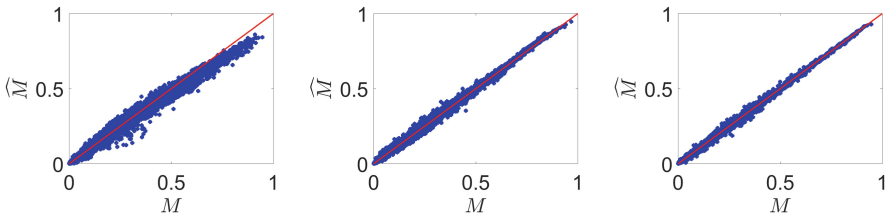


Fig. 5. Predicted M -factor \widehat{M} over numerically simulated M -factor M for the 8119 virtual microstructures from [19]. Classical regression analysis, i.e., Eq. (10) (left), as well as a trained neural network (center) and random forest (right) are used to predict M . Reprinted from [19], Fig. 6, with permission of J. Wiley & Sons.

microstructures in membranes of pH-sensors has been performed in [19], which shows that the prediction formulas are not only valid for model-based virtual microstructure.

5 Application to Fickian Diffusion in Silica Monoliths

For porous microstructures, there is a one-to-one relationship between effective conductivity and effective diffusivity of Fickian diffusion. We consider 3D image data of two different porous silica monoliths, in which Fickian diffusion takes place, as a further validation of the prediction formulas obtained by virtual materials testing. Since the diffusion takes place in the pores, the microstructure characteristics ε , τ_{geod} and β are computed for the pores in this section.

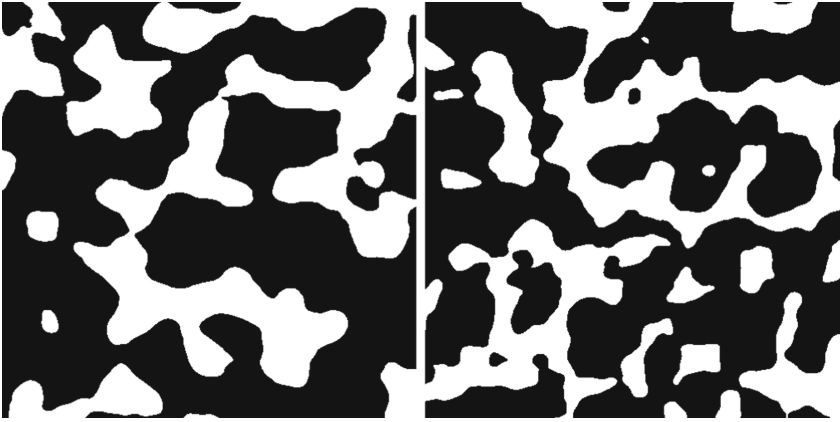


Fig. 6. Cutout of cross-sections ($15\ \mu\text{m} \times 15\ \mu\text{m}$) depicting the macropores (black) of the two silica monolith samples S_1 (left) and S_2 (right).

5.1 Description of the Material

The considered silica monoliths, manufactured by Merck, are used for high-performance liquid chromatography (HPLC) and catalysis. They consist of a single piece of silica with an interconnected pore phase, which contains macropores and mesopores [34]. Diffusion in the macropores is investigated based on tomographic image data of two samples, denoted by S_1 and S_2 with a resolution of 30 nm and sizes of $60 \times 60 \times 24.19\ \mu\text{m}^3$ and $60 \times 60 \times 24.57\ \mu\text{m}^3$, respectively. The mean size of the macropores is $1.36\ \mu\text{m}$ in sample S_1 and $0.82\ \mu\text{m}$ in sample S_2 . The image data depicting the macropores, see Fig. 6, is obtained via confocal laser scanning microscopy, see [12].

5.2 Effective Diffusivity

The diffusion process over time of a particle concentration u with intrinsic diffusivity D_0 in the pore phase can be analogously described to the conduction process, i.e. by the diffusion equation

$$\frac{\partial u}{\partial t} = D_{\text{eff}} \Delta u, \quad (11)$$

where D_{eff} denotes the effective diffusivity of the pore space. Due to the analogy between effective diffusivity and effective conductivity, the following relationship

$$M = \varepsilon \frac{D_{\text{eff}}}{D_0} \quad (12)$$

holds. A random-walk particle-tracking algorithm [12] is applied in order to simulate D_{eff} for the previously described cubic cut-outs. This method simulates a large number K of independent random walks $r_i(t) = (r_{i,x}(t), r_{i,y}(t), r_{i,z}(t))$ over time in the pore phase, where $r_i(t)$ is the position of the i -th random-walk at time $t > 0$. Then, the normalized effective diffusion coefficient D_{eff}/D_0 is defined by

$$\frac{D_{\text{eff}}}{D_0} = \lim_{t \rightarrow \infty} \frac{1}{2K D_0} \frac{\partial}{\partial t} \sum_{i=1}^K \left(r_{i,z}(t) - r_{i,z}(0) - \frac{1}{K} \sum_{j=1}^K (r_{j,z}(t) - r_{j,z}(0)) \right)^2. \quad (13)$$

Note that $r_{i,z}(t) - r_{i,z}(0)$ is the displacement of the i -th random-walk in z -direction at time t . It is important to note that we consider only displacements in z -direction since mean geodesic tortuosity τ_{geod} and the constrictivity β are computed in z -direction. The accuracy of the random-walk particle tracking algorithm used for the simulation of D_{eff} has been demonstrated in [35–37].

5.3 Results

For each of the two microstructures, we consider 64 (slightly overlapping) cubic cutouts with sizes of $15 \times 15 \times 15 \mu\text{m}^3$. The microstructure characteristics $\varepsilon, \tau_{\text{geod}}, \beta$ and D_{eff} are computed for each of these cutouts, which allows to validate the predictions for $M = \varepsilon D_{\text{eff}}/D_0$ derived by virtual materials testing. While the MAPE is 6% when using Eq. (10), it can be reduced using the predictions obtained by neural networks (2%) and random forests (2%). The neural network and random forest, which are trained in [19], are used for the prediction of $M = \varepsilon D_{\text{eff}}/D_0$. The goodness-of-fit is visualized in Fig. 7. One can observe that $M = \varepsilon D_{\text{eff}}/D_0$ is slightly underestimated by Eq. (10). However, the deviations are not larger than the ones from the virtual microstructures considered in [19].

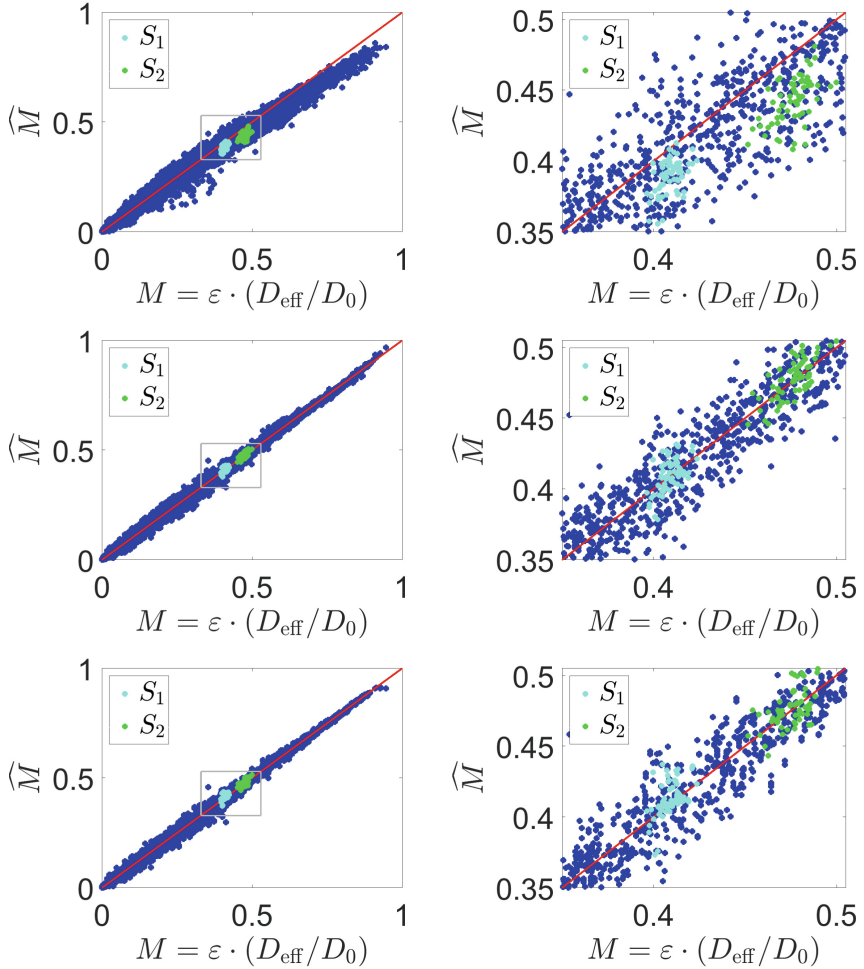


Fig. 7. The values of $M = \varepsilon D_{\text{eff}}/D_0$ computed by the random-walk particle-tracking algorithm are predicted using Eq. (10) (top), the neural network (center), and the random forest (bottom) from [19]. The values corresponding to the 8119 virtual microstructures (blue), to the cutouts from sample S_1 (cyan), and to the cutouts from sample S_2 (green) are represented in different colors. The right column shows a zoom to the square $[0.33, 0.53]^2$. (Color figure online)

6 Conclusions

Virtual materials testing is a powerful tool to establish quantitative relationships between microstructure characteristics and functional properties. An overview of results obtained by virtual materials testing to predict effective conductivity by volume fraction ε , mean geodesic tortuosity τ_{geod} and constrictivity β is given. The presented quantitative relationships enable the identification of improved

microstructures with respect to effective conductivity. Due to the mathematical analogy, the obtained results can be transferred from conduction processes to Fickian diffusion in order to predict the effective diffusivity in porous microstructures. This is exemplarily demonstrated based on 3D image data of two different microstructures from silica monoliths. The method of virtual materials testing itself is not restricted to conduction processes or Fickian diffusion in two-phase materials. It can also be used to investigate relationships between microstructure characteristics and further functional properties, like e.g. effective permeability or mechanical stress-strain curves.

References

1. DeQuilettes, D., Vorpahl, S.M., Stranks, S.D., Nagaoka, H., Eperon, G.E., Ziffer, M.E., Snaith, H.J., Ginger, D.S.: Impact of microstructure on local carrier lifetime in perovskite solar cells. *Science* **348**, 683–686 (2015)
2. Wilson, J.R., Cronin, J.S., Barnett, S.A., Harris, S.J.: Measurement of three-dimensional microstructure in a LiCoO₂ positive electrode. *J. Power Sour.* **196**(7), 3443–3447 (2011)
3. Prakash, B.S., Kumar, S.S., Aruna, S.T.: Properties and development of Ni/YSZ as an anode material in solid oxide fuel cell: a review. *Renew. Sustain. Energy Rev.* **36**, 149–179 (2014)
4. Nischang, I.: Porous polymer monoliths: morphology, porous properties, polymer nanoscale gel structure and their impact on chromatographic performance. *J. Chromatogr. A* **1287**, 39–58 (2013)
5. Torquato, S.: *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*. Springer, New York (2002)
6. Hashin, Z.: The elastic moduli of heterogeneous materials. *J. Appl. Mech.* **29**(1), 143–150 (1962)
7. Maire, E., Withers, P.J.: Quantitative X-ray tomography. *Int. Mater. Rev.* **59**(1), 1–43 (2014)
8. Holzer, L., Cantoni, M.: Review of FIB-tomography. In: Utke, I., Moshkalev, S., Russell, P., (eds.) *Nanofabrication Using Focused Ion and Electron Beams: Principles and Applications*, pp. 410–435. Oxford University Press, New York (2012)
9. Midgley, P.A., Dunin-Borkowski, R.E.: Electron tomography and holography in materials science. *Nat. Mater.* **8**(4), 271 (2009)
10. Chiu, S.N., Stoyan, D., Kendall, W.S., Mecke, J.: *Stochastic Geometry and Its Applications*, 3rd edn. Wiley, Chichester (2013)
11. Matheron, G.: *Random Sets and Integral Geometry*. Wiley, New York (1975)
12. Hlushkou, D., Hormann, K., Höltzel, A., Khirevich, S., Seidel-Morgenstern, A., Tallarek, U.: Comparison of first and second generation analytical silica monoliths by pore-scale simulations of eddy dispersion in the bulk region. *J. Chromatogr. A* **1303**, 28–38 (2013)
13. Doyle, M., Fuller, T.F., Newman, J.: Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *J. Electrochem. Soc.* **140**(6), 1526–1533 (1993)
14. Holzer, L., Iwanschitz, B., Hocker, T., Keller, L., Pecho, O.M., Sartoris, G., Gasser, P., Münch, B.: Redox cycling of Ni-YSZ anodes for solid oxide fuel cells: influence of tortuosity, constriction and percolation factors on the effective transport properties. *J. Power Sour.* **242**, 179–194 (2013)

15. Shikazono, N., Kanno, D., Matsuzaki, K., Teshima, H., Sumino, S., Kasagi, N.: Numerical assessment of SOFC anode polarization based on three-dimensional model microstructure reconstructed from FIB-SEM images. *J. Electrochem. Soc.* **157**(5), B665–B672 (2010)
16. Tippmann, S., Walper, D., Balboa, L., Spier, B., Bessler, W.G.: Low-temperature charging of lithium-ion cells part I: electrochemical modeling and experimental investigation of degradation behavior. *J. Power Sour.* **252**, 305–316 (2014)
17. Gaiselmann, G., Neumann, M., Pecho, O.M., Hocker, T., Schmidt, V., Holzer, L.: Quantitative relationships between microstructure and effective transport properties based on virtual materials testing. *AIChE J.* **60**(6), 1983–1999 (2014)
18. Stenzel, O., Pecho, O.M., Holzer, L., Neumann, M., Schmidt, V.: Predicting effective conductivities based on geometric microstructure characteristics. *AIChE J.* **62**, 1834–1843 (2016)
19. Stenzel, O., Neumann, M., Pecho, O.M., Holzer, L., Schmidt, V.: Big data for microstructure-property relationships: a case study of predicting effective conductivities. *AIChE J.* **63**(9), 4224–4232 (2017)
20. Stoeckel, D., Kübel, C., Hormann, K., Hölzel, A., Smarsly, B.M., Tallarek, U.: Morphological analysis of disordered macroporous-mesoporous solids based on physical reconstruction by nanoscale tomography. *Langmuir* **30**, 9022–9027 (2014)
21. NM-SESES. <http://nmtec.ch/nm-seses/>. Accessed 2017
22. GeoDict (2017). www.geodict.com
23. Clennell, M.B.: Tortuosity: a guide through the maze. *Geol. Soc. London Spec. Publ.* **122**, 299–344 (1997)
24. Soille, P.: *Morphological Image Analysis: Principles and Applications*. Springer, New York (2003)
25. VSG - Visualization Sciences Group - Avizo Standard (2017). <http://www.vsg3d.com/>
26. Petersen, E.E.: Diffusion in a pore of varying cross section. *AIChE J.* **4**(3), 343–345 (1958)
27. Holzer, L., Iwanschitz, B., Hocker, T., Keller, L., Pecho, O.M., Sartoris, G., Gasser, P., Münch, B.: The influence of constrictivity on the effective transport properties of porous layers in electrolysis and fuel cells. *J. Mater. Sci.* **48**, 2934–2952 (2013)
28. Münch, B., Holzer, L.: Contradicting geometrical concepts in pore size analysis attained with electron microscopy and mercury intrusion. *J. Am. Ceram. Soc.* **91**, 4059–4067 (2008)
29. Møller, J., Waagepetersen, R.P.: *Statistical Inference and Simulation for Spatial Point Processes*. Chapman & Hall/CRC, Boca Raton (2004)
30. Jaromczyk, J.W., Toussaint, G.T.: Relative neighborhood graphs and their relatives. *Proc. IEEE* **80**, 1502–1517 (1992)
31. Ross, S.: *Simulation*, 5th edn. Academic Press, New York (2013)
32. Stenzel, O., Hassfeld, H., Thiedmann, R., Koster, L.J.A., Oosterhout, S.D., van Bavel, S.S., Wienk, M.M., Loos, J., Janssen, R.A.J., Schmidt, V.: Spatial modeling of the 3D morphology of hybrid polymer-ZnO solar cells, based on electron tomography data. *Ann. Appl. Stat.* **5**, 1920–1947 (2011)
33. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*, 2nd edn. Springer, New York (2008)
34. Enke, D., Gläser, R., Tallarek, U.: Sol-gel and porous glass-based silica monoliths with hierarchical pore structure for solid-liquid catalysis. *Chemie Ingenieur Technik* **88**(11), 1561–1585 (2016)

35. Liasneuski, H., Hlushkou, D., Khirevich, S., Höltzel, A., Tallarek, U., Torquato, S.: Impact of microstructure on the effective diffusivity in random packings of hard spheres. *J. Appl. Phys.* **116**(3), 034904 (2014)
36. Hlushkou, D., Liasneuski, H., Tallarek, U., Torquato, S.: Effective diffusion coefficients in random packings of polydisperse hard spheres from two-point and three-point correlation functions. *J. Appl. Phys.* **118**(12), 124901 (2015)
37. Daneyko, A., Hlushkou, D., Baranau, V., Khirevich, S., Seidel-Morgenstern, A., Tallarek, U.: Computational investigation of longitudinal diffusion, eddy dispersion, and trans-particle mass transfer in bulk, random packings of core-shell particles with varied shell thickness and shell diffusion coefficient. *J. Chromatogr. A* **1407**, 139–156 (2015)

Distributed Simulations



Simulating Software Refactorings Based on Graph Transformations

Daniel Honsel¹(✉), Niklas Fiekas², Verena Herbold¹, Marlon Welter¹, Tobias Ahlbrecht², Stephan Waack¹, Jürgen Dix², and Jens Grabowski¹

¹ Institute of Computer Science, Georg-August-Universität Göttingen, Goldschmidtstraße 7, 37077 Göttingen, Germany
daniel.honsel@cs.uni-goettingen.de

² Department of Informatics, Clausthal University of Technology, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany

Abstract. We aim to *simulate* software processes in order to predict the structural evolution of software graphs and assure higher *software quality*. To make our simulation and therefore the results more accurate, we need to model real world practices. In this paper, we consider the specific problem of including software refactorings in our simulation. We describe these refactorings as graph transformations and apply parameters we collected from open source projects.

1 Introduction

In previous work (see Honsel et al. 2016; 2015) we presented agent-based simulation models for software processes in order to monitor the quality of the software under simulation. The aim is to assure the quality of software projects. Project managers can configure different parameters, for example, the number of developers involved and can compare the quality of different simulation runs.

In these models, developers act on *software entities*. Thus, it is their behavior which makes the software evolve over time. One identified challenge is to improve the structure of the simulated software graph. In order to do so, we require a more detailed description of which files are selected for a commit and how these files change. A common approach to describe well defined code structure changes are *software refactorings*. Therefore, we propose a general model for refactorings which can be easily integrated in our agent-based model developed earlier.

This work is based on well known theoretical foundations of software refactorings (see Fowler 2009) and graph transformations (see Ehrig et al. 1999). The idea to use graph transformations for the description of refactorings is based on the work of Mens et al. (2005), where the authors proved that refactorings can be formalized with graph transformations. Unfortunately, their proposed model is too detailed for simulation purposes. We therefore define a suitable abstraction of it.

This paper is structured as follows. Before we describe our approach in detail in Sect. 4 we present some related work in Sect. 2 as well as the theoretical

background in Sect. 3. We present our experiments and results in Sect. 5. Finally, we conclude with a discussion of our results and a short outlook in Sect. 6.

2 Related Work

There are only few approaches in the field of monitoring software quality with simulation methods. An agent-based simulation model for software processes was presented by Smith and Ramil (2006). The authors can reproduce different aspects of software evolution, e.g. the number of complex entities, the number of touches, and distinct patterns for system growth. In our tests, almost all of them need different parameter sets to get realistic results. The model we proposed in Honsel et al. (2016), has the following differences to the one presented by Smith et al.: First, our model is not grid-based and agents do not perform a random walk. In our work, all instantiated agents live in one environment and relationships are represented as graphs. Second, our simulation model requires only parameters for effort and size to simulate projects that have similar growth trends. The approach discussed in this paper extends our previous simulation model.

Another interesting study is presented by Spasic and Onggo (2012). The work is aimed to support project managers in their planning by simulating possible future software processes. It is not an entirely new approach to use simulation in this context, but for a long time it was dominated by discrete event simulation and system dynamics (because agent based simulation is a relatively new technique). The authors use data from a software department in an industrial context to estimate the simulation parameters. This work differs from other studies in that a maturity model is given (the capability maturity model integration, CMMI¹). During the creation of the agent-based model, the number of existing software components and the number of available developers is considered based on the design and the development phase. Then, the developers are assigned to certain (multiple) components. The components switch between different states. Finally, the model is validated by comparing the empirical project duration of different projects with the simulated results.

3 Background

In this section, we briefly introduce the required theory of software refactorings and graph transformations as well as the main idea of software developers' goals and plans and how one can simulate them. We will start with a short introduction of our previous work, in particular the agent-based model for simulating software evolution. In this paper, we will extend this model with three prominent refactoring types.

¹ <http://cmmiinstitute.com/>.

3.1 Agent-Based Simulation Model of Software Processes

The model that is to be extended is shown in Fig. 1. We are using agent-based modeling and simulation because software evolution results from the work of developers, who we can model in greater detail.

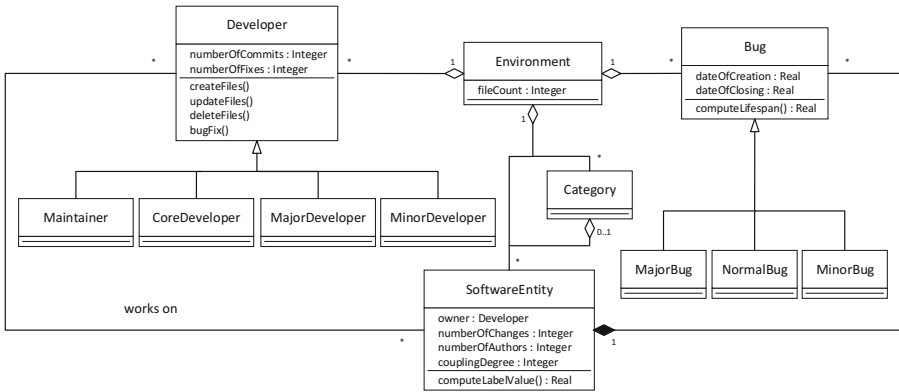


Fig. 1. Agent-based simulation model for software evolution (Honsel et al. 2016).

The main entity of our model is the developer who works on software entities. This work, performed as commits, makes the software evolve over time. Commits can be separated into the following three actions: *create*, *delete*, *update*. Applying a commit changes the state of several software entities simultaneously and can fix a bug with a certain probability.

Developers are divided into four different types which differ in the effort they spend on the project. Thus, each developer type has its own commit behavior. Bugs are divided into three different types according to their severity.

Required dependencies between entities are modeled as graphs. The most important one is the *ChangeCouplingGraph*: this graph represents dependencies between software entities that are changed together several times. This means that the structure of this graph depends on the developers’ commit behavior. In particular, the process of selecting files for a commit influences the structure of the change coupling graph. This process is partly based on random decisions. To reduce the randomness in selecting files for a particular commit, we extend our model with refactorings. This allows us to model structural changes according to rules for certain refactoring types.

Since the change coupling graph serves as input for our assessment tool, improvements to this graph are expected to result in more accurate quality assessments of the overall software under simulation.

This entire model including the introduced dependency graphs, the mining process to get the required parameters to run the simulation model, and the automated assessment using conditional random fields are described in detail in (Honsel et al. 2016; 2014; 2015).

3.2 Refactoring

The following definition of a refactoring as well as the description of the selected refactorings corresponds to Martin Fowlers definition².

Refactoring means a restructuring of software without changing its behavior. The aim is to make the software more readable to developers and/or to increase the maintainability.

Fowler 2009 presents a catalog with more than 70 different refactoring types. The mining process described in Sect. 4.2 revealed that the most frequently occurring refactoring types, that change the structure of the software graph, are *Move Method*, *Extract Method* and *Inline Method*. Therefore, we started with models for those three types, which are described below.

Move Method will be applied if a method calls more methods or features of another class than from its own. It moves the method to the class with the most calls.

Extract Method will be applied to large methods or if code fragments can be grouped together. It creates a new method that is called from the old one and moves code from the old method to the new one. In other words, the original method has been split.

Inline Method is the opposite of Extract Method. The code of the inlined method is moved to the calling method and afterwards the inlined method will be removed.

How our simulation model is extended by refactoring types using graph transformations is described in Sect. 4.1.

3.3 Graph Transformations

Structural changes to a software graph caused by applying refactorings are modeled using graph transformations as presented in (Mens et al. 2005). The following definition of graph transformations is based on (Kreowski et al. 2006).

Graph Transformations are used to transform a given graph G according to predefined rules. A rule has a left-hand side L and a right-hand side R , both are graphs. To apply a rule one has to execute three steps, which finally lead to the derived graph H . Firstly, one has to find a matching of L in G . Secondly, all vertices and edges that are matched by $L \setminus R$ are deleted from G . Finally, the graph H is created by pasting a copy of $R \setminus L$ to the result.

To restrict the allowed software graphs one can use a *type graph*. It is similar to a UML class diagram and expresses which nodes can be linked with a certain edge type.

How we use graph transformations to extend our model of software evolution and how the concrete type graph is defined is presented in Sect. 4.1.

² <https://refactoring.com>.

3.4 Developer Goals and Plans

As we have described more detailed in (Ahlbrecht et al. 2017), we are using the prominent BDI (Weiss 2013) approach to model refactorings. This means that developers formulate goals based on their beliefs and build plans to reach them. In the scenario for this paper we only consider the developer's goal *improve maintainability*, which results in applying a certain refactoring. Beliefs are the current state of the software under simulation, represented as software and graph metrics. The decision process of a developer is depicted in Fig. 2.

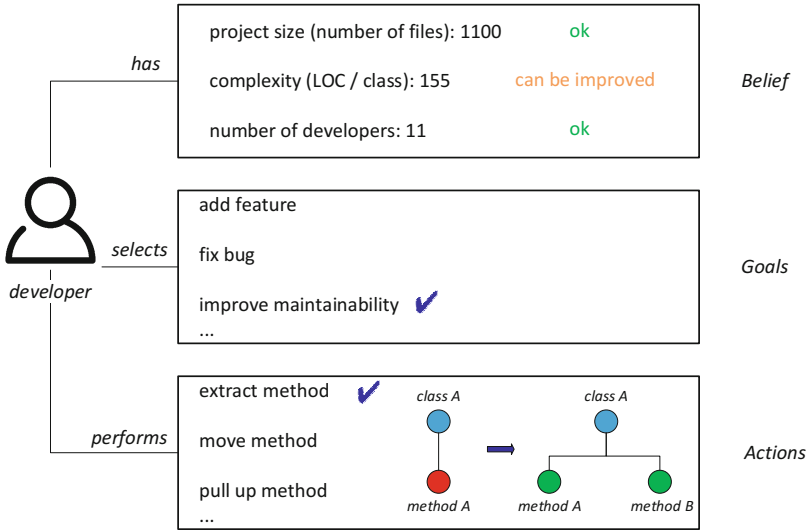


Fig. 2. Example for developer's goals and plans (Ahlbrecht et al. 2017). The developer works on a method that is hard to maintain because it has too many lines of code. To improve maintainability, the developer applies the refactoring *Extract Method*. This means that parts of the origin method are moved to a newly created method which is called from the origin method.

Such an approach has additional requirements for the simulation platform. In particular we want to take advantage of BDI while still scripting other parts in a classical programming language. Furthermore it is desirable to parallelize the simulation of multiple developers in large projects. We use a new simulation platform based on Python and Apache Spark that we presented in (Ahlbrecht et al. 2016).

Our BDI agents are programmed in AgentSpeak (Rao 1996). The beliefs of an agent are represented as logical terms like `calls("Config", "get", "HashMap", "get")`. Plans consist of a header and instructions which may remove and add beliefs with - and + operators. Unification on the plan headers is used to find suitable plans for the current situation. An example for a specific plan is presented in Sect. 4.3.

To create a plan for each considered refactoring we require detailed information about the state of the software before it is applied and state changes caused by the refactoring. This includes metric changes as well as structural graph changes. This information is gathered through mining software repositories as described in Sect. 4.2.

4 Approach

In this section, we present the model used to simulate refactorings based on graph transformations. Furthermore, we describe the mining approach to extract the required information from open source repositories. With this information we can describe developers' actions, in particular the structural changes to the software graph, and we can parametrize the model. Finally, we present the concrete BDI model and how we simulate it with our developed simulation platform.

4.1 General Refactoring Model

The graph representing the simulated software can be found in Definition 1. It is transformed by commits of the developers according to formulated transformation rules. Due to the simplicity of this graph one can easily extend it. For the modeling of inheritance, for example, one can add a further edge label representing links between classes that belong to an inheritance hierarchy.

Definition 1. Let $\Sigma = \{C, M\}$ be a set of node labels and $\Delta = \{mm, mc\}$ be a set of edge labels. The node types represent software entities: classes (C) and methods (M). Edges represent relationships between nodes: method memberships (mm) and method calls (mc). A graph over Σ and Δ is a System $G = (V, E, l)$, where V is a set of nodes, $l : V \rightarrow \Sigma$ is a function that assigns a label to each node, and $E \subseteq V \times \Delta \times V$ is the set of edges.

The *type graph* depicted in Fig. 3 restricts the edge creation. For instance, member method edges can only occur between a class and a method as well as method calls link two methods.



Fig. 3. Type Graph

We formulated transformation rules for the three most frequently occurring refactoring types. The results of the mining process for these types can be found in Sect. 5.1. The transformation rules are depicted in Fig. 4. One can see the left hand side of the rule which has to be replaced by the right hand side.

Since Extract Method is actually the opposite of Inline Method both are depicted in Fig. 4a.

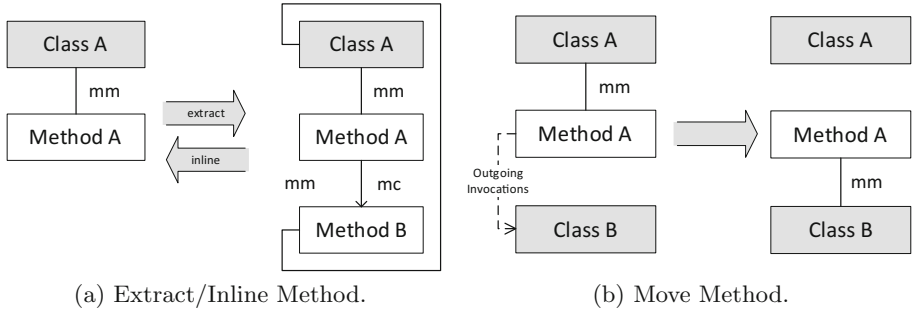


Fig. 4. Graph transformation rules for the selected refactorings. The left-hand side of the rule will be replaced by the right-hand side in the software graph. The dotted edge represents outgoing invocations, which means method calls to one or more methods of the other class.

To find a match for the rule’s left-hand side, not only the structure of the software graph is taken into account, but also appropriate software metrics. These are, for example, the size measured in lines of code (LOC) for *Extract Method* (Fig. 4a) and the coupling measured in number of outgoing invocations (NOI) for *Move Method* (Fig. 4b). Furthermore, we assume that NOI of the origin class of a method will be reduced when a *Move Method* is applied.

4.2 Collecting Required Parameters (Mining)

To apply the model described above to a software graph, we require detailed information about the state of the software before a refactoring is applied and how the state changes when a certain refactoring was applied. The state is represented by software metrics for size, complexity, and coupling. This section describes the entire process of gathering these metrics from open source repositories as depicted in Fig. 5.

Since we are interested in the behavior of single refactorings, we analyze transitions between two consecutive code revisions in a Git repository. For this purpose, we iterate over the version history of the software repository and search for refactorings in each transition. When a refactoring is found, the metrics of this transition will be computed.

We are using Ref-Finder (Prete et al. 2010) for the identification of refactorings between two commits. This tool implements 63 refactorings of Fowler’s catalog (Fowler 2009) and is based on logic programming.

To verify the results of Ref-Finder, we apply the recently published tool RefDiff (Silva and Valente 2017). RefDiff is able to detect 13 prominent refactoring types of Fowler’s catalog (Fowler 2009) and uses heuristics based on static code analysis as well as code similarity metrics to identify refactorings between two code versions.

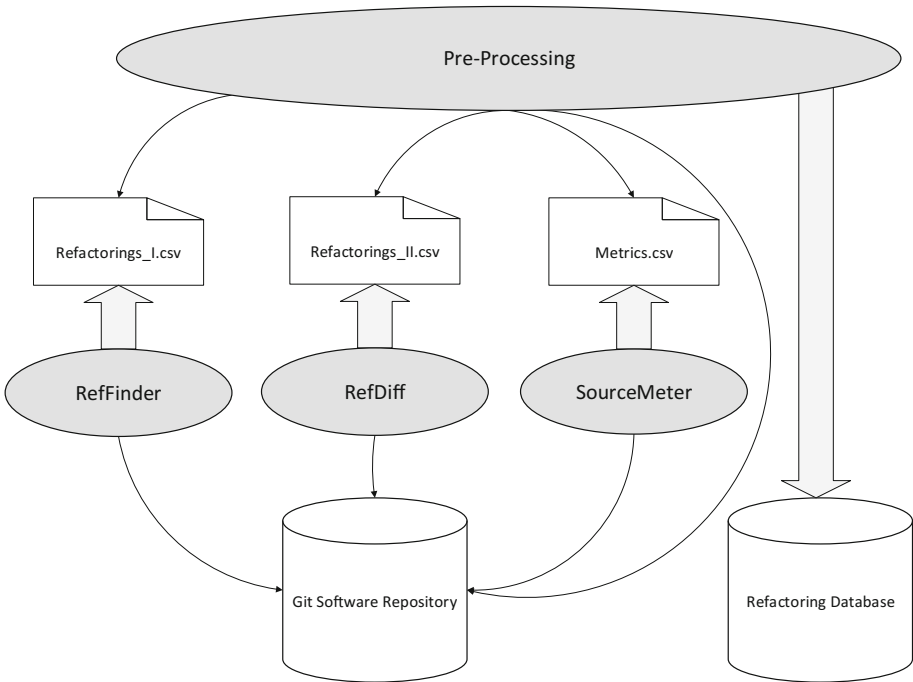


Fig. 5. The framework used to find refactorings in software repositories.

Both tools for finding refactorings between two commits are executed in parallel. As presented in Sect. 5.1, the results of RefDiff are more accurate for certain refactoring types.

If a transition between two commits contains at least one refactoring we are using SourceMeter³, a tool for static source code analysis, to retrieve the required source code metrics for classes and methods of both commits.

All these tools store their results in text files. Unfortunately, naming conventions of class or method names are not unique. To facilitate the analysis of the generated output of the three tools we have written a preprocessing application. This application analyses refactorings found between two commits based on the output of Ref-Finder and RefDiff, searches the appropriate metrics in the output of SourceMeter, computes metric changes, and enriches this dataset with information of the Git repository. One dataset for each found refactoring is stored in a MySQL database. The results presented in Sect. 5.1 are based on queries against this database.

³ <https://www.sourcemeeter.com/>.

4.3 Simulating Refactorings

The simulation is initialized with a snapshot of the code base representing one revision in the source code repository. This includes the graph from Sect. 4.1 represented as logical terms. Additionally, methods are annotated with the source code metrics for size *LOC* and for complexity *CC* (McCabe's Cyclomatic Complexity).

class (Class). Logical term stating that there is a class **Class**. This refers to classes which are defined in the code base under simulation, excluding the standard library and other dependencies.

method (Class, Method, LOC, CC). There is a method **Method** that belongs to **Class**, defined in the current codebase.

calls (A, Caller, B, Callee). The method **Caller** from class **A** calls method **Callee** from class **B**.

Then in each step a refactoring is randomly selected from the applicable plans. Plans are weighted using the frequency of the corresponding refactoring as mined from the source repository.

```
+!move_method(Class, Method, NewClass)
  : method(Class, Method, LOC, CC) &
    Class \== NewClass
<-
  -method(Class, Method, LOC, CC);
  +method(NewClass, Method, LOC, CC);
  while (calls(Class, Method, CalleeClass, CalleeMethod)) {
    -calls(Class, Method, CalleeClass, CalleeMethod);
    +calls(NewClass, Method, CalleeClass, CalleeMethod);
  }
  while (calls(CallerClass, CallerMethod, Class, Method)) {
    -calls(CallerClass, CallerMethod, Class, Method);
    +calls(CallerClass, CallerMethod, NewClass, Method);
  }.
```

Fig. 6. Example: Definition of the *Move Method* refactoring in AgentSpeak

Figure 6 shows how the terms from the snapshot are used in the definition of the `+!move_method (Class, Method, NewClass)` plan. The plan header requires that the moved method is present in the current code base and that **Class** and **NewClass** are not the same.

`+!extract_method (Class, Method)` and `+!inline_method (Class, Method)` are defined similarly: *Extract Method* requires that the origin method has more than one line of code and complexity greater than 1, *Inline Method* requires only that the method is defined in the current code base.

5 Results

In this section, we present our results. First, we summarize the outputs of mining open source repositories and analyze the consequences for our proposed model of refactorings. Second, we present results of simulating refactorings on our platform.

5.1 Mined Simulation Parameters and Patterns

To gain more knowledge about when software refactorings are applied and how they change the state of the software we analyzed the following three open source projects: JUnit⁴—a unit testing framework for Java, MapDB⁵—an embedded database engine for Java, and the GameController⁶ used for several RoboCup competitions. For the analysis of these projects the mining framework described in Sect. 4.2 is used.

The whole result set to parameterize the model described in Sect. 4.3 is depicted in the table in Appendix A. The values represent average metric changes for each analyzed refactoring type.

Due to the surprisingly high amount of Move Method refactorings found by Ref-Finder, we compare the results of Ref-Finder for this refactoring type with the results of RefDiff. This resulted in significantly less Move Method refactorings found by RefDiff. Thus, our results correspond with the outcome of the authors of RefDiff, who found in (Silva and Valente 2017) that Ref-Finder has a high number of false positives for some analyzed projects, in particular for the Move Method refactoring.

Another notable result concerns the target classes of the refactoring Move Method. As presented in Table 1, only a small amount of methods are moved to already existing classes. This means that in many cases where the refactoring is applied, methods are moved to newly created classes.

Table 1. The amount of applied refactorings of the type Move Method where the method is moved to an already existing class.

| Project | Tool | |
|----------------|------------|---------|
| | Ref-Finder | RefDiff |
| JUnit | 5,6% | 22,5% |
| MapDB | 34,9 % | 28,0% |
| GameController | 9,6% | 0,0% |

⁴ <https://github.com/junit-team/junit4>.

⁵ <https://github.com/jankotek/mapdb>.

⁶ <https://github.com/bhuman/GameController>.

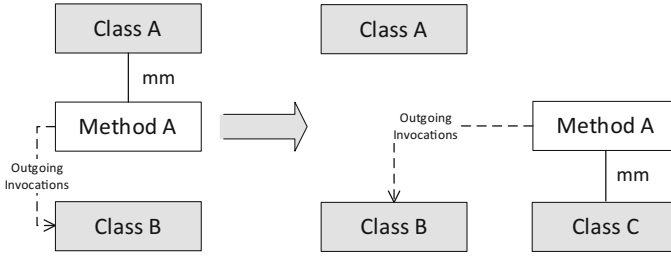


Fig. 7. Alternative rule for refactoring Move Method.

Considering this knowledge made us extend the model for Move Method presented in Sect. 4.1 as depicted in Fig. 7. The method is moved to a newly created class and the outgoing invocations move as well to the new class. This allows us to apply both versions of this type with a certain probability.

Using the results presented above to parameterize the model gives us the ability to run the simulation. The outcome of the simulation is presented in the next section.

5.2 Simulation

Figure 8 shows the simulation results for the three projects. JUnit, GameController and MapDB have similar method sizes in terms of LOC, but vastly different average method complexities to start with.

JUnit and GameController were simulated for 365 steps. The general trend is that complexity is traded for size. Note however, that LOC changes on a much smaller scale. For example extracting a method will add new lines for the function signature, closing brace and call into the new method, but can dramatically reduce complexity.

In the simulation of MapDB no further sensible refactorings were found before the predetermined step count, so the simulation ends effectively at step 265. This shows that other commit types, not based purely on refactorings, should be included in the simulation.

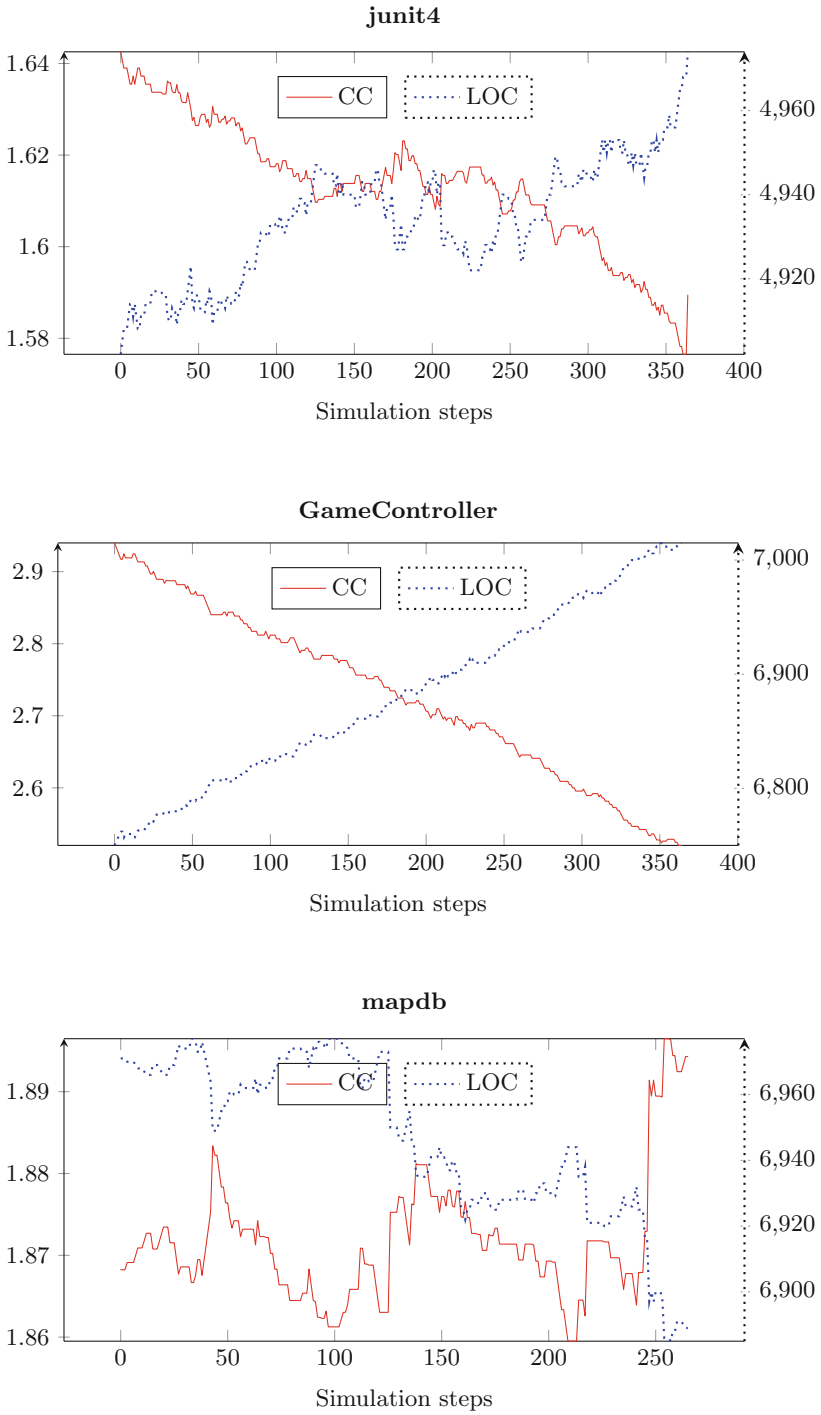


Fig. 8. Average method complexity and size in simulation of software evolution.

6 Conclusion and Outlook

In this paper, we presented an approach to model refactorings based on graph transformations: single rules are applied to initial graphs (snapshots of software repositories). The simulation results reflect a tradeoff between complexity and size.

For integrating this model completely in our simulation model for software processes *we need more information* about how other commit types like adding a feature or fixing a bug behave in detail. With this knowledge we are able to create transformation rules for other commit types which are required to simulate the entire process of software evolution. Furthermore, we plan to analyze and integrate more refactoring types which influence the structure of the software graph. These are, for example, *PullUpMethod* and *ExtractSuperclass*. We expect that agent based modelling on our new scalable simulation platform will help efficiently simulate detailed developer behavior (Ahllbrecht et al. 2017), even in large projects with many developers.

For future data mining tasks we plan to use the existing mining framework SMARTShark (Trautsch et al. 2016), it mines data from software repositories automatically and provides users with the ability to analyze the data with Apache Spark. Thus, we need to extend SMARTShark with a plug-in for finding refactorings in revisions of source code repositories based on the mining approach described in this paper.

Future simulation models should be able to answer more questions of project managers, for example, how different refactoring plans influence the quality of software projects.

Acknowledgment. The authors thank the SWZ Clausthal-Göttingen that partially funded our work (both the former projects “Simulation-based Quality Assurance for Software Systems” and “DeSim”, and the recent project “Agent-based simulation models in support of monitoring the quality of software projects”). <https://www.simzentrum.de/en/>.

A Mining Results

We analyzed the following metrics: size measured in lines of code (LOC), coupling measured in number of outgoing invocations (NOI), and complexity measured in McCabes cyclomatic complexity (McCC for methods) or weighted methods per class (WMC for classes). The results are presented in the following table.

| Move Method - refinder | | | | | | | | | | | | | | | | | | | |
|------------------------|----------------|------------------|-------|--------|--------------------|------|-------|--------------|-------|-------|------------------|------|-------|--------------------|------|------|--------------|------|------|
| Project | Analyzed Items | Delta Base Class | | | Delta Target Class | | | Delta Method | | | Start Base Class | | | Start Target Class | | | Start Method | | |
| | | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | MCC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | MCC |
| junit | 1766 | -28.33 | -2.57 | -4.37 | 29.23 | 2.82 | 4.54 | -0.05 | 0.00 | -0.01 | 35.18 | 3.37 | 5.60 | 4.14 | 0.40 | 0.74 | 5.39 | 1.23 | 1.23 |
| gc | 136 | -54.19 | -2.74 | -7.88 | 52.51 | 2.83 | 8.05 | -0.10 | -0.03 | -0.08 | 61.68 | 3.26 | 9.02 | 0.94 | 0.11 | 0.19 | 10.42 | 1.38 | 2.36 |
| mdb | 3114 | -169.18 | -6.45 | -26.76 | 159.29 | 6.12 | 25.12 | -0.08 | -0.02 | -0.02 | 194.75 | 7.63 | 30.81 | 25.71 | 0.85 | 4.76 | 9.35 | 1.19 | 2.00 |

| Move Method - refdiff | | | | | | | | | | | | | | | | | | | |
|-----------------------|----------------|------------------|--------|--------|--------------------|-------|-------|--------------|-------|-------|------------------|-------|-------|--------------------|------|-------|--------------|------|------|
| Project | Analyzed Items | Delta Base Class | | | Delta Target Class | | | Delta Method | | | Start Base Class | | | Start Target Class | | | Start Method | | |
| | | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | MCC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | MCC |
| junit | 356 | -45.86 | -5.06 | -8.60 | 56.59 | 6.51 | 10.65 | 0.03 | -0.04 | 0.00 | 84.60 | 8.94 | 15.35 | 19.87 | 1.79 | 3.90 | 5.64 | 1.84 | 1.41 |
| gc | 22 | -176.09 | -4.68 | -32.09 | 181.14 | 5.36 | 37.27 | -1.64 | 0.14 | 0.23 | 209.95 | 7.23 | 38.64 | 0.00 | 0.00 | 0.00 | 15.73 | 0.91 | 4.09 |
| mdb | 229 | -391.79 | -12.94 | -55.58 | 348.22 | 13.13 | 47.89 | 0.07 | 0.03 | -0.05 | 509.48 | 23.78 | 68.52 | 104.34 | 4.44 | 18.45 | 8.83 | 1.39 | 1.96 |

| Extract Method | | | | | | | | | | | | | | | | |
|----------------|----------------|-------------|------|------|-------------------|-------|-------|------------------|------|------|-------------|-------|--------|------------------|------|------|
| Project | Analyzed Items | Delta Class | | | Delta Base Method | | | Delta New Method | | | Start Class | | | Start New Method | | |
| | | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC |
| junit | 222 | 13.23 | 0.49 | 2.80 | -1.42 | -0.07 | -0.21 | 5.16 | 1.31 | 1.45 | 161.13 | 12.35 | 26.11 | 9.78 | 3.12 | 2.47 |
| gc | 12 | 23.50 | 0.42 | 5.67 | -12.42 | 0.17 | -0.17 | 14.17 | 0.83 | 2.83 | 249.75 | 11.25 | 38.25 | 59.58 | 4.08 | 6.50 |
| mdb | 104 | 25.87 | 0.25 | 8.25 | -1.42 | -0.39 | -0.54 | 11.90 | 2.05 | 3.70 | 1004.05 | 26.28 | 213.45 | 35.02 | 6.56 | 9.47 |

| Inline Method | | | | | | | | | | | | | | | | | | | |
|---------------|----------------|-------------|-------|--------|---------------------|-------|-------|----------------------|-------|-------|-------------|-------|--------|---------------------|------|-------|----------------------|------|------|
| Project | Analyzed Items | Delta Class | | | Delta Caller Method | | | Delta Inlined Method | | | Start Class | | | Start Caller Method | | | Start Inlined Method | | |
| | | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC | LOC | NOI | WMC |
| junit | 235 | -17.77 | -0.44 | -3.13 | 0.63 | 0.00 | -0.03 | -5.22 | -1.24 | -1.43 | 194.19 | 12.54 | 29.85 | 7.22 | 2.97 | 2.02 | 5.26 | 1.26 | 1.44 |
| gc | 3 | -1.33 | 0.00 | 0.33 | 15.33 | -0.67 | 3.33 | -12.67 | -2.67 | -1.33 | 208.33 | 9.67 | 26.67 | 80.00 | 9.33 | 11.00 | 12.67 | 2.67 | 1.33 |
| mdb | 100 | -52.12 | -2.74 | -16.85 | 1.46 | -1.33 | -0.62 | -15.04 | -2.14 | -4.21 | 929.81 | 23.52 | 203.49 | 35.81 | 7.67 | 8.49 | 16.52 | 2.39 | 4.55 |

References

Ahlbrecht, T., Dix, J., Fiekas, N.: Scalable multi-agent simulation based on MapReduce. In: Criado Pacheco, N., Carrascosa, C., Osman, N., Julián Inglada, V. (eds.) EUMAS/AT -2016. LNCS (LNAI), vol. 10207, pp. 364–371. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59294-7_31

Ahlbrecht, T., et al.: Agent-based simulation for software development processes. In: Criado Pacheco, N., Carrascosa, C., Osman, N., Julián Inglada, V. (eds.) EUMAS/AT -2016. LNCS (LNAI), vol. 10207, pp. 333–340. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59294-7_28

- Ehrig, H., Rozenberg, G., Kreowski, H.J.: Handbook of Graph Grammars and Computing by Graph Transformation, vol. 2. Applications, Languages and Tools. world Scientific (1999)
- Fowler, M.: Refactoring: Improving the Design of Existing Code. Pearson Education India (2009)
- Honsel, D., Honsel, V., Welter, M., Waack, S., Grabowski, J.: Monitoring software quality by means of simulation methods. In: 10th International Symposium on Empirical Software Engineering and Measurement (ESEM) (2016)
- Honsel, V., Honsel, D., Grabowski, J.: Software process simulation based on mining software repositories. In: ICDM Workshop (2014)
- Honsel, V., Honsel, D., Herbold, S., Grabowski, J., Waack, S.: Mining software dependency networks for agent-based simulation of software evolution. In: ASE Workshop (2015)
- Kreowski, H.-J., Klempien-Hinrichs, R., Kuske, S.: Some essentials of graph transformation. *Recent Adv. Formal Lang. Appl.* **25**, 229–254 (2006)
- Mens, T., Van Eetvelde, N., Demeyer, S., Janssens, D.: Formalizing refactorings with graph transformations. *J. Software Maintenance Evol. Res. Pract.* **17**(4), 247–276 (2005)
- Prete, K., Rachatasumrit, N., Sudan, N., Kim, M.: Template-based reconstruction of complex refactorings. In: 2010 IEEE International Conference on Software Maintenance (ICSM), pp. 1–10. IEEE (2010)
- Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J.W. (eds.) MAAMAW 1996. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996). <https://doi.org/10.1007/BFb0031845>
- Silva, D., Valente, M.T.: RefDiff: detecting refactorings in version histories. In: Proceedings of the 14th International Conference on Mining Software Repositories, pp. 269–279. IEEE Press (2017)
- Smith, N., Fernández Ramil, J.: Agent-based simulation of open source evolution. In: Software Process Improvement and Practice (2006)
- Spasic, B., Onggo, B.S.S.: Agent-based simulation of the software development process: a case study at AVL. In: Rose, O., Uhrmacher, A.M. (eds.) Winter Simulation Conference, WSC, pp. 400:1–400:11 (2012). <http://dblp.uni-trier.de/db/conf/wsc/wsc2012.html#SpasicO12>
- Trautsch, F., Herbold, S., Makedonski, P., Grabowski, J.: Addressing problems with external validity of repository mining studies through a smart data platform. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), pp. 97–108. IEEE (2016)
- Weiss, G.: Multiagent Systems. MIT Press (2013). ISBN 9780262018890



Transparent Model-Driven Provisioning of Computing Resources for Numerically Intensive Simulations

Fabian Korte¹(✉), Alexander Bufe², Christian Köhler³,
Gunther Brenner², Jens Grabowski¹, and Philipp Wieder³

¹ Institute of Computer Science, University of Goettingen,
37077 Goettingen, Germany

fabian.korte@cs.uni-goettingen.de

² Institute of Applied Mechanics, Clausthal University of Technology,
38678 Clausthal-Zellerfeld, Germany

³ Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH Goettingen (GWDG),
37077 Goettingen, Germany

Abstract. Many simulations require large amounts of computing power to be executed. Traditionally, the computing power is provided by large high performance computing clusters that are solely built for this purpose. However, modern data centers do not only provide access to these high performance computing systems, but also offer other types of computing resources e.g., cloud systems, grid systems, or access to specialized computing resources, such as clusters equipped with accelerator hardware. Hence, the researcher is confronted with the choice of picking a suitable computing resource type for his simulation and acquiring the knowledge on how to access and manage his simulation on the resource type of choice. This is a time consuming and cumbersome process and could greatly benefit from supportive tooling. In this paper, we introduce a framework that allows to describe the simulation application in a resource-independent manner. It furthermore helps to select a suitable resource type according to the requirements of the simulation application and to automatically provision the required computing resources. We demonstrate the feasibility of the approach by providing a case study from the area of fluid mechanics.

1 Introduction

Scientific simulations are often computation intensive and time consuming and can highly profit from choosing a suitable computing resource type and scale. However, choosing the right computing resource and an appropriate scale is not a trivial task, especially in modern computing centers that offer access to a heterogeneous infrastructure including cloud services, high performance computing clusters and clusters with specialized accelerators (e.g., GPU cards). All of these different resource types have their own technical peculiarities and require the user of the simulation application (the *simulation scientist*), to invest time to learn

when and how to use them. It might even be necessary to switch the resource type and scale during the lifetime of a simulation, e.g., when transitioning from testing simulation code to running parameter studies, or when the problem size increases and suddenly requires more computing resources. To overcome this burden, we develop a transparent integration mechanism for heterogeneous computing resources. The goal is to semi-automatically deploy and execute simulation applications on the most suitable resource type. To achieve this goal, we model the simulation application structure and behaviour in a resource-agnostic way and provide model transformers that automatically transform the abstract simulation application model into resource-specific models. In this paper, we introduce a conceptual framework that implements the concept of model-driven provisioning of computing resources for simulation application and provide a case study on the application of the framework by modeling and deploying a simulation from fluid mechanics.

The specific problems addressed in this work are:

- P1** Developing and/or using a simulation application and making that application run on various resource types are in principle orthogonal tasks. However both of these tasks are handled by the simulation scientist in practice.
- P2** The technical details which need to be learned in order to deploy the same application on heterogeneous infrastructures burden the simulation scientist with a significant investment of time that would be better spent on the application or its use case itself.
- P3** Even once several resource types have been shown to be compatible with a certain application, choosing the optimal compute resource for a given job in terms of hardware equipment and available software packages remains highly non-straightforward.

The remainder of this paper is structured as follows: In Sect. 2, we introduce the simulation application that serves as a use case in scope of this work. After that, we discuss the conceptual framework, we propose to homogenize the utilization of the heterogeneous infrastructure in Sect. 3, followed by a short introduction to the modeling language *Topology and Orchestration Specification for Cloud Applications* (TOSCA), which we use as a basis for building the models in scope of the framework in Sect. 4. In Sect. 5, we discuss the modeling of the use case application with help of the framework, followed by a discussion of our findings and the limitations of the approach in Sect. 6. Finally, we provide an overview of related work in Sect. 7 and draw our conclusions and give an outlook on future work in Sect. 8.

2 Use Case

As an exemplary use case, the simulation of the flow in porous media with the *Lattice-Boltzmann method* (LBM) was chosen. LBM originates from Boltzmann's kinetic molecular dynamics and may be understood as a discretization in space and time of the velocity-discrete Boltzmann equation. Its main advantages are

the inherent parallelism which leads to great performance on many architectures and easy handling of complex geometries.

Particle filled beds are of great technical importance e.g., in catalysator packings. Thus, the knowledge of the pressure drops in such packings is crucial. Confining walls can have a significant influence on the pressure drop and therefore the pressure drop as a function of the sphere diameter to wall distance ratio is systematically studied. A high number of packings is created using an algorithm and a scale-resolving simulation of the flow in the packing with LBM is performed (Fig. 1). In previous work, measured and simulated pressure drops in slit-type milli-channels with different packings were compared. Very good agreement between measured and simulated pressure drops was achieved [8].

This use case can be seen as a typical parameter study which are common in engineering. The same program is plurally started with different parameters (in this case sphere diameter to wall distance ratio). The computational cost of a single program run is relatively low and the high effort is mainly caused by the multiple execution. The demands on the infrastructure are therefore different from a single large simulation like an aerodynamic simulation with a great number of mesh cells and more complex domain boundaries, where the simulation is distributed over many nodes which must be synchronized after every time step and thus has higher demands on the network connection. A more involved simulation may therefore necessitate the switch to a different infrastructure.

To shield the simulator from the manual adaptation of the provided resources to the different simulation types, we build an infrastructure that helps to provision resources of the correct type and scale transparently.

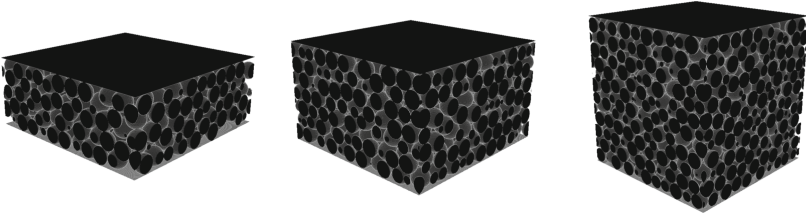


Fig. 1. Exemplary packings of spheres between two plates with a sphere diameter to wall distance ratio of 4, 6 and 10.

3 Model-Driven Resource Provisioning

In our work, we use a formal model of the simulation application topology and its behaviour to automatically provision suitable computing resources. The overall workflow that is implemented by our infrastructure is depicted in Fig. 2. To distinguish the models used on the different layers we orientate on the notations introduced with the *Model Driven Architecture* (MDA) [15], developed by the *Object Management Group* (OMG).

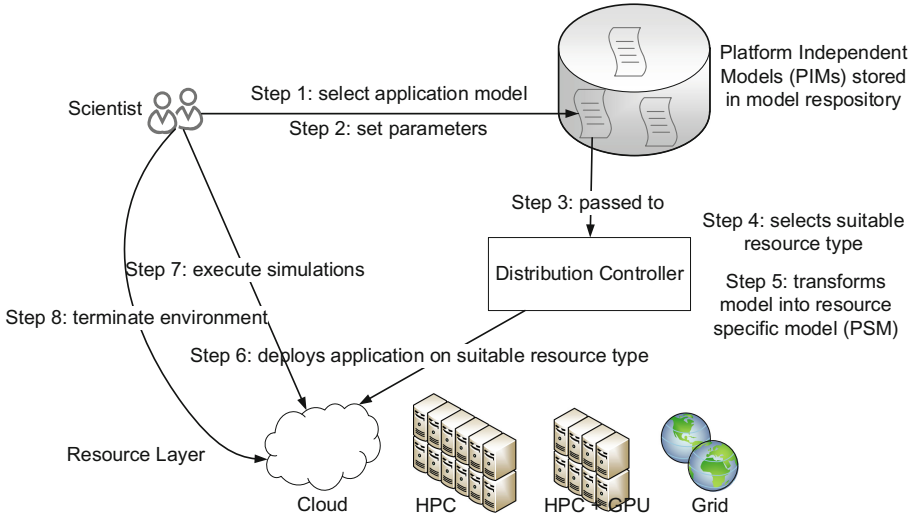


Fig. 2. Workflow for model-driven resource provisioning for simulation applications.

A *Platform Independent Model* (PIM) encodes the structure and behaviour of the simulation application in a target resource independent way and is stored in a model repository. The simulator is then able to select (Step 1) and adapt (Step 2) the existing models from the repository. The selected and instantiated model is then passed to a *Distribution Controller* (Step 3) that evaluates the parameters of the model and selects the suitable target infrastructure accordingly (Step 4) and finally transforms the selected model into a *Platform Specific Model* (PSM) that matches the requirements of the targeted infrastructure (Step 5). In the next step, the resource provisioning and the automated deployment of the simulation application on the targeted resource is triggered (Step 6). After that, the simulator is given access to the provided resource via a *Command Line Interface* (CLI) and can execute his simulations accordingly (Step 7). When all simulation runs are done, the simulator can collect the output data and triggers the cleanup and termination of the provided infrastructure (Step 8).

Different resource-specific formats for defining the simulation application for the different target infrastructures exist. We adopt the *Topology and Orchestration Specification for Cloud Applications* (TOSCA) [12] which is currently developed by a large technical consortium and allows to define the topology and the behaviour of cloud applications in a provider-agnostic way.

To build a format for describing the PIM, we extend TOSCA to not only be able to capture cloud-specific information, but also the information that is necessary to deploy the application on the other targeted resources, like classical HPC or HPC+GPU setups. We first concentrate on utilizing the *IBM Load Sharing Facility* (LSF) [9], i.e. generating jobscripts ready for submission. For a given simulation application, this entails taking into account parameters that are

fixed upon deployment, for instance the infrastructure model for the compute cluster, as well as parameters that may vary between execution steps like differing mesh resolutions or choices of which data to operate on.

The model-driven approach is also employed to describe the infrastructure on which the simulation application will be deployed and executed. Keeping the application and infrastructure models up to date regarding evolving software versions, for example of the Load Scheduler, is as important as making them available to the simulation scientist in the first place. We therefore intend the repository to be maintained independently for the application and infrastructure models. This way, a PIM of the Load Scheduler is combined with information about a specific cluster setup by its administrator to yield a PSM for the users. The result is combined with the PIM for the simulation application, which can in turn be maintained by its developers. Our approach exemplifies a separation of concerns of the knowledge about the specific compute cluster and about the simulation application from the technical details regarding the transformation of their respective models in the *Distribution Controller*. Consequently, each type of model can be developed by the group of people best equipped to do so, which is a core tenet of MDA.

In summary, the MDA approach addresses the previously stated problems in the following ways:

- S1** Separation of concerns is an integral part of MDA - in the case at hand this approach translates to models concerning the simulation application and those describing the various resource types being designed by the application developers/users and compute resource administrators respectively, each contributing their domain-specific knowledge, instead of both sides being handled by the simulation scientist as outlined in (**P1**).
- S2** If a convention for describing the interface between applications and resource types is defined and adhered to by the designers of both types of models, the time-investment for making a new resource type eligible for automatic deployment has to be made only once instead of once per application, thereby addressing (**P2**).
- S3** Given the model for a specific intended application run and several resource models, programmatically determining the resulting valid combinations can lead to a more well-informed manual decision by the simulation scientist (cf. (**P3**)) and ideally to an automated resource choice.

4 Application Modeling with TOSCA

The *Topology and Orchestration Specification for Cloud Applications* (TOSCA) is a standard which is currently developed by the *Organization for the Advancement of Structured Information Standards* (OASIS). Its goal is to standardize a template format to describe cloud applications in a portable and reusable manner, such that they can be deployed to TOSCA-compliant clouds of different cloud providers. While the targeted resource type for TOSCA are cloud

environments, the defined modeling concepts are not resource type specific and we utilize the concepts defined by TOSCA to model simulation applications in a resource type independent way. According to the specification [12], TOSCA is “a language to describe service components and their relationships using a service topology, and it provides for describing the management procedures that create or modify services using orchestration processes.” Therefore, it is able to describe both the service structure as well as the processes that can be executed on this structure.

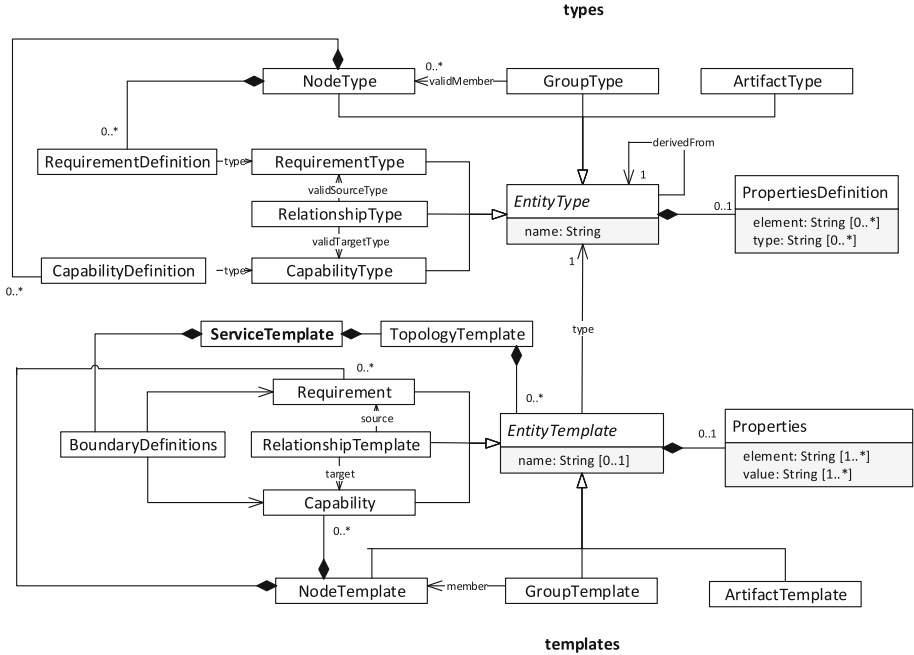


Fig. 3. A simplified subset of the TOSCA metamodel.

A simplified subset of the TOSCA metamodel is depicted in Fig.3. A **ServiceTemplate** captures the structure and also the life cycle operations of the application. For the sake of brevity, we omit the modeling elements that are used to define the life cycle operations, because they are not relevant for the current status of our work. As part of **ServiceTemplates**, **TopologyTemplates** can be defined. **TopologyTemplates** contain **EntityTemplates**, which can be **NodeTemplates** that define e.g., the virtual machines or application components, **RelationshipTemplates** that encode the relationships between the **NodeTemplates**, e.g., that a certain application component is deployed on a certain

virtual machine, or `GroupTemplates`¹ that allow to group a number of `NodeTemplates`, which e.g., should be scaled together. Additionally, TOSCA defines the `EntityTemplates` `Capability`, `Requirement` and `ArtifactTemplate`. `Capabilities` are used to define that a `NodeTemplate` has a certain ability, e.g., providing a container for running applications, and `Requirements` are used to define that a certain `NodeTemplate` requires a certain `Capability` of another `NodeTemplate`. The aforementioned `RelationshipTemplates` are used to connect the `Requirement` of one `NodeTemplate` with a matching `Capability` of another `NodeTemplate`. The `ArtifactTemplate` is used to model all kinds of artifacts, such as source code, or binaries. All `EntityTemplates` can have `Properties`, e.g., an IP address for a virtual machine, and a certain `type` that references a corresponding `EntityType`. The `EntityType` defines the allowed `Properties` through `PropertyDefinitions`, and the allowed `Capabilities` and `Requirements` through `Capability-` and `RequirementDefinitions` respectively. Besides this abstract metamodel, the TOSCA [12] specification defines `normative types` that should be supported by each TOSCA conformant cloud orchestrator. These normative types include e.g., base types for cloud services and virtual machines. More details on the model elements can be found in the TOSCA specifications [12, 13].

5 Evaluation

In the following, we demonstrate how the defined framework can be used to model our use case application from fluid mechanics and provision resources on two different target platforms, namely an *High Performance Computing* (HPC) system and an *Infrastructure as a Service* (IaaS) cloud. Therefore, we present and discuss the models in the provisioning process for the different target resources. To foster comprehensibility, we slightly modified the models and omit some technical details.

5.1 Utilized Tooling

The *Eclipse Modeling Framework* (EMF) defines a common standard for structured data models, which conform to metamodels specified in the *Ecore* format [19]. We automatically converted the *XML Schema Definition* (XSD)-based TOSCA specification to such a metamodel using the EMF tools, and parsed the normative type definitions specified in *YAML Aint Markup Language* (YAML) manually. Once the complete TOSCA metamodel was available in the *Ecore* format, we were able to edit the application and resource models, verify them against the constraints imposed by the respective metamodel, and persist them using the *XML Metadata Interchange* (XMI) format.

¹ At the time of this writing, two versions of the TOSCA standard exist, a formalized version [12] in XML and a simplified rendering in YAML [13]. `GroupTemplates` and `GroupTypes` are currently part of the TOSCA YAML rendering, but not part of the TOSCA XML specification.

With the help of the domain-specific languages provided by the *Eclipse Epsilon* project [20], *Model-to-Model* (M2M) and *Model-to-Text* (M2T) transformations can be performed using various modeling techniques, in particular EMF. For example, we implemented M2M transformations with the *Epsilon Transformation Language* (ETL) in order to obtain the PSM for a concrete setup from the PIM of the respective resource type, modeling an IaaS cloud standard or an HPC batch system. Using the resulting model, the application PIM is transformed to the PSM for deployment in a similar fashion. M2T transformations, on the other hand, are realized using the *Epsilon Generation Language* (EGL), a template-based format that is suitable for amend generic IaaS deployment scripts or HPC jobsheets with the data of the deployment model.

While the metamodels, models and transformations can be implemented and debugged in the Eclipse workbench for development purposes, in order to enable the user to focus on application deployment, we implement a command-line interface which handles model transformations and template rendering as a standalone application.

5.2 Model of the Fluid Mechanics Application

In Fig. 4 the application model for *Adaptive Mesh Renement in Objectoriented C++* (AMROC) [4], the fluid solver framework our simulation use case is based on, is depicted. The software uses the parallelization standard *Message Passing Interface* (MPI).

Type definitions: The central node type `AMROCNType` is equipped with three requirement definitions: `MPIClusterRDef` models the fact that an MPI cluster is necessary to run the software and allows specifying the necessary number of nodes and the required MPI version via the properties of the corresponding capability type `MPIClusterCType`, the requirement definition `PackageRDef` defines the requirement for a specific mpi implementation given by the respective capability type `PackageCType`, and finally `HostRDef` addresses the hardware requirements imposed on the involved compute nodes, namely the amount of memory and CPU cores, which constitute the properties of its respective capability type `HostCType`. In addition, an artifact type `AnsibleRoleAType` enables the application model to provide scripts for the configuration management tool Ansible².

Templates: The requirements of a concrete simulation application based on AMROC are modeled by a corresponding node template `AMROCNTemplate` to which instances of the requirement definitions are assigned, providing values for the properties specified by `MPIClusterCType`, `PackageCType`, and `HostCType`. Finally, `AMROCCode` references the provided configuration management script.

² <https://www.ansible.com/>.

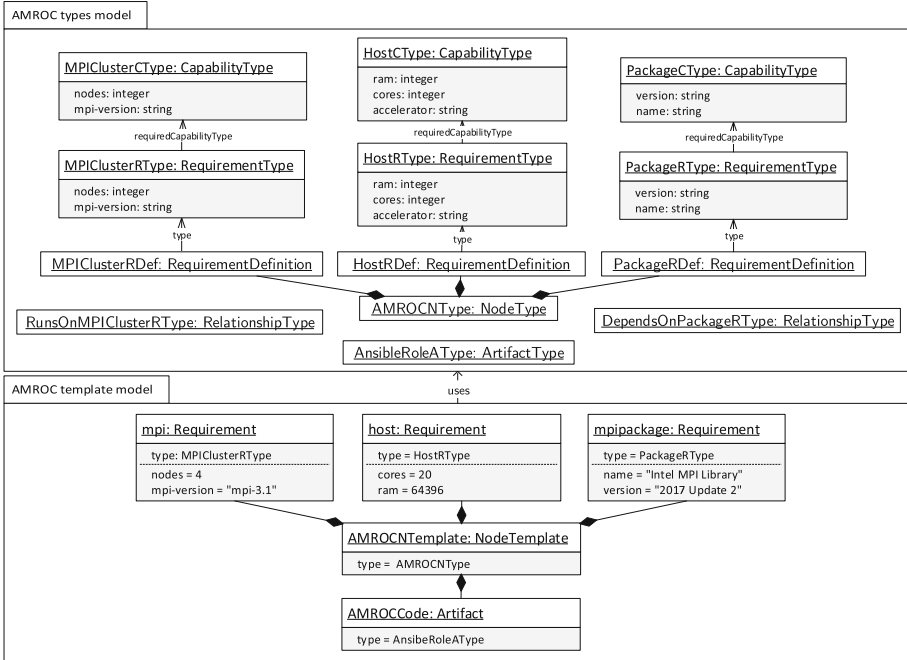


Fig. 4. Platform independent model of the fluid mechanics application.

5.3 Model of Batch Target System

Our model of an HPC cluster using the *Load Sharing Facility* (LSF) as the batch system is shown in Fig. 5. Again, the graphical representation is separated for type and template definitions: While the types section is specific only to the batch system, the templates represent parts of the Scientific Compute Cluster configuration as it is hosted at the *Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen* (GWDG).

Type definitions (LSF cluster): The group type `HPCClusterGType` serves as a root element providing the capability `PackageCDef`, which exposes preinstalled software packages, usually made available in the form of environment modules. These are specified using the properties of the corresponding capability type `PackageCType`. The compute nodes in an LSF cluster, which are modeled by `HPCNodeType`, are organized in a tree-like structure of host groups modeled by group elements `HPCNodeGType`. A capability definition `HostCDef` of type `HostCType` (cf. Fig. 4) is used to define each node’s hardware specifications.

For the available job submission queues of type `HPCQueueGType` further capabilities are generated dynamically: `IncludedHostCDef`, which is of the same capability type used for individual hosts, allows the Distribution Controller to match the application model’s requirements for `HostCType` against

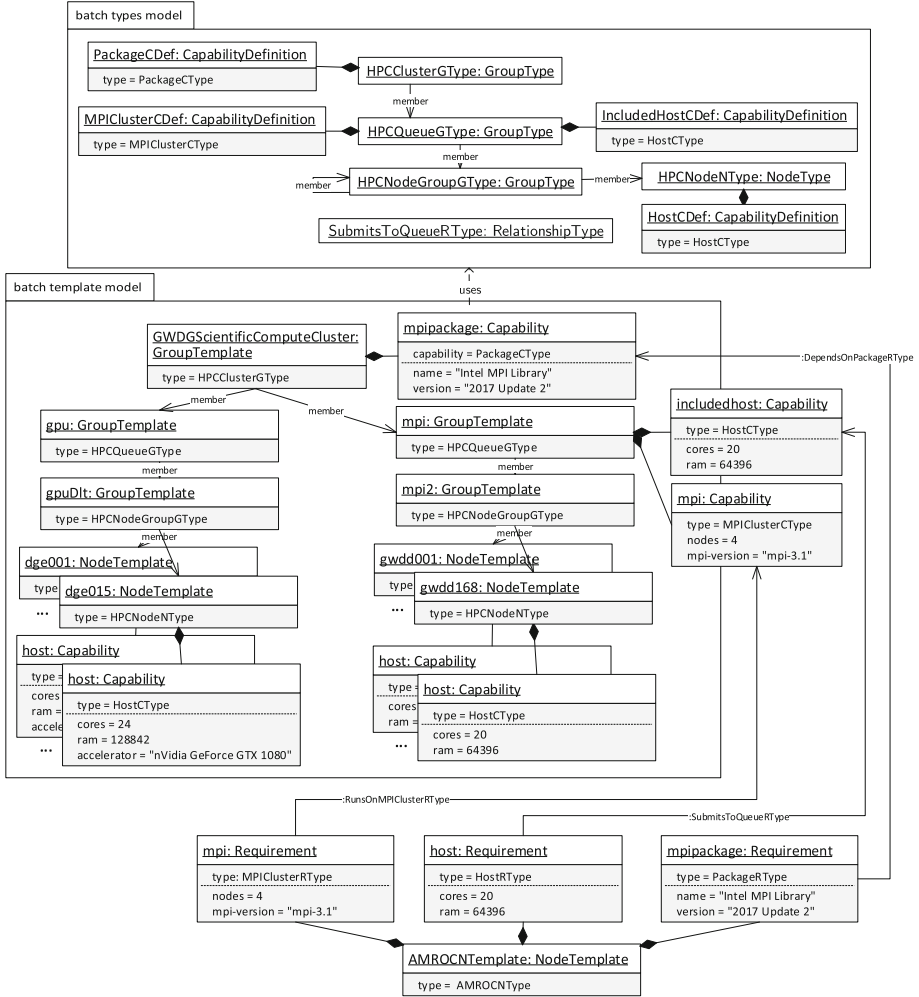


Fig. 5. Provisioning and deployment model for the HPC target resource type.

the configured queues directly. `MPIClusterCDef` marks the queue as suitable for submitting MPI jobs (cf. Fig. 4 for the definition of `MPIClusterCType`).

The relationship type `SubmitsToQueueRType` is used for choosing one of the queues during deployment of the application. Finally, the queue references one or more host groups as members.

Templates (GWDG Scientific Compute Cluster): The central group type `HPCClusterGType` is instantiated as the group template `GWDGScientificComputeCluster`. The capability type `PackageCType` is used to specify the available MPI library version by its instance `mpipackage`.

Two representative examples, the `mpi` and `gpu` queues are given as group instances of `HPCQueueGType`, along with representatively chosen host groups as members of type `HPCNodeGType`. Their members are in turn models of the available compute nodes, specified by a node template of type `HPCNodeNTType` each.

The hardware capabilities of these nodes, namely the number of cores, amount of installed memory and type of eventually installed GPU accelerator are described by capability assignments of type `HostCType`.

AMROC deployment on the batch system: As an example, the model elements relevant for deploying the AMROC application and submitting a job to the `mpi` queue are shown as well: An instance `AMROCNTemplate` of `AMROCNTType` provides concrete values for its associated requirements, which are matched to the globally defined `mpipackage` as well as the queue-specific capabilities `includedhost` and `mpi` via the respectively suitable relationship.

5.4 Model of IaaS Cloud Target System

Figure 6 depicts the model for the IaaS cloud system. One fundamental difference to the batch system is that compute nodes can be created and configured on demand according to simulation application requirements, including the (virtual) hardware, the installed software and the operating system on the compute nodes.

Type definitions (Cloud orchestration): The type definitions depicted in the upper part of the figure orientate on the TOSCA types that are defined and utilized by the TOSCA-compliant cloud orchestrator Cloudify³ for modeling cloud resources of an OpenStack⁴ cloud. The group type `MPIClusterGType` serves as a root element providing the capability `PackageCDef`, which exposes software packages for which installation scripts are available and which can therefore be installed on demand. The node type `ServerNTType` abstracts the notion of a *Virtual Machine* (VM) that can be started in the IaaS cloud. Its properties `flavor` and `image` encode the hardware configuration and the operating system respectively. The node type `VolumeNTType` models an external block-storage device that can be attached to a running VM and the node type `FloatingIPNTType` allows to define a publicly reachable IP address for a VM. Finally, `ApplicationNTType` is a node type used to model an application that should be installed on the deployed virtual machine, in the case of the prototypical implementation via Ansible roles referenced by its property `roles`. The group type `ScalingGroupGType` allows to group templates of the type `ServerNTType` and `ApplicationNTType` to be scaled together. Moreover, it is used to define the virtual hardware configuration of the contained VMs via its capability definition `OfferedHostConfig` of type `HostCType`. The fact that the group is configured as an MPI cluster is captured by defining the capability `MPIClusterCDef` (cf. Fig. 4 for the definitions of both capability types). This capability is also the target of the relationship `DeployedOnScalingGroup` used for application deployment.

³ <http://cloudify.co>.

⁴ <http://www.openstack.org>.

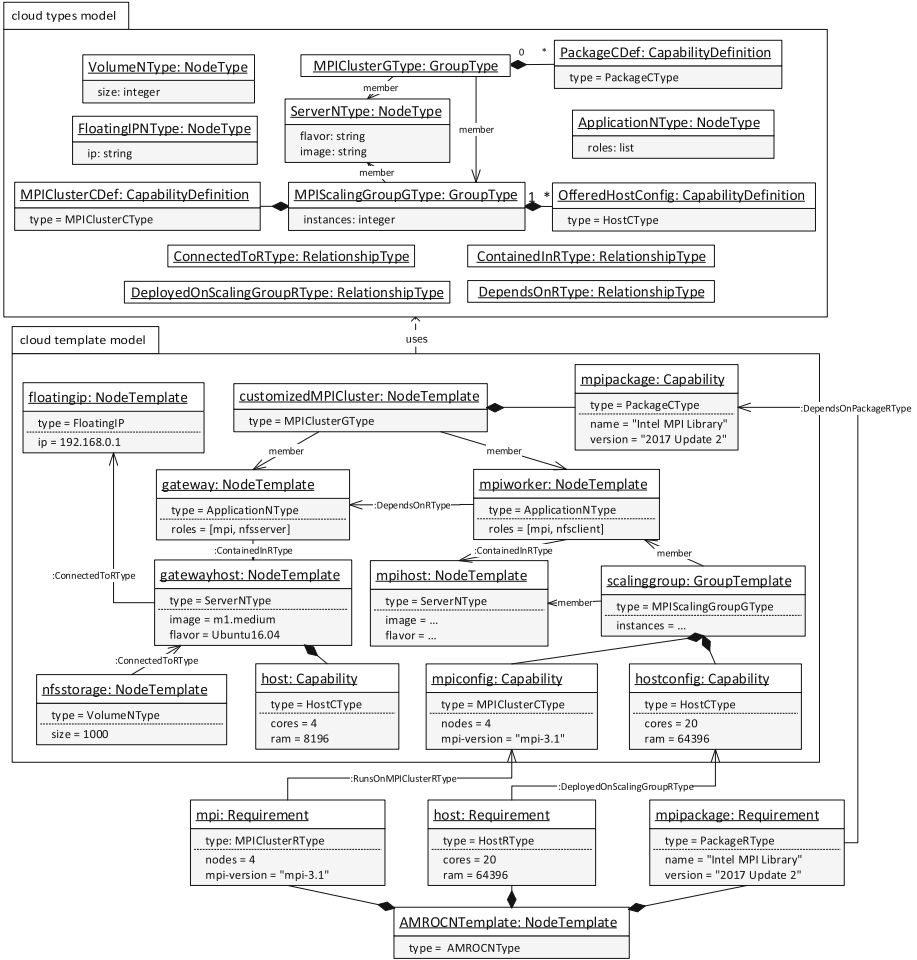


Fig. 6. Provisioning and deployment for the IaaS target resource type.

Three additional relationship types `ConnectedToType`, `ContainedInType` and `DependsOnType` are defined that have the following semantics: `ConnectedToType` expresses a connection between two entity types, `ContainedInType` expresses the fact that an entity type is part of another entity type and the relationship type `DependsOnType` expresses a general dependency between two entity types.

Templates (MPI cluster infrastructure): The group type `MPIClusterGType` is instantiated as the node template `customizedMPICluster`, whose `PackageCType` instance `mpipackage` specifies the MPI version that can be installed.

The node template `gatewayhost` models a VM that is reachable from outside the cloud via an assigned IP-address, modeled by the node template

`floatingip`, while the node template `gateway` models its software configuration. The `gatewayhost` is connected to an *Network File System* (NFS)-storage volume, modeled by the node template `nfsstorage` that is shared among the MPI-enabled worker nodes. The MPI worker nodes are modeled by the node template `mpihost` and its assigned software configurations `mpiworker`. The node templates `mpihost` and `mpiworker` are members of a group template `scalinggroup`: According to the number of compute nodes that are required by the simulation application (cf. Fig. 4), the property `instances` is used to replicate the node templates given by the property `members` and all relationships connecting them. In this way, we are able to scale the number of compute nodes according to the resource demand of the application. The capabilities `mpiconfig` and `hostconfig` assigned to `scalinggroup` provide concrete values for the available MPI version and the virtual hardware specifications respectively.

AMROC deployment in the cloud: As in the aforementioned case of the batch system, the deployment model adds the instance `AMROCNTemplate` of `AMROCNType` providing concrete values for the associated hardware and MPI requirements which are connected to the infrastructure model's capabilities by relationships.

6 Discussion

Our approach applies the MDA solution strategies to the problems outlined above as follows:

- (P1,S1) We were able to construct the application model for AMROC in a resource-independent way, while on the other hand the models describing an IaaS cloud and an HPC cluster using LSF are focused on the logic of the respective resource types only. These examples indicate that modeling anything but the application itself can in principle be removed from the scope of the simulation scientist.
- (P2,S2) Exposing both the requirements of the simulation application and the capabilities of both resource types considered here by referring to a common set of capability types makes application and resource models respectively interchangeable.
- (P3,S3) The existence of a particular Capability within a given resource model as well as the concrete values of its properties enable an automatic choice of a suitable compute resource.

6.1 Limitations

Currently, we focus on the provisioning and deployment of the resources for the simulation application, and especially do not consider the following points:

- We assume the compiled simulation to be able to run on all eligible resource types. However, incorporating the build process as part of the model transformation would enable the same application to be automatically deployed to all hardware architectures it compiled for.

- Platform-dependent code modifications are currently out of scope, therefore only the simulation code as whole is part of the application model. Examples would be the configuration of load-balancing schemes in a parallelization workload, moving parts of a calculation to an accelerator depending on its availability or even generating the simulation code from a model of the underlying algorithm (cf. [14]).
- In the present work, the automated choice of a compute resource is based solely on the data found in a corresponding resource model. Including the results from monitoring previous jobs, in particular in the form of job chains composed of comparable workloads such as parameter studies, would enable us to adjust the parameters of subsequent deployments more precisely.
- The scope of our architecture is currently restricted to handling a single application run. In practice, a combination of multiple jobs, such as the cleaning of input data, choosing the next simulation step based on the data of the previous one and reduction of the results, compose the entire workflow that is ultimately of interest. A model of this high-level view of the simulation needs to include the results of the present work as a combinable unit. Vukojevic-Haupt et al. [22] describe an approach for the provisioning of a cloud-based middleware intended to particularly handle simulation workflows.
- We do not consider data management tasks associated with the application run at this point, such as copying data back and forth between the simulation scientist's system and the compute resource or persisting and sharing the results using independent research data management infrastructures.

7 Related Work

Different projects address specifically the use of models or *Domain-Specific Languages* (DSLs) to target the execution of scientific software in a cloud environment, e.g., Bunch et al. [3] define a DSL for the management of HPC applications in the cloud, and Qashsa et al. [16] utilize TOSCA to model scientific workflows for the cloud. Furthermore research has been conducted to provide scientific applications as services in a cloud environment, e.g., Vukojevic-Haupt et al. [21] define a middleware to deploy scientific applications as services, and Limmer et al. [10] utilize the cloud-standard *Open Cloud Computing Interface* (OCCI) [11] to steer simulation applications in the cloud. Similar to the problem of heterogeneous resource types, is the problem of vendor-specific *Application Programming Interfaces* (APIs) and the service heterogeneity of different cloud providers, because this makes it infeasible to switch between the offerings. Ardagna et al. [1] propose the use of MDA to model cloud applications in a cloud-provider independent fashion, and Quinton et al. [17] provide a platform to select and configure a specific cloud-offering based on models. Further approaches aiming to provide provider-agnostic frameworks for cloud-based applications [5,7] and to improve cloud interoperability by combining existing applications using TOSCA [18] exist, but none of the aforementioned solutions discuss the problem of addressing different types of target resources outside the realm of cloud

providers. The OCCIware project [23] showed that also the cloud standard OCCI can be used to model and manage all kinds of resources, but the standard does not provide the functionality to define requirements and capabilities as needed for our architecture, and the automated mapping to a specific infrastructure type is not addressed.

Flissi et al. [6] developed a model-driven method to deploy distributed systems on Grids. In contrast to our work, they do not consider the automated mapping to a specific resource and also do not face the problem with dynamically adaptable resource target types, such as cloud systems. Ober et al. [14] discuss the use of model-driven engineering techniques for the development of HPC applications and Arkin et al. [2] provide a model-driven method to map algorithms to a certain parallel computing platforms, such as MPI or OpenMP. In contrast to our work, both works directly consider the developed code, whereby we focus on the provisioning of the computing resources.

8 Conclusions and Outlook

We develop an architecture which provides a transparent resource provisioning mechanism for simulation applications for heterogeneous computing infrastructures, which are today's reality and in many data centers. The goal is to shield the simulation scientist from complicated infrastructure internals. In this paper, we presented the initial architecture, which orientates on the MDA and demonstrated its feasibility with the help of a LBM simulation from fluid mechanics.

Future work includes the adaptation of the strict resource requirements modeled on a per-host basis in our approach to a more flexible format that allows the Distribution Controller to split, for example, the same total amount of CPU cores and memory in different ways according to the available hardware. Another possible extension of the architecture consists of logging and analyzing the achieved application performance in order to improve the choices made in following iterations of deploying the same application.

Acknowledgements. We thank the Simulationswissenschaftliches Zentrum Clausthal-Göttingen (SWZ) for financial support.

References

1. Ardagna, D., et al.: MODAClouds: a model-driven approach for the design and execution of applications on multiple clouds. In: 2012 4th International Workshop on Modeling in Software Engineering (MISE), pp. 50–56. IEEE, June 2012. <https://doi.org/10.1109/MISE.2012.6226014>
2. Arkin, E., Tekinerdogan, B., İmre, K.M.: Model-driven approach for supporting the mapping of parallel algorithms to parallel computing platforms. In: Moreira, A., Schätz, B., Gray, J., Vallecillo, A., Clarke, P. (eds.) MODELS 2013. LNCS, vol. 8107, pp. 757–773. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41533-3_46

3. Bunch, C., Chohan, N., Krintz, C., Shams, K.: Neptune: a domain specific language for deploying HPC software on cloud platforms. In: Proceedings of the 2nd International Workshop on Scientific Cloud Computing, pp. 59–68. ScienceCloud 2011. ACM (2011). <https://doi.org/10.1145/1996109.1996120>
4. Deiterding, R.: AMROC - Adaptive Mesh Refinement in Object-Oriented C++ (2017). <http://www.vtf.website/asc/wiki/bin/view/Amroc/WebHome>. Accessed 8 Nov 2017
5. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M.: Building a mosaic of clouds. In: Guarracino, M.R., et al. (eds.) Euro-Par 2010. LNCS, vol. 6586, pp. 571–578. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21878-1_70
6. Flissi, A., Dubus, J., Dolet, N., Merle, P.: Deploying on the grid with deployware. In: 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), pp. 177–184. IEEE (2008). <https://doi.org/10.1109/CCGRID.2008.59>
7. Guillén, J., Miranda, J., Murillo, J.M., Canal, C.: A service-oriented framework for developing cross cloud migratable software. *J. Syst. Softw.* **86**(9), 2294–2308 (2013). <https://doi.org/10.1016/j.jss.2012.12.033>
8. Hofmann, S., Bufe, A., Brenner, G., Turek, T.: Pressure drop study on packings of differently shaped particles in milli-structured channels. *Chem. Eng. Sci.* **155**, 376–385 (2016). <https://doi.org/10.1016/j.ces.2016.08.011>
9. IBM Corporation: Introduction to IBM Platform LSF. https://www.ibm.com/support/knowledgecenter/SSETD4.9.1.2/lsf_foundations/lsf_introduction_to.html. Accessed 8 Nov 2017
10. Limmer, S., Srba, M., Fey, D.: Performance investigation and tuning in the interoperable Cloud4E platform. In: Lopes, L., et al. (eds.) Euro-Par 2014. LNCS, vol. 8806, pp. 85–96. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14313-2_8
11. Nyrén, R., Edmonds, A., Papaspyrou, A., Metsch, T., Parák, B.: Open Cloud Computing Interface - Core, September 2016. <http://ogf.org/documents/GFD.221.pdf>
12. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) 1.0, November 2013. <http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html>. Accessed 8 Nov 2017
13. OASIS: TOSCA Simple Profile in YAML Version 1.1, August 2016. <http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/csprd01/TOSCA-Simple-Profile-YAML-v1.1-csprd01.html>. Accessed 8 Nov 2017
14. Ober, I., Palyart, M., Bruel, J.M., Lugato, D.: On the use of models for high-performance scientific computing applications: an experience report. *Softw. Syst. Model.* **17**(1), 319–342 (2018). <https://doi.org/10.1007/s10270-016-0518-0>
15. Object Management Group: Model Driven Architecture. <http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf>. Accessed 8 Nov 2017
16. Qasha, R., Cala, J., Watson, P.: Towards automated workflow deployment in the cloud using TOSCA. In: 2015 IEEE 8th International Conference on Cloud Computing, pp. 1037–1040. IEEE (2015). <https://doi.org/10.1109/CLOUD.2015.146>
17. Quinton, C., Romero, D., Duchien, L.: SALOON: a platform for selecting and configuring cloud environments. *Softw. Pract. Experience* **46**(1), 55–78 (2016). <https://doi.org/10.1002/spe.2311>
18. Soldani, J., Binz, T., Breitenbücher, U., Leymann, F., Brogi, A.: ToscaMart: a method for adapting and reusing cloud applications. *J. Syst. Softw.* **113**, 395–406 (2016). <https://doi.org/10.1016/j.jss.2015.12.025>

19. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional, Amsterdam (2009)
20. The Eclipse Foundation: Epsilon. <https://eclipse.org/epsilon/>. Accessed 8 Nov 2017
21. Vukojevic-Haupt, K., Haupt, F., Leymann, F.: On-demand provisioning of workflow middleware and services into the cloud: an overview. *Computing* **99**(2), 147–162 (2017). <https://doi.org/10.1007/s00607-016-0521-x>
22. Vukojevic-Haupt, K., Haupt, F., Leymann, F., Reinfurt, L.: Bootstrapping complex workflow middleware systems into the cloud. In: 2015 IEEE 11th International Conference on e-Science, pp. 126–135. IEEE (2015). <https://doi.org/10.1109/eScience.2015.69>
23. Zalila, F., Challita, S., Merle, P.: A model-driven tool chain for OCCI. In: Panetto, H., et al. (eds.) *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*, pp. 389–409. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_26



Extending the CMMI Engineering Process Areas for Simulation Systems Engineering

Somaye Mahmoodi¹, Umut Durak^{1,2(✉)}, Torsten Gerlach²,
Sven Hartmann¹, and Andrea D'Ambrogio³

¹ Department of Informatics, Clausthal University of Technology,
Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany
{somaye.mahmoodi, umut.durak, sven.hartmann}@tu-clausthal.de

² Institute of Flight Systems, German Aerospace Center (DLR),
Lilienthalplatz 7, 38108 Braunschweig, Germany
{umut.durak, torsten.gerlach}@dlr.de

³ Department of Enterprise Engineering, University of Rome Tor Vergata,
Via del Politecnico, 1, 00133 Rome, Italy
dambro@uniroma2.it

Abstract. Today's companies in high-tech industries develop products of high complexity which consist of complicated subsystems with many heterogeneous components integrated together. As the system complexity increases, it becomes increasingly more challenging to manage the tedious development process. The Capability Maturity Model Integration (CMMI) was proposed as a general framework for process management and improvement which judges the maturity of a process. Simulations have long been regarded as complex and integrated systems. Simulation system engineering manages the total simulation system's life-cycle process. The adaptation of the CMMI for simulation life-cycle processes is envisioned as a domain specific solution for simulation process management and improvement. This article investigates the opportunities of extending the CMMI engineering process area with emphasis in simulation system engineering, having its roots from IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP).

Keywords: Distributed Simulation Engineering and Execution Process (DSEEP) · Capability Maturity Model Integration (CMMI) Simulation process improvement

1 Introduction

The Capability Maturity Model Integration (CMMI) is a framework which provides essential practices for process improvement [1]. The CMMI *Maturity Levels* and *Capability Levels* describe how much the product development process in

one organization is mature and capable. The CMMI currently supports three areas of interest; namely development, services and acquisition. The CMMI for development includes four categories: process management, project management, engineering and support. Each category contains a number of process areas, and each process area consists of number of specific goals (SG). Furthermore, each specific goal has several specific practices (SP).

Organizations intend to utilize the CMMI for managing and improving their processes. The CMMI assessments, also referred to as appraisals, provide feedback to the organization about their compliance and measure of effectiveness in particular process areas. Goldenson and Gibson indicate that companies which make use of CMMI process management and improvement achieve lower costs, a better schedule, high quality products and eventual high customer satisfaction [2].

The simulation life cycle process has long been an area of interest. Balci published his modeling and simulation life cycle process in 1990 [3]. Later, IEEE Std 1516.3-2003, IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP) was developed for federation development using High Level Architecture (HLA) for distributed simulation [4]. FEDEP is generalized by Simulation Interoperability Standards Organization (SISO), which leads to the IEEE Std 1730-2010 IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP). The DSEEP is introduced as an engineering process for all types of distributed simulation development and execution [5].

Simulation and systems engineering has long been in a synergistic relationship [6]. Systems engineering cannot be considered without its relationship with simulation. The contribution of systems engineering to simulation on the other hand empowers simulation, especially for large and complex simulation systems which lead to simulation systems engineering. The harmonization of the system and simulation life-cycle processes is currently an active research topic [7]. Accordingly, investigating the applicability of a process improvement standard that has its roots in the systems engineering community in the simulation domain is legitimate.

In 2002, Richey proposed the utilization of CMMI in modeling and the simulation community [8]. While he argued that conforming such an international standard will result in cost-effective, easy and cohesive simulation engineering, he also remarked that it will be beneficial to have discipline-specific amplifications for incorporating the industry standards and best practices for building modeling and simulation products.

There have been various other efforts that target adapting the CMMI for particular domains. Withalm et al. proposed the CMMI adaptation to support skill assessment and development in tourism [9]. Chen and colleagues proposed the teaching capability maturity model for higher education [10]. In [11], Williams proposed the application of the CMM in the medical domain. Neuhaser discussed adapting a maturity model for online course design in [12]. Bofinger et al. extended the CMMI for safety-related systems [13].

After Richey's proposal [8] about CMMI in modeling and simulation, Fujimoto et al. [14] also recently supported the necessity of CMMI for modeling and simulation. This article investigates the feasibility of tailoring the CMMI engineering process areas for simulation system engineering. The effort aims at adapting the CMMI engineering process areas for simulation life-cycle processes in order to optimize its full potential in the simulation domain to reach a higher level of process coverage for simulation system engineering.

The paper first introduces the assessment of the DSEEP against the CMMI Engineering process areas. Then, in Sect. 3, we propose an extension to the CMMI to cover the DSEEP requirements and objectives. Later, Sect. 4 describes the method that was used to evaluate the proposed CMMI extension. Lastly, we conclude the paper with a discussion of the crucial points of this study as well as future work.

2 Assessment of CMMI Against DSEEP

The CMMI Engineering process areas are: requirements development (RD), technical solution (TS), product integration (PI), verification (VER), and validation (VAL). As presented in Table 1, each of these process areas has their specific goals and specific practices. As an example, selecting product component solutions, developing the design and implementing the product design are the specific goals of the TS process area and specific practices of SG1 in this process area are: SP 1.1 - Develop Alternative Solutions and Selection Criteria, and SP 1.2 - Select Product Component Solutions.

The DSEEP is a framework for simulation system engineering processes which consists of seven steps, with each step having its own set of activities. Each activity has inputs, outputs and recommended tasks [5]. As listed in Table 2, the seven steps of the DSEEP are: defining simulation environment objectives, performing conceptual analysis, designing the simulation environment, integrate and testing the simulation environment, executing the simulation, analyzing data, and evaluating results. As an example, the activities for Develop the Simulation Environment are developing the simulation data exchange model, establishing the simulation environment agreements, implementing member application designs, and implementing the simulation environment infrastructure. One of the recommended tasks for the Develop the Simulation Environment step is identifying reliable data sources for simulation environment databases and member application(s).

It is important to first analyze the coverage of the CMMI in simulation system engineering processes. The analysis comprises the comparison of the CMMI Engineering Process Areas and the seven steps of the DSEEP. We sought all the activities and their recommended tasks of the DSEEP within corresponding CMMI Engineering specific goals. We have seen that the general CMMI framework fails to cover some of the domain-specific simulation engineering process requirements. In order to exemplify the shortcoming of the CMMI against the DSEEP, the DSEEP Step 4 - Develop Simulation Environment will be further

Table 1. CMMI engineering process areas, specific goals, and specific practices.

| RD | TS | VER | PI | VAL |
|---|---|---|--|--|
| <p>SG 1- Develop Customer Requirements</p> <ul style="list-style-type: none"> ● SP 1.1- Elicit Needs ● SP 1.2- Develop the Customer Requirements | <p>SG 1- Select Product Component Solutions</p> <ul style="list-style-type: none"> ● SP 1.1- Develop Alternative Solutions and Selection Criteria ● SP 1.2- Select Product Component Solutions | <p>SG 1- Prepare for Verification</p> <ul style="list-style-type: none"> ● SP 1.1- Select Work Products for Verification ● SP 1.2- Establish the Verification Environment ● SP 1.3- Establish Verification Procedures and Criteria | <p>SG 1- Prepare for Product Integration</p> <ul style="list-style-type: none"> ● SP 1.1- Determine Integration Sequence ● SP 1.2- Establish the Product Integration Environment ● SP 1.3- Establish Product Integration Procedures and Criteria | <p>SG 1- Prepare for Validation</p> <ul style="list-style-type: none"> ● SP 1.1- Select Products for Validation ● SP 1.2- Establish the Validation Environment ● SP 1.3- Establish Validation Procedures and Criteria |
| <p>SG 2- Develop Product Requirements</p> <ul style="list-style-type: none"> ● SP 2.1- Establish Product and Product Component Requirements ● SP 2.2- Allocate Product Component Requirements ● SP 2.3- Identify Interface Requirements | <p>SG 2- Develop the Design</p> <ul style="list-style-type: none"> ● SP 2.1- Design the Product or Product Component ● SP 2.2- Establish a Technical Data Package ● SP 2.3- Design Interfaces Using Criteria ● SP 2.4- Perform Make, Buy, or Reuse Analyses | <p>SG 2- Perform Peer Reviews</p> <ul style="list-style-type: none"> ● SP 2.1- Prepare for Peer Reviews ● SP 2.2- Conduct Peer Reviews ● SP 2.3- Analyze Peer Review Data | <p>SG 2- Ensure Interface Compatibility</p> <ul style="list-style-type: none"> ● SP 2.1- Review Interface Descriptions for Completeness ● SP 2.2- Manage Interfaces | <p>SG 2- Validate Product or Product Components</p> <ul style="list-style-type: none"> ● SP 2.1- Perform Validation ● SP 2.2- Analyze Validation Results |
| <p>SG 3- Analyze and Validate Requirements</p> <ul style="list-style-type: none"> ● SP 3.1- Establish Operational Concepts and Scenarios ● SP 3.2- Establish a Definition of Required Functionality ● SP 3.3- Analyze Requirements ● SP 3.4- Analyze Requirements to Achieve Balance ● SP 3.5- Validate Requirements | <p>SG 3- Implement the Product Design</p> <ul style="list-style-type: none"> ● SP 3.1- Implement the Design ● SP 3.2- Develop Product Support Documentation | <p>SG 3- Verify Selected Work Products</p> <ul style="list-style-type: none"> ● SP 3.1- Perform Verification ● SP 3.2- Analyze Verification Results | <p>SG 3- Assemble Product Components and Deliver the Product</p> <ul style="list-style-type: none"> ● SP 3.1- Confirm Readiness of Product Components for Integration ● SP 3.2- Assemble Product Components ● SP 3.3- Evaluate Assembled Product Components ● SP 3.4- Package and Deliver the Product or Product Component | |

Table 2. DSEEP seven steps and activities - adapted from [5].

| Step | 1-Define simulation environment objectives | 2-Perform conceptual analysis | 3-Design simulation environment | 4-Develop simulation environment | 5-Integrate and test simulation environment | 6-Execute simulation | 7-Analyze data and evaluate results |
|------------|--|---|---|--|--|--|---|
| Activities | <ol style="list-style-type: none"> 1. Identify user/sponsor needs 2. Develop objectives 3. Conduct initial planning | <ol style="list-style-type: none"> 1. Develop scenario model 2. Develop conceptual environment requirements | <ol style="list-style-type: none"> 1. Select member applications 2. Design simulation environment 3. Prepare detailed plan | <ol style="list-style-type: none"> 1. Develop simulation data exchange model 2. Establish simulation environment agreements 3. Implement member application designs 4. Implement simulation environment infrastructure | <ol style="list-style-type: none"> 1. Plan execution environment 2. Integrate simulation environment 3. Test simulation environment | <ol style="list-style-type: none"> 1. Execute simulation 2. Prepare simulation environment outputs | <ol style="list-style-type: none"> 1. Analyze data 2. Evaluate and feedback results |

Table 3. Activities and recommended tasks for DSEEP Step 4- Develop Simulation Environment.

| DSEEP Step 4- Develop Simulation Environment | Recommended Tasks |
|---|--|
| Activity 1- Develop Simulation Data Exchange Model | <ul style="list-style-type: none"> (a) Select an approach for Simulation Data Exchange Model (SDEM) development (b) Determine appropriate SDEMs (c) Looking for relevant SDEM elements in dictionaries (d) Develop and document the SDEM (e) Verify that the SDEM supports the conceptual model |
| Activity 2- Establish Simulation Environment Agreement | <ul style="list-style-type: none"> (a) Decide the behavior of all objects (b) Identify member applications software modifications were not previously identified (c) Decide common databases and algorithms (d) Identify reliable data sources for simulation environment databases and member application (e) Decide managing the time in the simulation environment (f) Create synchronization points for initialization of simulation environment and procedures (g) Decide save and restore strategy for the simulation environment (h) Decide data distribution across the simulation environment (i) Convert scenario description to an executable scenario (j) Establish security procedures according to security agreements |
| Activity 3- Implement Member Application Designs | <ul style="list-style-type: none"> (a) Implement member application modifications (b) Implement interfaces modifications or extensions of all member applications (c) Develop a required new interface for member applications (d) Implement design of required new member applications (e) Implement supporting databases |
| Activity 4- Implement Simulation Environment Infrastructure | <ul style="list-style-type: none"> (a) Confirm availability of basic facility services like electric power, air conditioning, etc. (b) Confirm availability of required integration/test hardware or software (c) Perform required functions for system administrations for example create user accounts (d) Install and configure required hardware (e) Install and configure required software (f) Test infrastructure (g) Confirm infrastructure adherence to the security plan |

discussed. The DSEEP [5] defines the main activities and recommended tasks for this step, as depicted in Table 3.

Activity 1 - Develop Simulation Data Exchange Model corresponds to the interface design practices that are specific to simulation engineering. It is necessary to specify how member applications interact with each other. The simulation data exchange model must be created in such a way that it is in agreement to be able to consistently describe runtime interactions among member applications. The interfaces are captured in SDEM and the execution and data exchange policy is then documented in the simulation environment agreements. According to Activity 2 - Establish Simulation Environment Agreement, operating agreements should be addressed as well. The CMMI does not cover these two activities. However, Activity 1 should be added in CMMI, TS process area, SG 2 - Develop the Design, SP 2.3 - Design Interfaces Using Criteria. Therefore, CMMI SP 2.3 - Design Interfaces Using Criteria needs to be altered for Activity 1 - Develop Simulation Data Exchange Model. Activity 2 needs to be added in CMMI, RD process area, SG 2 - Develop Product Requirements, SP 2.1 - Establish Product and Product Component Requirements. Figures 1 and 2 depict the adaptation requirements of the CMMI for these two activities.

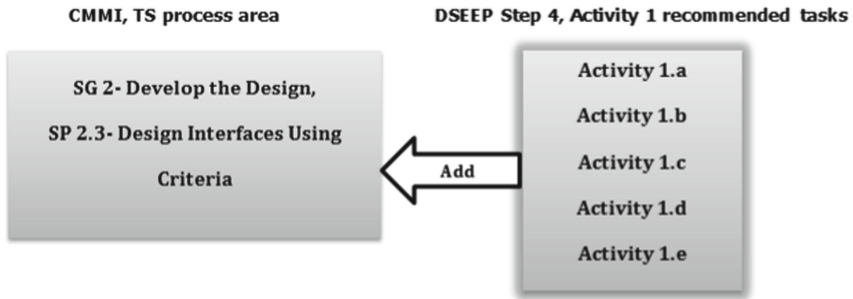


Fig. 1. The DSEEP activity 1 added to the CMMI SP2.3

CMMI, TS process area, SG 3 - Implement the Product Design, SP 3.1 - Implement the Design is responsible for designing the product. Activity 3 - Implement Member Application Designs corresponds to software coding, while Activity 4 - Implement Simulation Environment Infrastructure corresponds to construction of the facilities. As illustrated in Fig. 3, CMMI SP 3.1 can be extended with Activity 3 by adding the recommended tasks 3.b, 3.c, 3.d, and 3.e. There are corresponding clauses in the CMMI for all parts of the Activity 4 - Implement Simulation Environment Infrastructure, with exception to the recommended tasks (c) and (e) which can be added in CMMI, TS process area, SG 3 - Implement the Product Design, SP 3.1 - Implement the Design. Figure 4 depicts the CMMI adaptation requirements for this activity. It is important to

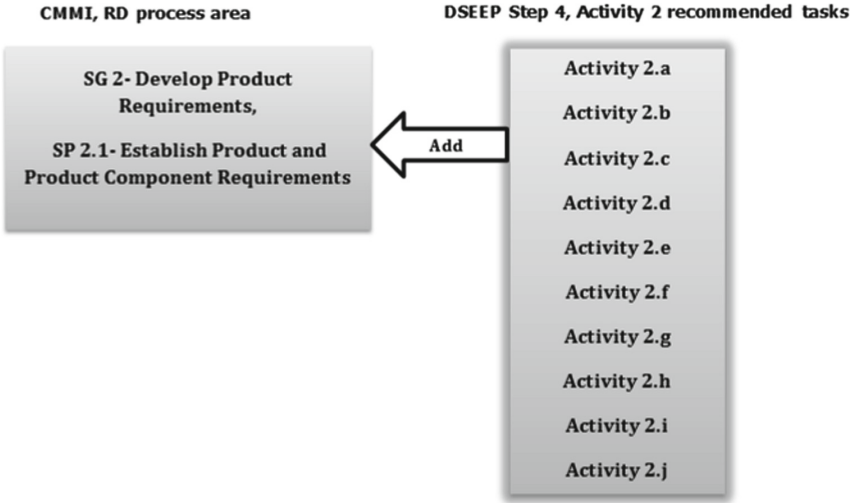


Fig. 2. The DSEEP activity 2 added to the CMMI SP2.1

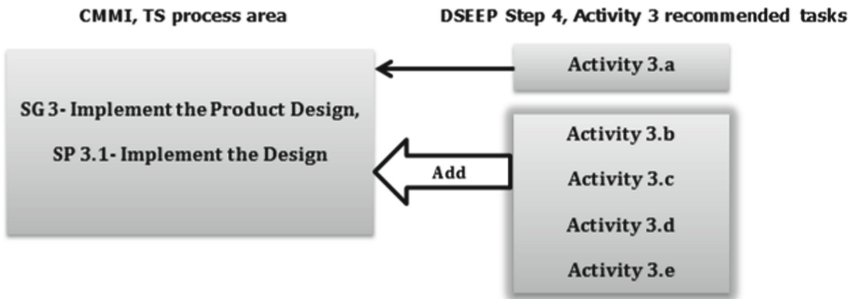


Fig. 3. Adaptation requirements for CMMI SP3.1

also mention that in the TS process area, SG 3, SP 3.1, Sub practice 2 and 4 are:

Sub Practice 2 - Adhere to applicable standards and criteria.

Sub Practice 4 - Perform unit testing of the product component as appropriate.

3 Extending the CMMI for Simulation Systems Engineering

The result of the assessment leads us to extend the CMMI for simulation systems engineering. All DSEEP activities not covered by the CMMI Engineering process areas are added into the specified segments of the standard CMMI which results in a tailored CMMI for simulation. Thereby, the CMMI is extended

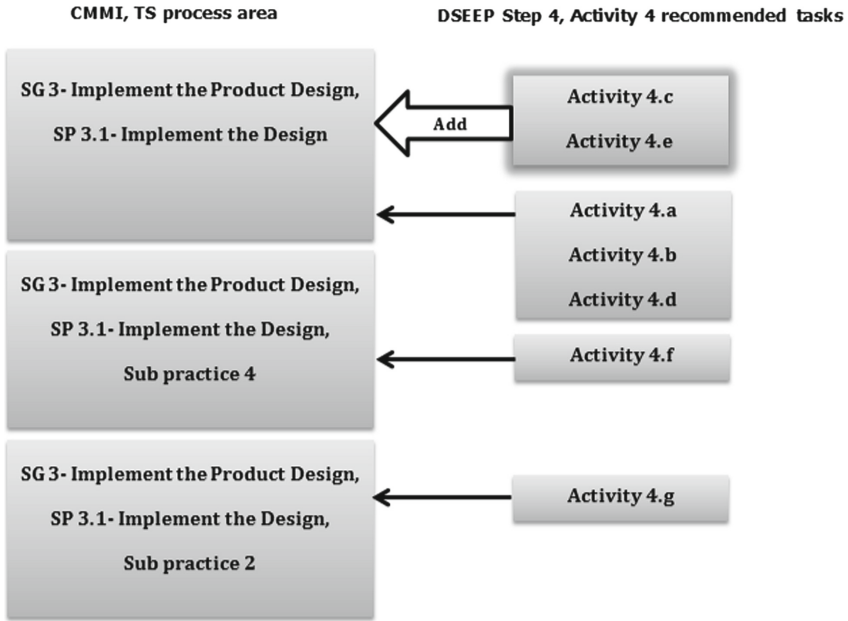


Fig. 4. Adaptation requirements for CMMI SP3.1

for more comprehensive coverage of the simulation engineering process requirements. Tables 4 and 5 present the extensions for the example case discussed in the previous assessment section. They present the standard CMMI Technical Solution and Requirements Development process areas on the left side, and their extension(s) on the right side. As demonstrated in Table 4, extensions to the Technical Solution process area are in SP 2.3 - Design Interfaces Using Criteria and in SP 3.1 - Implement the Design. In Table 5, the extensions to the Requirements Development process area are only in SP 2.1 - Establish Product and Product Component Requirements.

4 Evaluation of Proposed Extension

For the evaluation process, a questionnaire is first prepared based on the tailored CMMI. This questionnaire is derived from rephrasing the Specific Goals, the Specific practices, and DSEEP activities not covered by CMMI into questions. Thereby, it reveals the advantages of a specialized, tailored CMMI in contrast to the standard CMMI model. In this questionnaire, as shown in Fig. 5, the questions of DSEEP activities that are not covered by CMMI are featured in boldface font.

The questionnaire that is used for the assessment consists of 146 questions, 50 of which are based on DSEEP activities not covered by the standard CMMI. The questions specific to the DSEEP enabled us to enrich our assessment and

Table 4. Extensions to the CMMI Technical Solution area for DSEEP Develop Simulation Environment step.

| CMMI- Technical Solution process area (default) | CMMI- Technical Solution process area (enhancement) |
|--|--|
| SG 1- Select Product Component Solutions <ul style="list-style-type: none"> ● SP 1.1- Develop Alternative Solutions and Selection Criteria ● SP 1.2- Select Product Component Solutions | SG 1- Select Product Component Solutions <ul style="list-style-type: none"> ● SP 1.1- Develop Alternative Solutions and Selection Criteria ● SP 1.2- Select Product Component Solutions |
| SG 2- Develop the Design <ul style="list-style-type: none"> ● SP 2.1- Design the Product or Product Component ● SP 2.2- Establish a Technical Data Package ● SP 2.3- Design Interfaces Using Criteria ● SP 2.4- Perform Make, Buy, or Reuse Analyses | SG 2- Develop the Design <ul style="list-style-type: none"> ● SP 2.1- Design the Product or Product Component ● SP 2.2- Establish a Technical Data Package ● SP 2.3- Design Interfaces Using Criteria <ul style="list-style-type: none"> – Select an approach for SDEM development – Determine appropriate SDEMs – Looking for relevant SDEM elements in dictionaries – Develop and document SDEM – Verify that the SDEM supports the conceptual model ● SP 2.4- Perform Make, Buy, or Reuse Analyses |
| SG 3- Implement the Product Design <ul style="list-style-type: none"> ● SP 3.1- Implement the Design ● SP 3.2- Develop Product Support Documentation | SG 3- Implement the Product Design <ul style="list-style-type: none"> ● SP 3.1- Implement the Design <ul style="list-style-type: none"> – Implement interfaces modifications or extensions of all member applications – Develop a required new interface for member applications – Implement design of required new member applications – Implement supporting databases – Install and configure required software – Perform required functions for system administrations for example create user accounts ● SP 3.2- Develop Product Support Documentation |

provide better insight for the improvement of the simulation systems engineering processes. For this reason, the tailored CMMI provides superior process coverage and guidance for a coherent and a complete set of processes.

Table 5. Extensions to the CMMI Requirements Development area for DSEEP Develop Simulation Environment step.

| CMMI- Requirements Development process area (default) | CMMI- Requirements Development process area (enhancement) |
|--|---|
| SG 1- Develop Customer Requirements <ul style="list-style-type: none"> ● SP 1.1- Elicit Needs ● SP 1.2- Develop the Customer Requirements | SG 1- Develop Customer Requirements <ul style="list-style-type: none"> ● SP 1.1- Elicit Needs ● SP 1.2- Develop the Customer Requirements |
| SG 2- Develop Product Requirements <ul style="list-style-type: none"> ● SP 2.1- Establish Product and Product Component Requirements ● SP 2.2- Allocate Product Component Requirements ● SP 2.3- Identify Interface Requirements | SG 2- Develop Product Requirements <ul style="list-style-type: none"> ● SP 2.1- Establish Product and Product Component Requirements <ul style="list-style-type: none"> – Decide the behavior of all objects – Identify member applications software modifications were not previously identified – Decide common databases and algorithms – Diagnose reliable data sources for simulation environment databases and member application – Decide managing the time in the simulation environment – Create synchronization points for initialization of simulation environment and procedures – Decide save and restore strategy for the simulation environment – Decide data distribution across the simulation environment – Convert scenario description to an executable scenario – Establish security procedures according to security agreements ● SP 2.2- Allocate Product Component Requirements ● SP 2.3- Identify Interface Requirements |
| SG 3- Analyze and Validate Requirements <ul style="list-style-type: none"> ● SP 3.1- Establish Operational Concepts and Scenarios ● SP 3.2- Establish a Definition of Required Functionality ● SP 3.3- Analyze Requirements ● SP 3.4- Analyze Requirements to Achieve Balance ● SP 3.5- Validate Requirements | SG 3- Analyze and Validate Requirements <ul style="list-style-type: none"> ● SP 3.1- Establish Operational Concepts and Scenarios ● SP 3.2- Establish a Definition of Required Functionality ● SP 3.3- Analyze Requirements ● SP 3.4- Analyze Requirements to Achieve Balance ● SP 3.5- Validate Requirements |

Requirements development

| No. | Questions | 100% Yes | 50% Yes & No | 0% No |
|-----|---|-------------|-----------------|----------|
| 1. | Do you extract stakeholder's needs, expectations and constraints? | X | | |
| 2. | Do you identify available resources? | X | | |
| 3. | Do you prioritize the product objectives? | X | | |
| 4. | Do you make metrics for each objective? | | X | |
| 5. | Do you plan development and execution of the product? | X | | |
| 6. | Do you make verification and validation plan? | | X | |
| 7. | Do you make configuration management plan? | | X | |
| 8. | Do you make security plan? | | X | |
| 9. | Do you make integration plan? | | X | |
| 10. | Do you make data management plan? | | X | |
| 11. | Do you make plan for tools to support the initial plan? | | X | |

Fig. 5. Part of questionnaire made from Tailored CMMI for simulation systems engineering

The questionnaire is further utilized to conduct an assessment of simulation engineering processes that are practiced for the development, employment and maintenance of the Air Vehicle Simulator (AVES) in German Aerospace Center (DLR). AVES, with its two interchangeable cockpits; one for rotorcraft and the other for airplanes; is one of the largest research flight simulator facilities in Europe [15]. Both simulation cockpits are equipped with replicas of the real cockpit devices. Controls are simulated using reconfigurable active control loading systems. The real cockpit environment is supported by large operator cabins to control the simulation, observe the simulator experiments or software development, integration and testing. Both cockpits can be operated on motion and fixed-base platforms according to the particular need(s). Parallel to its physical architecture, AVES possesses two simulations environments; one for rotorcraft and the other for airplanes. Each environment has more than 40 member

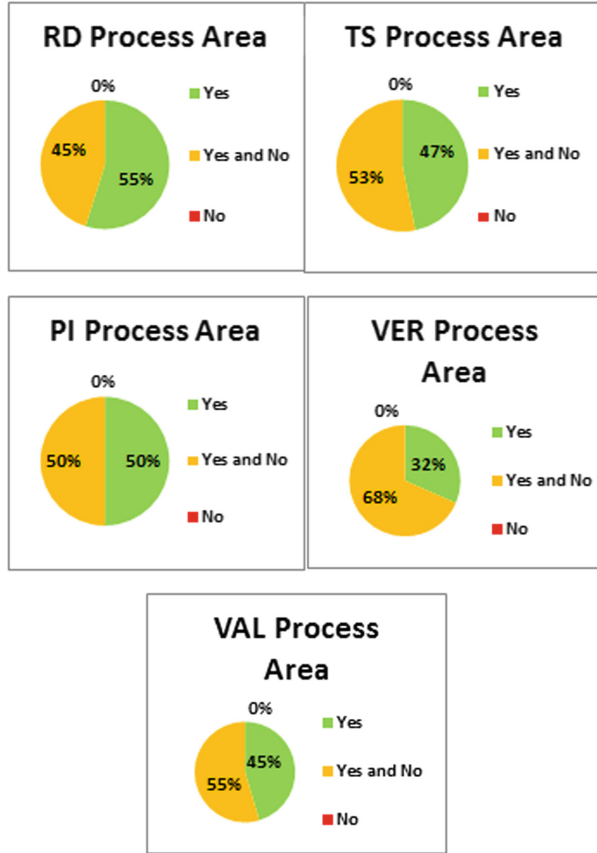


Fig. 6. Tailored CMMI assessment in DLR for each process area (Color figure online)

applications [16]. The number of researchers actively developing, maintaining and conducting experiments in AVES can go up to a couple of dozen.

A contextualized questionnaire is featured in Fig. 6 in order to demonstrate an excerpt from the results of this process assessment. Figure 6 displays this further with a pie-chart for each process area. They illustrate what percentage of the practiced processes are in accordance with the tailored CMMI. The green-colored portion of the charts represents the percentage of the practiced processes that are in full accordance with the tailored CMMI. The orange-colored portion of the charts indicates the areas where further process improvement possibilities exist in order to fulfill compliance.

5 Conclusion

It is evident that as simulation systems become larger and more complex, process improvement is becoming crucial for simulation industry to achieve success.

In this study, we investigated the applicability of the industry standard Capability Maturity Model Integration (CMMI) that provides a well-applied general process improvement framework to simulation systems engineering. We then conducted an assessment of the CMMI Engineering process area against the IEEE recommended practices for Distributed Simulation Engineering and Execution Process (DSEEP). Its results indicated that the standard CMMI Engineering process areas are not able to fully cover the recommended simulation engineering process in DSEEP. Accordingly, we identified the gaps and proposed extensions to the CMMI Engineering process areas. Thereafter we evaluated the proposed extensions by conducting a process assessment based on the tailored CMMI. The proposed extensions enriched the assessment by providing superior coverage.

Future work includes the dissemination of the proposed extension in the simulation community through the Simulation Interoperability Standards Organization's DSEEP Product Support Group as well as collecting feedbacks and contributions. This study may be used as a baseline for preparing a guideline for using the CMMI as a simulation process improvement framework in the simulation industry.

References

1. CMMI Product Team: CMMI for development, version 1.3 (2010)
2. Goldenson, D., Gibson, D.L.: Demonstrating the impact and benefits of CMMI: an update and preliminary results (2003)
3. Balci, O.: Guidelines for successful simulation studies. In: Proceedings of the 22nd Conference on Winter Simulation. IEEE Press (1990)
4. IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP): IEEE Std 1516.3-2003 (2003)
5. IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP): IEEE Std 1730-2010 (2011)
6. Durak, U., Ören, T.: Towards an ontology for simulation systems engineering. In: Proceedings of the 49th Annual Simulation Symposium. Society for Computer Simulation International, p. 13 (2016)
7. D'Ambrogio, A., Durak, U.: Setting systems and simulation life cycle processes side by side. In: 2016 IEEE International Symposium on Systems Engineering (ISSE), pp. 1–7. IEEE (2016)
8. Richey, F.: Modeling and simulation CMMI: a conceptual view. *CrossTalk J. Def. Softw. Eng.* **15**(3), 29–30 (2002)
9. Withalm, J., Wölfel, W., Duin, H., Peters, M.: Combining CMMI with serious gaming and e-learning to support skill assessment and development in tourism. In: International Conference on Interactive Computer Aided Learning (ICL2008). University Press, Kassel (2008)
10. Chen, C., Kuo, C., Chen, P.: The teaching capability maturity model for teachers in higher education: a preliminary study. In: 2011 International Conference on Frontiers in Education: Computer Science and Computer Engineering (2011)
11. Williams, P.: A practical application of CMM to medical security capability. *Inf. Manag. Comput. Secur.* **16**(1), 58–73 (2008)
12. Neuhauser, C.: A maturity model: does it provide a path for online course design. *J. Interact. Online Learn.* **3**(1), 1–17 (2004)

13. Bofinger, M., Robinson, N., Lindsay, P., Spiers, M., Ashford, M., Pitman, A.: Experience with extending CMMISM for safety related applications. In: 12th Annual Symposium International Council on Systems Engineering, Las Vegas, NV (2002)
14. Fujimoto, R., Bock, C., Chen, W., Page, E., Panchal, J.H.: Research challenges in modeling and simulation for engineering complex systems (2017)
15. Duda, H., Gerlach, T., Advani, S., Potter, M.: Design of the DLR AVES research flight simulator. In: AIAA Modeling and Simulation Technologies Conference (2013)
16. Gerlach, T., Durak, U.: AVES SDK: bridging the gap between simulator and flight systems designer. In: AIAA Modeling and Simulation Technologies Conference (2015)



Learning State Mappings in Multi-Level-Simulation

Stefan Wittek^(✉) and Andreas Rausch

Department of Informatics, Clausthal University of Technology,
Clausthal-Zellerfeld, Germany
{switt,arau}@tu-clausthal.de

Abstract. Holistic simulation aids the engineering of cyber physical systems. However, its complexity makes it expensive regarding computation time and modeling effort. We introduce multi-level-simulation (Our Multi-Level-Simulation approach was already published in [1]). The description of our approach in this paper is based on this publication and updates it. This description is the context to the results on learning State mappings within Multi-Level-Simulations presented in this paper.) as a methodology to handle this complexity. In this methodology, the required holistic perspective is reached on a coarse level, which is linked with multiple detailed models of small sections of the system. In order to co-simulate the levels, mappings between their states are required. This paper gives an insight into the current state of progress of using well known machine learning techniques for regression to generate these mappings using small sets of labeled training data.

Keywords: Co-simulation · Multi-level-simulation · Machine learning
Regression

1 Introduction

Cyber physical systems (CPS) are large scale sets of hardware and software components interacting in numerous ways to realize some common behavior. Examples of such systems are automated production facilities or networks of autonomous vehicles.

Applying simulation to CPS provides numerous chances. Aside from the possible reduction of prototyping effort, the system can be improved and its costs can be lowered. Real-time simulations can be employed at runtime to infer from a few measurement points to numerous virtual sensors located in between these physical sensors. This allows to reduce the amount and quality of sensory equipment used, which in turn leads to efficient designs. The cost of the system can also be lowered by allowing deviation in the physical part of the CPS. If these deviations (i.e. the bending of a robot arm due to the mass it is lifting) is well known through simulation, it can be compensated using the software part of the system. This compensation in turn can be evaluated in simulation.

In order to seize the opportunity laying in simulation for CPS it must be both: Holistic enough to capture the dependencies between its components, but only as complex as feasible, regarding modelling effort and computation times.

Figure 1 describes this trade-off in more detail. In the figure, simulation scenes are depicted as dots and arranged according to their complexity and holism (x- and y-axis). Without considering resources available, a simulation of the whole system is modeled, using the most detailed technique available, which provides the potentially most accurate results. Under restricted resources and for larger systems, such a simulation is infeasible.

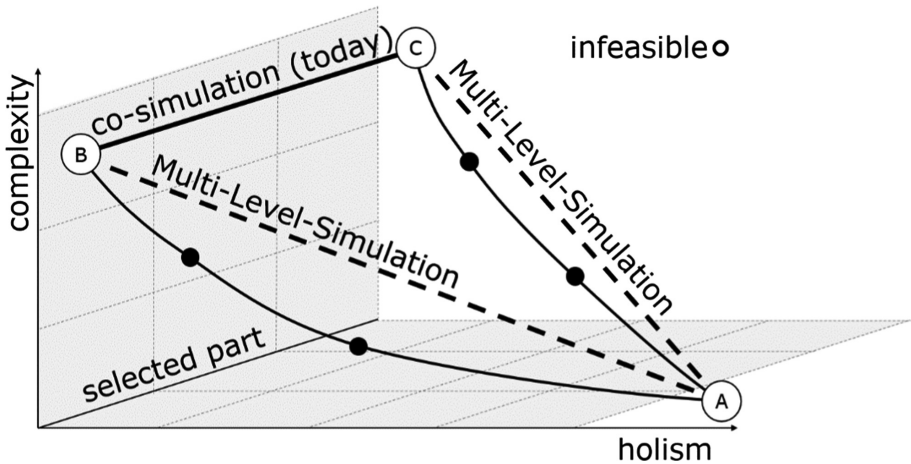


Fig. 1. Trade-off between complexity and holism in simulation.

Instead, the whole system can only be simulated using a coarse simulation (A). If more detail is needed to answer specific questions, only a smaller portion of the system can be considered, ultimately leading to a very small part of the system using the most detailed technic (B). Such a zoom-in could be done with any part of the system, i.e. (C). But, since these scenes are simulated independently, interdependencies between them are lost. To acquire a more holistic view, these scenes can be connected directly using a methodology or by building interfaces between these scenes and employing co-simulation. Both approaches are difficult, expensive and often only valid for particular instances of these scenes.

Therefore we propose a simulation methodology that is efficient regarding complexity called multi-level-simulation. We simulate the CPS on multiple levels of abstraction simultaneously. On a coarse level, a holistic perspective of the system is reached. This level is focusing on the interactions between the elements of the system. It uses a simplified modeling technique and is less complex, resulting in a fast simulation. On a detailed level, only some relevant sections of the system are chosen to be simulated using a more complex technique. These level allows precise predictions required for specific simulation goals, but results in a slow simulation. Note that these sections are not tied to the modularity of the components, but to the purpose of the simulation. They may include several components or only fragments of them.

In order to run a co-simulation between the levels, the state of these detailed sections and the state of the corresponding sections on the coarse level have to be synchronized. Because the coarse level uses a simplified modelling technique, the states must be converted using the mappings Up and $Down$. $Up(x) = y$ maps the detailed level state x to a coarse level state y . Note that a set of detailed level states A is mapped to the same y on the coarse level ($Up(x) = y, \forall x \in A$). Up is not injective and thusly not reversible. $Down(y) = x$ maps a coarse level state y to the detailed level state x . $Down(y)$ has to choose a single s among all states in A that would be mapped to x by Up . We aim to use machine learning techniques for regression to provide this mappings.

2 Related Research

In this section we will describe research related to our work. The co-simulation of heterogeneous systems is the aim of a variety of tools and frameworks. A selection of these works is presented. The idea to simulate systems on different levels of abstraction can be found in several approaches. Some focus on certain application domains while others aim to provide a general framework. We will discuss both directions. Cloud infrastructures in general and the deployment of simulation into this infrastructure are an active research field. We will provide a brief overview and discuss known approaches in this field.

A variety of works focus on the co-simulation of different simulations tools. Examples of this are the High Level Architecture (HLA) specification for simulation interoperability [2] the Functional Mockup Interface standard for model exchange and co-simulation [3] and the Mosaik Simulation API [4]. Another approach is to integrate different simulation semantics into a single tool. The Ptolemy project is an example for this approach [5]. All these works aim towards a holistic simulation of the system under investigation. The simulation of different abstraction levels is only addressed in terms of tool integration. The task to provide proper interfaces to connect simulation on different levels has to be done by the modeller.

Much effort is put into approaches that provide such concepts for specific domains such as material flows [6, 7], traffic [8] or agent based behavior simulation. They center on the dynamic switching of abstraction levels of model parts at runtime. To do so, explicit mappings between the states of different levels are provided. These mappings are tightly bound to the domain and the simulations they connect and are not designed to be generalizable.

Some research is conducted investigating more general concepts for the problem. The approach of Dynamic Component Substitution describes a co-simulation as a set of connected software components [9]. They communicate through given interfaces. Switching a part of the simulation to a more detailed version corresponds to substituting one such component with another. Both components are required to have exactly the same interfaces. This is a critical limitation. If the components are situated on different levels of abstraction, it is plausible to expect the same for their interfaces. Multi Resolution Entities [10] define a mapping that is used to synchronize the simulation state on different levels. These mappings are defined as invertible to use them in

both directions. This requirement is only met, if no information is lost mapping a detailed state to a more coarse state, which does not apply in general, as we will describe in Sect. 3. The concept of Multi Resolution Modelling Space introduces adapters between the interfaces and several mappings between the states of simulation on different levels [11]. However, the problem of information loss is not addressed in this approach.

Our approach of Multi-Level-Simulation is different from these approaches. We do not force the engineer to tailor the coarse level simulations into components connected by interfaces. We consider this approach as too inflexible. The coarse level can be modeled with no dependency on the detailed level. In fact, even cutting arbitrary parts out of existing coarse level simulations to be linked to a detailed level is possible. The detailed simulations are linked into a single simulation on the coarse level using a state synchronization mechanism.

This mechanism is the key towards multi-level-simulation. It requires mappings between the states of the different levels. Among the variety of works described in this section, two main solutions to provide this mappings can be distinguished. One group of the approaches provides specific instances of such mappings for a given combination of modeling techniques [6–8]. The other group is more general, but require an engineer to code the mappings [9–11]. This approaches are not optimal, because the mappings need to be defined per project and are hard to define explicitly.

Our approach aims to uses machine learning techniques for regression to generate *Up* and *Down*. The engineer only has to provide labeled training sets.

3 Multi-Level-Simulation

To describe our approach of Multi-Level-Simulation in more detail, we consider the example of a lift. Figure 2 shows the structure of this example.

(A) On the coarse level it consists of a simulation modelling the structure of the lift and a lift program. The structure consists of a shaft in which a cabin can move. The cabin is rigged to a cable. The weight of the cabin (w) is altered when it stops at one of the exits. A motor manipulates the length of the cable (l). The program simulation is connected to the structure and handles the speed and direction of the motor. In this setup all parts of the structure are modelled as rigid bodies. The program has no sensor for l and positions the cabin only indirectly using the last position of the cabin and a timer. On this level, realistic scenarios of use are modelled. An example of this would be a whole day cycle of an office building. Most persons want to go up in the morning and down in the evening. The simulation on this level is fast.

(B) During the development of the lift and its program the engineers want to investigate, if the stretching of the cable caused by the weight of the cabin and the aging of the cable will lead to a wrong positioning of the cabin. To do this, a detailed but computationally intensive simulation of the cable is activated. This simulation is stateful to reflect the aging of the cable. Only the parts of the cable that are stretched in a particular time step age. If the misplacement is a problem, the engineer has to implement an extension to deal with the phenomena in the program.

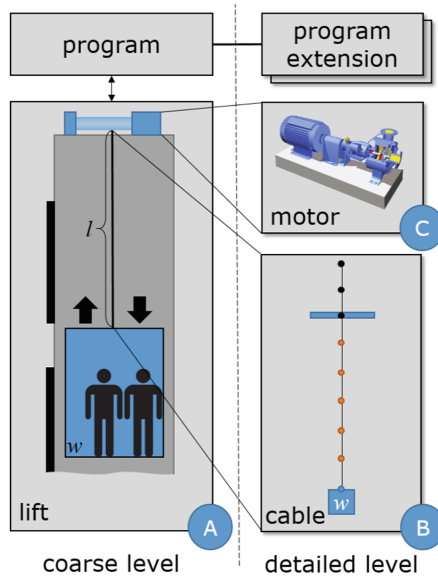


Fig. 2. Structure of the lift example.

(C) After this, the dynamics of the cabin are investigated closely. A computationally intensive simulation of the motor is activated. This simulation models the acceleration of the motor and allows to precisely determine the travel times of the lift. The simulation is stateful to model the heating of the motor which influences acceleration. Because the stretching of the cable is considered irrelevant for this question, the cable simulation and the corresponding program extension are deactivated. Because the program on the coarse level does not account for the acceleration when calculating the timers, a corresponding extension must be implemented and linked to the program.

Note that the program finally deployed needs to include both program extensions.

In both cases, parts of the lift are simulated on two levels at the same time. This leads to the challenge of maintaining the consistence between the states of both levels. If for example in (A) l is increased by 0.1 m, all elements of the cable in (B) must be placed 0.1 m lower. If in (B) the cable is stretched by 10%, displacing the lowest point from -3 m to -3.3 m, l must be set from 3 m to 3.3 m in (A).

Figure 3 shows a schematic overview of the example. Each simulation consists of two parts. The state of a simulation is defined as a valuation of a fixed set of attributes. The behaviour of a simulation is defined as a mapping which has this state as input and produces a new state as output. Considering the lift simulation in the example, this state $y = (w, l)$ is mapped by a to a succeeding state y' . This corresponds to a step in the simulation. Note that this representation of simulations is used in many approaches for co-simulation like Ptolemy [5] or the FMI standard [3] for model exchange. For all simulations the time Δt elapsing in one step is the same. The coarse simulation of the lift is linked to a number of detailed simulations. Note that in the lift example only one of these simulations is connected in a particular simulation run.

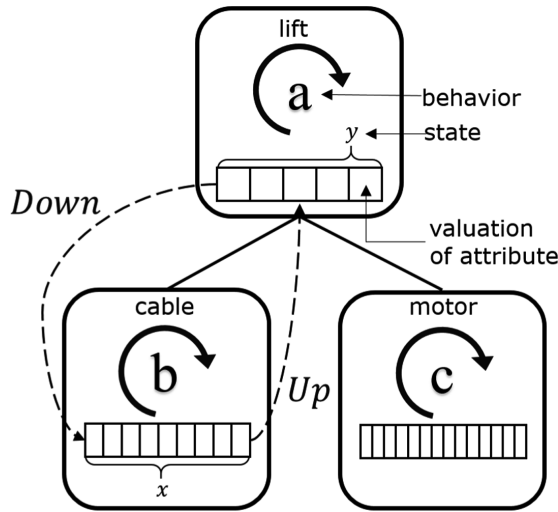


Fig. 3. The problem of state synchronisation.

Because the simulation models (i.e. the cable) are different, the attributes valued in a state x and y are different. The states need to be converted between the simulations. This is done using the state mappings Up and $Down$. Up maps the detailed level state to the coarse level state. $Up(x) = y$. It is typically not reversible, because information is lost. Referring to the lift example, there are a number of different positions and age levels of the cable elements that map to the same l . $Down$ maps the coarse level state to the detailed level. In the example, $Down$ restores the position of the cable elements using only l . To do so, $Down$ has to choose among a possible infinite set of states that are mapped to l by Up . To account for this problem, we propose $Down$ as a mapping of the coarse state and the last state of the detailed state. Figure 4 gives an overview of the execution of the example. In general, changes on different levels occur concurrently, regarding simulation time.

Let us consider the lift simulation starts with the initial state y and the cable simulation with the initial state x . The states are chosen so that $Up(x) = y$. The coarse level state y is an abstraction of the detailed state x . Now both simulations step using the behaviour functions a and b . The cable simulation ages a number of cable elements, stretching the cable by 0.1 m leading to the state x' . In the same time step, the lift simulation unwinds the cable by 0.2 m according to the initial speed of the motor, leading to the state y' . Converting y' to a state of the cable simulation using $Down$ results in an intermediate state \hat{x}' . This state is in conflict to x' which was calculated using the behaviour b of the cable simulation. Simply overwriting x' using \hat{x}' would annihilate the unwinding of the cable. To avoid this, an integrator function I must be employed to merge the two states. The resulting state contains both changes. Using Up on this state leads to an integrated state of the lift simulation that contains again both changes. This state finally becomes the new state of the lift.

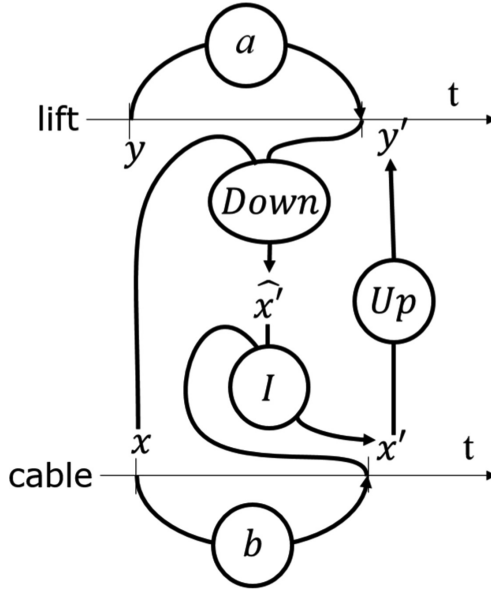


Fig. 4. Execution of the lift example.

4 Learning State Mappings

In the current state of our work we focus on the mapping Up . The state of each simulation can be formalized as a n-dimensional vector of real values. This vector captures the valuation of each variable of the simulation and is updated by the behaviour of the simulation as described in Sect. 3.

In practice this vector will only be a subset of the variables of the actual state of the simulation, because not all variables are affected by the state synchronization. A simple example of this can be found in the lift example. When the detailed simulation of the cable is active and its state is synchronized with the coarse simulation of the lift, Up will only map to variables of the lift representing the cable, like l and not i.e. the variable containing the number of passengers in the lift. The selection of this subset has to be done by the engineer. In other words, the engineer explicitly links coarse to detailed level states, that are reasonable to be synchronized.

We assume that each of the outputs of Up can be predicted independently. To handle m-dimensional output, m independent regression problems have to be solved. Thus, we will consider $Up : \mathbb{R}^n \rightarrow \mathbb{R}$ as a many-to-one mapping. There exist several well known algorithms for regression to solve this problem.

As described above, an engineer is supposed to provide a training set. This set contains k observations $X = \{x_1, x_2, \dots, x_k\}, x_i \in \mathbb{R}^n$ of detailed level states and their corresponding coarse level states Y where $Y = \{y_1, y_2, \dots, y_k\}, y_i \in \mathbb{R}$ and x_i corresponds to y_i for $i = 1..k$.

We focused this paper on representatives of three large classes of this algorithms. Namely these three classes are linear base function models, sparse kernel machines and

neuronal networks (following the definitions given in [12]). Linear base function models describe the output as a linear combination of a fixed set of nonlinear base functions. The weights associated to these base functions are the parameters of the model. The models are fitted using approaches to minimise the sum of squared errors or to maximise the log-likelihood on the training set. Examples of this group are linear and polynomial regression. Sparse kernel machines fit a plane between data points that maximizes the margin between the plane and the points. The kernel trick is used to embed the data in high dimensional space to become linearly separable by the plane. They are named sparse because the output only depends on a subset of the learned training data. Support vector machines and Relevance vector machines are examples of this class. Neuronal networks model the generative function behind the observations as the set of connected artificial neurons organized in layers. The training of the network is again based on the minimization of the squared error but uses the so call back-propagation algorithm to efficiently calculate the gradient of the weights in deeper layers.

We used the following setup to investigate the performance of three well known algorithms for regression out of these classes.

Four different data sets were defined. The LIFT data set was derived from the lift example described in Sect. 3. An implementation of the cable was used to generate 151 9-dimensional x values. The hand coded Up mapping of the example was utilized to generate the corresponding y values. Three additional data sets were generated using 150 random generated, normally distributed 9-dimensional real values $x \in [0, 1]^9 \subset \mathbb{R}^9$ and the basic aggregation functions SUM, MEAN and MEDIAN to generate the corresponding y values.

The training data was sampled from each of these sets using three distinct methods. A set of 8 random drawn values (R8). A set of 20 random drawn values (R20). A set of 8 values selected manually, simulating the engineer providing 8 “useful” examples (S8). The rest of the data sets was used as test sets.

Three algorithms were used to generate predictors for y . A polynomial regression minimizing the squared error (POLY). A ϵ -support-vector-regression (SVR) using a radial kernel function. A multilayer feedforward network trained by the RProp algorithm (MLP). This setup results in 4(data sets) \times 3(sampling methods) \times 3(algorithms) = 36 training runs.

Figure 5 depicts the results of six of these runs. The horizontal axis of each diagram rates the labelled value of y while the vertical axis rates the prediction by the algorithm.

The Table 1 summarizes the mean squared error of the predictors for all three sampling methods. Because the training sets S8 and R8 had 8 data points with 9 Dimensions, the Polynomial regression does not find a solution if the dimension of the data is higher than the number of data points and simply predicts a constant output. This result was cleaned form the table.

In all runs the results on the S8 sample worked better than on the R8 sample for all algorithms. Over all setups neither SVR nor MLP were dominant. POLY using R20 setup was better than SVR and MLP with the SUM and MEAN data set, but worse with MEDIAN data set.

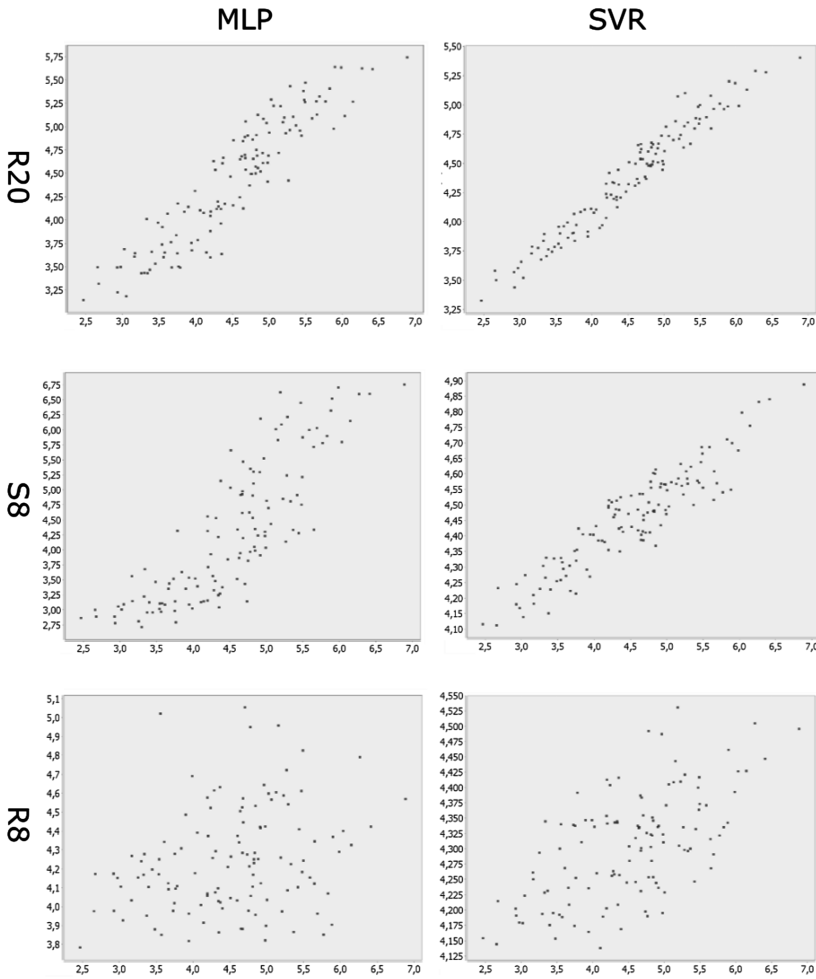


Fig. 5. Performance of SVR on the SUM data set using the three different sampling methods. The horizontal axis rates the y value labelled to that point in the test set while the vertical axis rates prediction by algorithm.

Table 1. Report of the mean squared error of all runs.

| Mean squared error | | MLP | SVR | POLY |
|--------------------|-----|--------|--------|----------|
| LIFT | R20 | 0,0015 | 0,2667 | 0,03120 |
| | S8 | 0,0025 | 0,1818 | – |
| | R8 | 0,0032 | 0,6385 | – |
| SUM | R20 | 0,0082 | 0,2338 | 5,25E-09 |
| | S8 | 0,0219 | 0,5235 | – |
| | R8 | 0,0410 | 0,7604 | – |

(continued)

Table 1. (continued)

| Mean squared error | | MLP | SVR | POLY |
|--------------------|-----|--------|--------|----------|
| MEAN | R20 | 0,0058 | 0,0086 | 4,88E-10 |
| | S8 | 0,0225 | 0,0012 | – |
| | R8 | 0,0413 | 0,0100 | – |
| MEDIAN | R20 | 0,0182 | 0,0126 | 0,3437 |
| | S8 | 0,0179 | 0,0058 | – |
| | R8 | 0,0913 | 0,0172 | – |

5 Conclusion and Future Work

The presented results provide a first insight into the performance of machine learning algorithms for multi-level-simulation. Further investigations are needed to provide actual evidence for our findings. It fits the intuition, that manually selected samples perform better than randomly drawn ones. Assisting the engineer in this task could lead to even better results. The good performance of POLY with the SUM and MEAN data set is in line with our expectations since both functions can easily be expressed as a polynomial. Although the overall performance of SVR and MPL was within our expectations, we aim to improve the predictions by providing additional guidance for the algorithms using constraints provided by the engineer. Examples of these constraints would be boundaries for the target values or structural knowledge about the states.

Another important next direction is to find a suitable learning approach for the *Down* function. This is significantly harder, because the maximum likelihood or minimum error approach will provide only a single deterministic x for a given y which is a very poor model for *Down*. Actually a indeterministic *Down* function is needed. The distribution of the output of such a *Down* function needs to be in line with the distribution of the observations.

Acknowledgments. We thank the Simulationswissenschaftliches Zentrum Clausthal-Göttingen (SWZ) for financial support.

References

1. Wittek, S.H.A., Göttsche, M., Rausch, A., Grabowski, J.: Towards multi-level-simulation using dynamic cloud environments. In: Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 297–303 (2016). <https://doi.org/10.5220/0005997502970303>
2. Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M.: The Department of Defense High Level Architecture (1997)
3. Blochwitz, T., et al.: Functional mockup interface 2.0: the standard for tool independent exchange of simulation models (2012)

4. Schütte, S., Scherfke, S., Tröschel, M.: Mosaik: a framework for modular simulation of active components in Smart Grids. In: 2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS), pp. 55–60. IEEE (2011)
5. Eker, J., et al.: Taming heterogeneity-the Ptolemy approach. *Proc. IEEE* **91**, 127–144 (2003)
6. Dangelmaier, W., Mueck, B.: Using dynamic multiresolution modelling to analyze large material flow systems. In: Proceedings of the 36th Conference on Winter Simulation, WSC 2004. Winter Simulation Conference, Washington, D.C., pp. 1720–1727 (2004)
7. Huber, D., Dangelmaier, W.: A method for simulation state mapping between discrete event material flow models of different level of detail. In: Proceedings of the Winter Simulation Conference, WSC 2011. Winter Simulation Conference, Phoenix, Arizona, pp. 2877–2886 (2011)
8. Claes, R., Holvoet, T.: Multi-model traffic microsimulations. In: Winter Simulation Conference, WSC 2009. Winter Simulation Conference, Austin, Texas, pp. 1113–1123 (2009)
9. Rao, D.M.: Study of dynamic component substitutions. Dissertation, University of Cincinnati (2003)
10. Reynolds Jr., P.F., Natrajan, A., Srinivasan, S.: Consistency maintenance in multiresolution simulation. *ACM Trans. Model. Comput. Simul.* **7**, 368–392 (1997). <https://doi.org/10.1145/259207.259235>
11. Hong, S.-Y., Kim, T.G.: Specification of multi-resolution modeling space for multi-resolution system. *Simulation* **89**, 28–40 (2013). <https://doi.org/10.1177/0037549712450361>
12. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, New York (2006)



Unifying Radio-in-the-Loop Channel Emulation and Network Protocol Simulation to Improve Wireless Sensor Network Evaluation

Sebastian Böhm^(✉) and Michael Kirsche

Computer Networks and Communication Systems Group,
Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany
{sebastian.boehm,michael.kirsche}@b-tu.de

Abstract. Evaluations of Internet of Things (IoT) and Wireless Sensor Network (WSN) applications demonstrate the significant and still existing gap between examinations with generic simulation environments and real-life (e.g., field test) or controlled (e.g., testbed) sensor network deployments in terms of realistic and accurate results. The separated use of single examination approaches is often not enough to overcome all evaluation challenges. We therefore propose a combination of discrete-event simulation, radio-channel emulation, and real hardware working together on different layers of the protocol stack of the system-under-test. Our combined approach reduces the gap between abstract simulations and network testbed experiments by providing adjustable radio conditions for repeatable evaluations of WSN and IoT networks.

1 Introduction

Practical applications of Wireless Sensor Networks (WSNs) require extensive testing and evaluation strategies that cover all layers of the protocol stack: from application and protocol data flows to radio channel influences and physical side-effects. These pre-deployment tests can be performed either via simulation, through emulation, or with the help of real-life testbeds. While a simulation of a networked system requires (abstract) models and representations of the system-under-test, emulations use components from the original system and try to emulate the usage scenarios. Real-life testbeds, in turn, try to replicate the exact application conditions prior to a rollout. Each of these network evaluation techniques has specific strengths and weaknesses when applied to WSN and IoT use cases. We discuss all three techniques in the next paragraphs and motivate our hybrid approach that combines simulative and emulative methods. Additional discussions of the evaluation methods are available in [1–4].

WSN and IoT Simulation

Simulation allows one to easily construct layered network protocol architectures, device topologies, and algorithmic applications. Simulation model parameters

can be customized to examine certain protocol behavior and performance issues. In the academic world, simulation is the de facto first step for the implementation of WSN solutions because no hardware is required to test new protocol designs. Common network simulators like *OMNeT++* [5] or *NS-3* [6] focus on the Discrete Event Simulation (DES) paradigm which models a system by its states. System state changes occur at discrete points in time. Examples of state changing events can be the start of a packet transmission or the expiration of a timer. A DES simulator jumps from one event to the next, skipping the time between events. Pseudo Random Number Generators (PRNGs) are used within the simulation process to randomize state changes and event occurrence. Simulator implementations use a Future Event Set (FES) and event routines (handlers) to create and schedule events. Algorithm 1 (adapted from [7]) shows the described general operations flow of discrete event scheduling in pseudo-code.

Algorithm 1. Discrete Event Scheduling

Precondition: Initialize *simulation model* and *FES*

```

while (FES not empty) and (simulation not complete) do
  | fetch first event e from FES
  | advance simtime with event timestamp t
  | Function process_event(e):
  |   | perform model state transition
  |   | if (create new event v) or (cancel event w) then
  |   |   | insert v into FES and delete w from FES
  |   |   end
  |   return
end

```

The radio channel conditions and the operation environment characteristics have a strong influence on WSNs. These two aspects are typically simplified or completely omitted in network protocol simulations. Add-on frameworks that enable the modeling of wireless environments and channel conditions are discussed in [8, Sect. 2]. Their implementation, however, is based on mathematical functions with varying abstraction levels and complexity. Wireless channel modeling is a very complicated process (cp. [9]) with a significant tradeoff between the realism of simulation models and their performance and scalability [8, Sect. 4]. These drawbacks generally lead to a low confidence in WSN simulation results.

WSN and IoT Testbeds

Application-specific *physical testbeds* for distributed sensor networks, on the other hand, are complex to set-up, to manage, and to operate. They are also cost intensive in comparison to other evaluation approaches. [10] summarizes and compares existing WSN experimentation testbeds and their characteristics. Regardless of the application scenario, testbed nodes operate in a radio

environment which is often uncontrollable and may differ from the actual working environment of the system-under-test. Testbed evaluations enable an accurate representation of a sensor node’s hardware (if available and installed in the testbed) and software. Monitoring and inspection of run-time characteristics is possible when firmware extensions and additional hardware are used to provide a feedback loop. However, such extensions may change the run-time execution or introduce unforeseen behavior and errors during the evaluation process.

Radio Channel Emulation

Radio Channel Emulation offers a controllable radio environment with constant Radio Frequency (RF) conditions to evaluate a system-under-test. WSN nodes are isolated from each other and connected over their radio interfaces to the RF channel emulator hardware, which emulates signal propagation effects. A common and important feature of these systems is the ability to control the large-scale fading between transmitters and receivers. Practical deployments vary from laboratory test setups with coaxial-based radio links to complex analog or digital radio channel emulators with support for hundreds of interconnected nodes. Nevertheless, applications run as static firmware implementations on real-life WSN hardware and flexible adjustments of applications or protocol parameters are complicated compared to network simulators.

The evaluation of WSN designs still offers many unsolved research challenges (cp. [3] for example), as the separated use of single approaches is often not enough to overcome all evaluation challenges (e.g., realistic cross-layer communication evaluation). To mitigate most of the flaws of the evaluation methods discussed in the previous three paragraphs we propose a combination of simulation, radio-channel emulation, and real hardware working together on different layers of the protocol stack of the system-under-test. The remainder of the paper describes the methodology of our hybrid approach in Sect. 2, gives a short overview of our background and related work in Sect. 3, presents a case study of our Hardware-in-the-Loop-based network emulation in Sect. 4, and discusses evaluation methods in Sect. 5 before we conclude the paper in Sect. 6.

2 Methodology

Hardware-in-the-Loop (HIL) evaluation concepts are well established for automotive, aerospace, and robotic application domains. In these areas, HIL approaches are used to verify existing hardware module implementations by triggering the hardware inputs via simulated events and observing the generated outputs and reactions of the hardware-under-test. In case of WSNs and wireless networked embedded systems, HIL concepts (e.g., [11, 12]) for the evaluation of WSN applications and protocols are rarely implemented (cp. [13]). Depending on the development stage and the actual part of the system that is tested and fed with simulated data, name variations like model- or software-in-the-loop are widely-used. Our approach of combining radio channel emulation with a HIL concept for WSN testing is subsequently called Radio-in-the-Loop (RIL).

Figure 1 depicts a protocol stack of a sensor node and the three collaboration levels (numbered ① to ③). Time-discrete event simulations (level ① in Fig. 1) of the upper layers of the protocol stack (e.g., application, transport, and network layer) simplify the construction of different architectures and application models by using simulators like OMNeT++ and accompanied frameworks like INET¹.

Using real hardware (level ② in Fig. 1) to accurately represent parts of the Medium Access Control (MAC) sublayer and the complete Physical (PHY) layer mitigates a major drawback of typical WSN simulations. MAC and especially PHY simulation models are often abstracted, even though simulation frameworks might provide accurate models for upper layer protocols. WSN-specific simulators like Cooja [14] can provide more accurate representations of real WSN hardware due to real-life code execution, but their PHY and wireless propagation models are abstracted just like the ones from generic simulators like OMNeT++. Even more problematic is the restriction of WSN-specific simulators like Cooja to specific operating systems (i.e., Contiki [15] for Cooja). We combine real WSN hardware with upper layer network simulations over a Hardware-in-the-Loop approach to enable an accurate representation of the lower layers combined with the protocol and model variety of OMNeT++/INET.

We use RF channel emulation (level ③ in Fig. 1) to bypass the discussed drawbacks of WSN testbeds. By emulating the wireless channel in a controllable environment, we can provide adjustable PHY conditions comparable to simulation environments, while refraining from using abstracted wireless propagation and PHY models. Section 4 describes all three collaboration levels in detail. This Radio-in-the-Loop concept leverages on the strengths of the individual approaches to enable accurate and likewise flexible tests of WSNs.

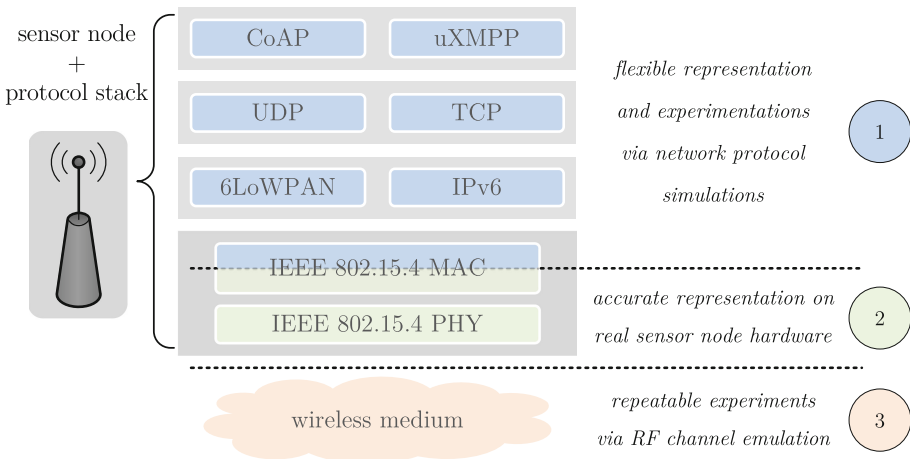


Fig. 1. Combining protocol simulation, real hardware, and RF channel emulation

¹ INET framework website: <https://inet.omnetpp.org/>.

The following three example scenarios benefit from using our RIL approach:

- (i) The radio channel emulation (level ③ in Fig. 1) enables the definition of exceptional network topologies and various channel conditions that are difficult to reproduce within real world tests, while we can observe the behavior of the upper layer protocols on the simulated level.
- (ii) With respect to the first scenario, cross-layer communication approaches and protocols that also involve radio medium access and physical data transmissions can benefit from accurate parameters and measurements of the transceiver chip hardware (level ② in Fig. 1) to get dependable evaluation results within a simulation-driven test setup.
- (iii) Nevertheless, a small-scale setup of nodes in a controlled channel emulation environment can be extended easily with additional purely virtual nodes from collaboration level ① in Fig. 1 to increase the network traffic and simulate communication data stimuli for real wireless transmissions.

3 Related Work and Background Information

The majority of current simulation approaches [16] abstract heavily from real target hardware. Specialized simulators as discussed in [8, 17] allow only a partial modeling of wireless environments and channel conditions. The underlying models are always based on abstracted mathematical functions with varying complexity. COOJA [14] provides an operation-system-specific simulation engine to enable software-based emulation of WSNs with a focus on simulating the code execution on the target hardware. COOJA supports three different models for wireless transmissions with varying parametrization: Unit Disk Graph Medium (UDGM), Direct Graph Radio Medium (DGRM), and Multi-path Ray-tracer Medium (MRM). Other simulation and emulation concepts for signal propagation or PHY layer support are surveyed, for example, in [8] and [16].

Coupling OMNeT++ with hardware is also considered in [18]. The authors describe a new HIL interface and changes in the OMNeT++ event scheduler to support a real-time exchange of messages over external hardware interfaces. [18] differs from our approach in terms of the underlying idea of joining radio channel emulation with simulation and in regard of the application scenario (i.e., home automation). The report describes an OMNeT++ gateway that basically forwards messages from one real-life device to another one via OMNeT++ without considering a radio channel emulation. Tests and evaluation of real-life hardware via simulated stimuli are also not yet considered in [18].

3.1 Network Protocol Simulation with OMNeT++

OMNeT++ is a popular open source DES simulator that is frequently used for communication network research. OMNeT++'s concept of exchanging messages via gates between modules (reusable building blocks) facilitates the development and simulation of complex scenarios. The INET add-on framework provides a multitude of simulation models for upper layer protocols.

In [19], we introduced a new simulation model for the popular IoT and WSN communication standard IEEE 802.15.4 [20]. The OMNeT++ model was created to simulate the complex behavior of the 802.15.4 MAC and PHY layers in a detailed fashion. We modeled the two layers with their connecting interfaces and the used service primitives according to the IEEE standard specifications and general modeling guidelines for 802.15.4 [21]. The structure of the OMNeT++ simulation model is depicted in Fig. 2.

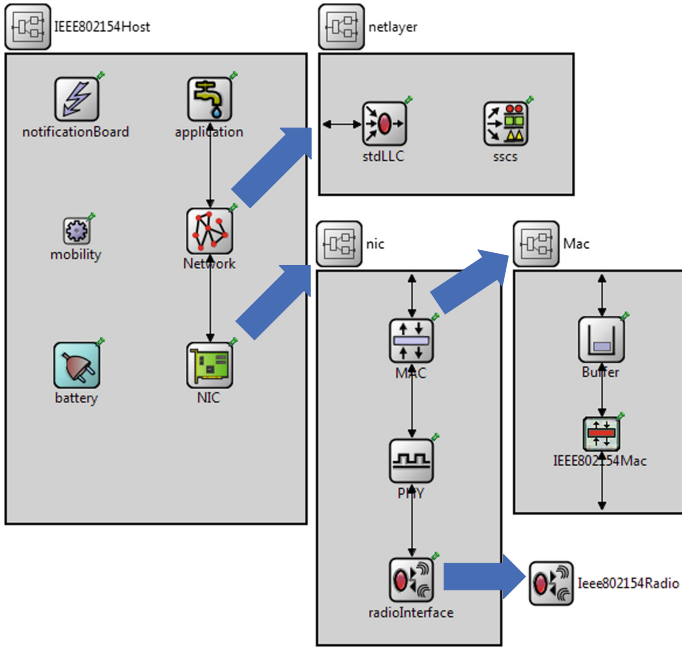


Fig. 2. Block diagram of the OMNeT++IEEE 802.15.4 model

The model itself consists of several layers and individual model components that are combined into a so-called *IEEE802154Host*. The *NIC* includes the parts of the IEEE standard that are most relevant for the communication. Listing 1.1 shows that the Protocol Data Unit (PDU) packet definitions include all types and fields that are specified in the IEEE 802.15.4 standard [20].

```

packet mpdu
{
    unsigned short fcs;           // 16-Bit Frame Check Sequence
    Ash ash;                     // Auxiliary Security Header
    // ... MAC frame payload is encapsulated ...
    MACAddressExt src;           // 0, 16 or 64-Bit Source Address
    unsigned short srcPANid;     // 0 or 16-Bits for Source PAN ID
    MACAddressExt dest;         // 0, 16 or 64-Bit Destination Address
    unsigned short destPANid;    // 0 or 16-Bits for Destination PAN ID
    unsigned char sqnr=0;        // 8-Bit Sequence number
    unsigned short fcf=0;        // 16-Bit Frame Control Field
}
    
```

Listing 1.1. MPDU packet definition - excerpt from MPDU.msg

3.2 RF Channel Emulation with the RoSeNet Testbed

In addition to OMNeT++, we work with RoSeNet², a network emulation platform for low-power wireless technologies that focuses on hardware-based channel emulation via a controllable coaxial cable radio environment (level ③ in Fig. 1). The modular system incorporates interconnected *emulation panels* that manage multiple sensor nodes on designated slots. By adjusting the signal attenuation values among nodes and panels in this shielded RF environment, it is possible to emulate distances or geographical positions and topologies of networked nodes. At designated signal supply points, interference signals can also be injected into the signal path. The overall architecture enables the emulation of large-scale networks with up to 1000 wireless sensor nodes (Fig. 3).



Fig. 3. RoSeNet emulation and test platform (taken from RoSeNet web page)

For our first HIL coupling experiments [22], we developed interfaces to transmit generated MAC layer frames via real sensor node hardware to achieve control over the communication flow in the network. In order to generate traffic for an initial test of the HIL system we simply inject generated protocol data frames. We added message handlers for the *emulation control server* application that are responsible to start the packet transmission to the hardware. The *Management Controller* on the addressed hardware panel forwards this packet to the designated slot of the transmitter node. A detailed description of the frame transmission and reception at node level is given in Sect. 4.3.

4 An OMNeT++ and RoSeNet RIL Architecture

Our current prototype includes extensions for OMNeT++/INET and the IEEE 802.15.4 simulation model (cp. Sect. 3.1), the hardware interfaces on the RoSeNet emulation testbed, and a *Forwarder* implementation that acts as bridge between the two domains. Figure 4 illustrates the basic Radio-in-the-Loop setup and the abstracted message exchange among the collaborating entities.

² RoSeNet radio channel emulation platform: <https://www.dresden-elektronik.de/ingenieurtechnik/development/research/rosenet/>.

For data exchange, we use the Packet Capture (PCAP) file format, the de facto standard capture format for network packet traffic. With the *PCAP Next Generation (PCAPNG)*³ extension, we can exchange additional information for data packets, for example the interface identifier for multiple external devices.

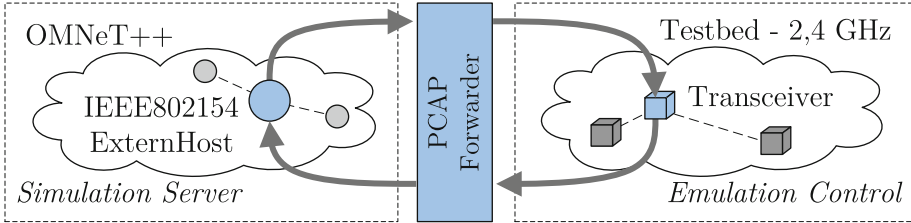


Fig. 4. Simplified message flow between OMNeT++ and testbed hardware

4.1 RIL Simulator Interfaces

We extended the IEEE 802.15.4 simulation model with a new module, called `IEEE802154ExtHost`, that enables the RIL operation. In the current implementation, an *external host* sends simulated MAC frames to an *external interface* instead of the PHY model. Figure 5 shows that the `IEEE802154ExtInterface` operates with a `IEEE802154Serializer` and a `PCAPScheduler` to convert between raw PCAP data bytes and OMNeT++ MAC frame objects.

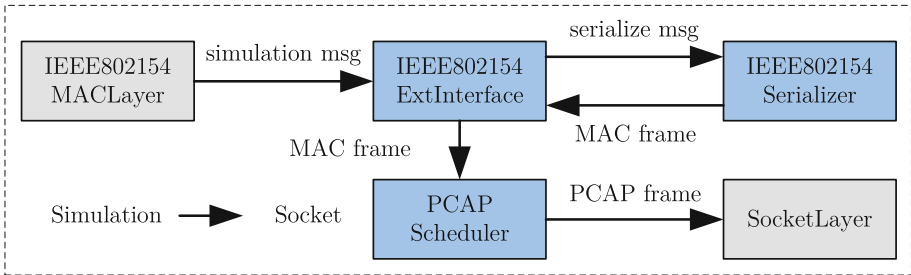


Fig. 5. IEEE 802.15.4 external interface in OMNeT++/INET

`PCAPScheduler` is derived from OMNeT++’s `cRealTimeScheduler` class. In OMNeT++, the event scheduler is one of the most important components, as it controls the event processing and manipulates the Future Event Set (cp. Sect. 1). The scheduler class provides a function, called `setInterfaceModule()`, which enables the connection of external interfaces (i.e., a Transmission Control Protocol (TCP) socket in our case) to the simulation. The scheduler function

³ PCAPNG capture file format: <https://github.com/pcapng/pcapng>.

`getNextEvent()` is synchronized to the real-time clock and checks periodically if an event occurred at the socket. With respect to the PCAPNG file format, the scheduler implements various functions for writing and especially reading the specified blocks from the socket stream in the separate `PCAPNGReader` module. For the handling of PCAPNG, we only implemented the three basic block types that are relevant to our use case. These are:

- `handleSHB()`: Section Header Block (SHB) *(init PCAP handling)*
- `handleIDB()`: Interface Description Block (IDB) *(set hardware interface)*
- `handleEPB()`: Enhanced Packet Block (EPB) *(process MAC packet)*

The `IEEE802154ExtInterface` cooperates with the `IEEE802154Serializer` module to convert between simulation and real-life packet formats. The *external interface* is also responsible for handling all incoming and outgoing packet data traffic for the *scheduler*. A function named `handleMessage()` deals with events that can be either Radio-in-the-Loop messages or regular simulation events. Incoming *external* 802.15.4 MAC frames are deserialized from the serializer and directly send to the corresponding simulation node module.

The mapping between simulation nodes and PCAPNG data is performed with the help of an interface table that stores the simulation module identifier for the corresponding hardware identifier.

4.2 PACP Forwarder

A transparent forwarder application interchanges data in the PCAPNG format between the simulator and the target emulation system. The implementation uses threads and acts as a dispatcher and aggregator of PCAPNG data streams. On the hardware side, assigned destinations can be single sensor node platforms, individual transceiver chips, or whole testbed control systems. While PCAPNG is able to handle multiple hardware interfaces and link layer protocols, the forwarder application can aggregate data from different end-devices to constitute a single socket data stream for further scheduling in the OMNeT++ simulation. Figure 6 depicts the main PCAPNG block types and the packet processing.

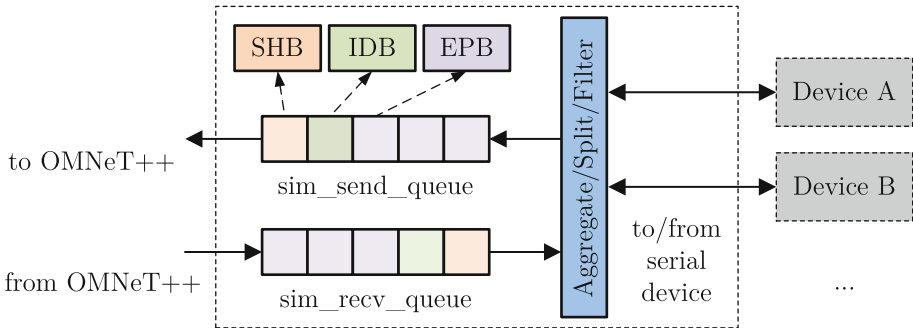


Fig. 6. PCAP forwarder architecture

For our purposes, we use the PCAPNG block types SHB, IDB, and EPB. The Section Header Block (SHB) and one Interface Description Block (IDB) for every used hardware node are exchanged at the beginning of the scenario execution. The Enhanced Packet Blocks (EPBs) represent the MAC data frames.

4.3 Node Simulation/Emulation Firmware

Contiki OS [15] is used to enable the reception of PCAP protocol frames via a Universal Asynchronous serial Receiver and Transmitter (UART) interface to create the respective MAC events and to transmit real radio frames among sensor nodes on the introduced RoSeNet emulation platform. We currently use nodes with ATmega128RFA1 radios⁴, but the Transceiver (TRX) firmware can also be implemented for an arbitrary node platform and other WSN operating systems. Algorithm 2 illustrates the transceiver process with the corresponding transmit and receive functions in pseudo-code, derived from our current Contiki-based implementation. The required *promiscuous mode* in Algorithm 2 enables a Network Interface Controller (NIC) to pass all received network traffic captured from the medium to the system's Central Processing Unit (CPU).

Algorithm 2. Nodes Transceiver Process

Precondition: Initialize radio driver in promiscuous mode

Function *receiver_callback()* ▷ called by radio driver

```

| record timestamp and create PCAP frame
| send out PCAP frame via serial interface
return
while (true) do
| wait until PCAP serial interface event occur
| copy MAC frame into the packetbuf
| Function sender_callback()
| | send out frame via radio interface
| return
end

```

Our implementation is actually using a fixed transmission channel with a permanently activated RF transceiver. The Carrier Sense Multiple Access Collision Avoidance (CSMA-CA) protocol is used to control and regulate the channel access. A *pcap_line* input process state machine creates PCAP events from frames received over the UART interface. A *sender_callback* function immediately transmits the frame onto the wireless channel. At the moment, we are able to transmit

⁴ ATmega128RFA1: <http://www.microchip.com/wwwproducts/en/ATmega128RFA1>.

frames in both directions between OMNeT++/INET and common RS232 UART interfaces of typical IEEE 802.15.4 transceiver chips (e.g., ATmega128RFA1) and sensor nodes on the introduced RoSeNet emulation system.

4.4 Radio Emulation

The setup of the emulation scenario and the necessary parameters follow the typical OMNeT++ guidelines for the creation of simulation scenarios. We create `.ned` and `.ini` files for OMNeT++ that include the definition of the network topology as well as necessary parameters and simulation options. By using RoSeNet's radio emulation architecture, we are able to emulate a long term fading of signal transmissions. With reference to the Free Space Path Loss (FSPL) (cp. [9, Sect. 2]), we can *arrange* testbed nodes in virtual geographic positions. We take two-dimensional coordinates of the modeled scenario and the RF parameters of the sensor node hardware to determine the signal fading between individual wireless sensor nodes. We compute the signal loss between nodes (FSPL) with a simplified uniform spread of energy in free space given by the path loss model in Eq. 1, adapted from [9, Sect. 2.5].

$$FSPL(dB) = -20 \log_{10} \left(\frac{4\pi d}{\lambda} \right) \quad (1)$$

λ – transmission channel center frequency wavelength
 d – the distance between sender and receiver

One important constraint is that RoSeNet has both fixed (i.e., integrated into the platform, non-adjustable) and variable signal attenuators (i.e., adjustable by the user) in its coaxial environment. The quintessence and problem at the same time is to allocate nodes and set all involved attenuators to their according values to achieve the desired radio topology. For our RIL scenarios, we therefore need to create radio topologies that are representable in the two domains.

Hardware Allocation on the Emulation Testbed

A RIL testbed architecture with coaxial-based radio links can be represented as an undirected communication graph G . With RoSeNet, we have a plain tree structure (cp. Fig. 7) in which the root is the central *anchor node* connected to *chains*. One chain has several modular entities called *panels*, which include the actual sensor nodes (the tree leaves), as it was introduced in Sect. 3.2.

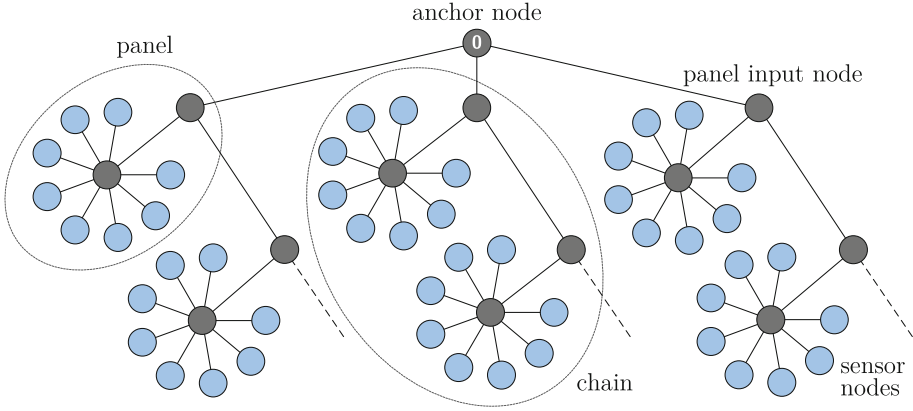


Fig. 7. RoSeNet’s emulation architecture as an undirected communication tree G_{AT}

We modeled the graph-based abstraction and several RF dependencies of the target emulation hardware for our allocation scheme. An overview of a number of important variables of the model definition is given below.

| | | |
|---------------|---------------------------------------|---|
| N : | a set of WSN hardware node types | $\{ 'RCB128RFA1', \dots \}$ |
| H : | a set of emulator RF node types | $\{ 'anchor', 'splitter', 'input' \}$ |
| A : | a set of static attenuation values | $\{ 1, 3, 11, 21, 31 \} \subseteq \mathbb{N}$ |
| I_λ : | a set of graph G invariants | |
| ϵ : | deviation of attenuation values | $\{ \epsilon \in \mathbb{R}_+ \mid \epsilon < 1 \}$ |
| a_e : | attenuation value of a graph G edge | $e \in E(G_S), a_e \in \mathbb{N}$ |
| G : | an undirected communication graph | $G = (V, E)$ |
| | a set of nodes from G | $V = \{ v \mid v \in N \}$ |
| | a set of weighted edges from G | $E \subseteq \{ (i, j, a) \mid i, j \in V; a \in \mathbb{N} \}$ |
| G_A : | an undirected allocation graph | $G_A = G = (V, E)$ |
| | a set of nodes from G_A | $V = \{ v \mid v \in \{ N \cup H \} \}$ |
| G_{AT} : | an undirected allocation tree | $G_{AT} \subseteq G_A = G = (V, E)$ |
| | a set of nodes from G_{AT} | $V = \{ v \mid v_r = anchor, v_l \in N \}$ |
| G_H : | an emulation hardware graph | $G_S = G_{AT}$ |
| G_S : | a scenario graph | $G_S = G = (V, E)$ |

We designed a multiple stage allocation process to be able to automatically find attenuation values of the RF signal path on testbed platforms like RoSeNet. We use graph analysis and Mixed Integer Linear Programming (MILP) based on graph representations for the scenario and the hardware platform for this. The abstracted allocation procedure is given in Algorithm 3.

Algorithm 3. Node Hardware Allocation

Precondition: $|\text{scenario nodes}| \leq |\text{hardware nodes}|$
Postcondition: $G_A \subseteq G_H$ with tolerance to a_e
 generate G_S and G_H
 $I_\lambda = \text{analyze scenario } G_S$ $\triangleright I_\lambda$ not finally specified

```

while (true) do
     $G_A = \text{allocate nodes from } G_H \text{ with } G_S \text{ and } I_\lambda$ 
    Function calculate_attenuation( $G_A, G_S$ )
         $\triangleright$  create the Linear Program and add constraints
        make LP from  $|V(G_S)|$  and  $|V(G_A)|$ 
        add LP scenario constraints from  $G_S$ 
         $\triangleright$  allow deviation from fixed values
        for  $\epsilon \in \text{Set}_\epsilon$  do
            add LP hardware constraints from  $G_H$  with  $\epsilon$ 
            solve LP
            if attenuation found then
                allocate graph  $G_A$ 
                return
            end
        end
    end
return
end
    
```

The allocation process includes multiple steps, starting with the generation of the scenario graph G_S and the emulation hardware graph G_H from the currently available hardware installation. In the second step, several parameters (graph invariants I_λ) of the scenario graph are calculated and processed by an initial node allocator, which makes a first decision regarding the panel-node placement G_A . This step of the process is only statically implemented for now. In the third step, we transfer our graph representation with the allocated nodes and all additional constraints (coming from the hardware and the RF dependencies) into a Linear Program (LP) which can then be solved using MILP. If a solution is calculable then we are done, otherwise we have to adjust our parameters (e.g., the deviation of the fixed attenuation parameters ϵ) or we need to move back to step two to calculate a new panel-node placement G_A . If no solution can be calculated at all (e.g., due to hardware constraints), we have to change the initial radio topology or inform the user of the unsupported scenario.

5 Evaluation

The preceding sections gave an overview of our methodology and introduced the different submodules of our hybrid RIL approach. As the implementation and testing of the OMNeT++ and RoSeNet RIL architecture are still ongoing work, we will discuss evaluation approaches for submodules and different aspects of our

approach and their feasibility to verify the usefulness of the OMNeT++/RoSeNet coupling for combined channel emulation and protocol simulation. Furthermore we present first results and fundamental steps of our submodule evaluation. A performance evaluation of the RIL simulator interfaces is discussed in Sect. 5.1. The PCAP Forwarder component is functionally evaluated to assure its correct behavior in Sect. 5.2. Section 5.3 shortly discusses a rudimentary performance evaluation of the emulation firmware prototype while Sect. 5.4 rounds off the evaluation part with an analysis of the hardware allocation and the MILP solver.

5.1 RIL Simulator Interfaces

An important aspect of real time simulations is the performance of the simulation implementation. When designing a simulation model, its implementation is usually less efficient when compared to the implementation of the real system. On the other hand, simulations typically run on high performance hosts. This difference is especially significant for WSN and IoT simulations of resource-constrained devices. Since we interconnect simulations and real hardware in our RIL approach, we have to ensure that the simulator maintains real-time capabilities for accessing the radio transceiver hardware.

Exemplary research like [23] already demonstrated sufficient throughput measurements of similar external interfaces for OMNeT++ in the past (as a result of their examples they fully utilize a link of 10 Mbit/s). When compared to the maximum throughput of IEEE 802.15.4 radio transceiver hardware (with a maximum over-the-air data rate of 250 kbit/s for the popular 2.45 GHz frequency band), we consider large scale test setups with dozens of sensor nodes to discover the limitations of our approach, for example in terms of the maximum number of RIL nodes. These results can finally be compared to the maximum theoretical throughput on a link.

As a fundamental evaluation step, we modified simulation modules to generate all frame types specified in the IEEE 802.15.4 standard [20, Sect. 7.2] and send them via our interface implementation. A test data receiver connects to the OMNeT++ socket at the local host and writes all received traffic into a single PCAP trace file. We use Wireshark⁵, the de facto standard network protocol analyzer, to verify the correctness of the frame transmission.

5.2 PCAP Forwarder

Our forwarder application acts as a transparent bridge between the simulation and the emulation domain. We thus have to ensure the fast reception, processing, and transmission of frames. Depending of the exact scenario, the forwarder application can be evaluated by performance measurements when aggregating, splitting, and filtering PCAP data streams. For running corresponding performance measurements we need to define these concrete scenarios and generate suitable PCAP traces or schedule test runs together with a simulation run. We

⁵ Wireshark network protocol analyzer: <https://www.wireshark.org/>.

consider precise time-stamping of processed PCAP frames with the help of high resolution clock timers to record the arrival and departure time for each frame.

We started by stress testing the application and measuring the overall data throughput by passing PCAP trace files from a sender socket via our forwarder to a receiver socket and vice versa (the receiver socket acts as a TCP repeater). Figure 8 shows the corresponding test setup (the dashed arrows show virtual connections; other connections represent the real data flow). We did not observe any dropped frames; all packets sent by the PCAP traffic generator were received by it after they passed through the PCAP forwarder and were returned by the TCP repeater. The test proves the correct functionality of the PCAP forwarder.

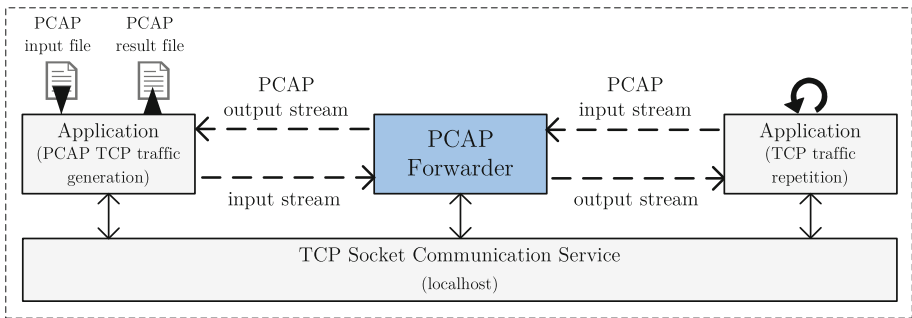


Fig. 8. Setup of the PCAP Forwarder application test

Looking at a data transfer of maximum sized IEEE 802.15.4 data frames (10k data frames with 127 bytes each – overall data amount of 1.52 MB), we measured an exemplary throughput of 91.5 Mbit/s from the sender (TCP traffic generation) to the PCAP Forwarder and 91.4 Mbit/s from the sender to the receiver module (TCP traffic repetition) on an Ubuntu Linux (64 bit) virtual machine with 7 vCPUs (Intel Xeon X3470 Quad Core @ 2.93 GHz). While traffic at the localhost gets processed by a loopback adapter in the kernel we cannot evaluate the precise throughput readings, but our first measurements show a 0.11% performance decrease caused by the forwarder. We assume that our packet handling routines, currently without considering aggregating, splitting, and filtering frames, do not have a significant influence on the overall data throughput.

5.3 Node Simulation/Emulation Firmware

For our emulation firmware prototype, we started with measurements of the maximum over-the-air transmission rates for different frame lengths for the currently used hardware platform RCB128RFA1⁶. We used the Texas Instruments

⁶ RCB128RFA1 Radio Controller Board: <http://www.dresden-elektronik.de/funktechnik/products/reference-designs/atmel-radio-controller-boards/radio-controller-boards/>.

(TI) CC2531⁷ transceiver module with TI’s own SmartRF Packet Sniffer⁸ software. We calculated the theoretical maximum transmission rate for frame transmissions between the serial and the RF interface, based on the used PHY specification and the microcontroller specs of our test hardware. Figure 9 depicts the results of the frame processing compared to the theoretical limitations. We achieve an overall throughput (independent of the frame size) of approximately >90% of the theoretical maximum packet rate at the serial interface. This performance decrease should primarily be caused by the PCAP-handling at the customized serial input driver and the frame type classification as well as the buffering of air-frames at the radio interface.

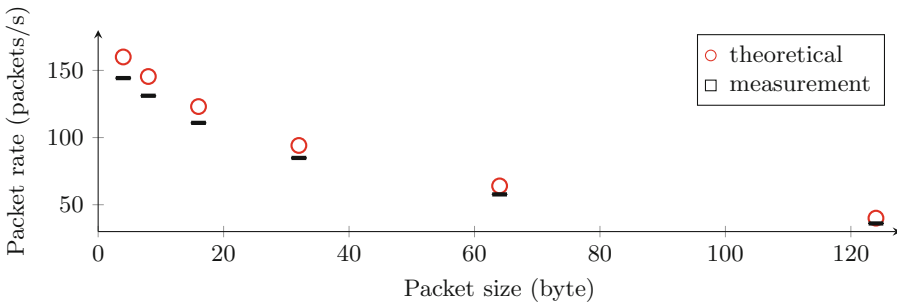


Fig. 9. Throughput of the PCAP firmware implementation

5.4 Hardware Allocation

The allocation process can be evaluated by measuring the algorithm run time in dependance of the scenario parameters (especially the number of nodes and the considered network topology). First of all, comprehensive and allocatable test scenarios need to be designed again. For the current panel hardware architecture the corresponding MILP model incorporates 378 constraints modeled by a total of 756 variables. Initial measurements of the allocation algorithm show a disproportionate increase in the iterations and calculation time with the number of scenario constraints to get optimal solutions from the MILP solver (see Fig. 10 – optimal). For the first steps of the allocation procedure we are not interested in optimal solutions but only in whether a model is feasible or not. To find a feasible solution for small scale test setups the solver in our simplified test cases needs less than 1000 iterations (see Fig. 10 – feasible). While running on an Ubuntu Linux (64 bit) virtual machine with 7 vCPUs (Intel Xeon X3470 Quad Core @ 2.93 GHz), the solver is able to calculate a solution whether a model is feasible or not in less than 0.5 s.

⁷ Texas Instruments (TI) CC2531: <http://www.ti.com/product/CC2531>.

⁸ SmartRF Protocol Packet Sniffer <http://www.ti.com/tool/PACKET-SNIFFER>.

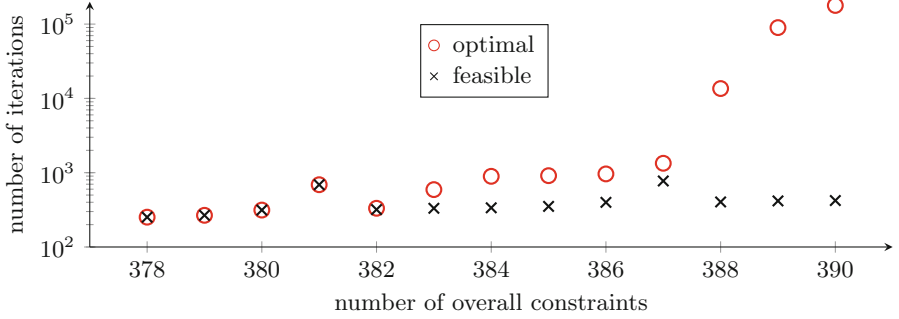


Fig. 10. Number of iterations for solving the allocation model

An overall evaluation of the system depends, in particular, on well defined emulation scenarios besides the interaction between all components. Furthermore, evaluation systems should always be additionally evaluated in an application-specific manner to determine which application types and which specific device topologies are meaningful to evaluate in general.

6 Summary and Outlook

WSN usage is spreading into all kinds of application domains over the last years. Applications and protocol stacks get more complex and thus more difficult to test and to evaluate. There is a growing demand in the simulation community to include hardware and RF environment-related details into network protocol simulation to facilitate the validation of simulative investigations and increase the realism of the abstracted representations of wireless channel characteristics.

In this work, we presented an approach to extend pure WSN protocol simulation through a Radio-in-the-Loop concept. We apply radio channel emulation with real sensor node hardware on the PHY layer to constitute a more precise behavior of WSN use cases. We focus on OMNeT++ and RoSeNet in our prototypical implementations and tests. However, the proposed ideas are not limited to a specific emulation system, hardware type, or network simulation environment. The support for wireless channel emulation using real hardware within network simulations will help to improve examinations of the interactions of protocol data flow and, for example, node energy consumption in reproducible radio conditions. Ultimately, the objective of our approach is to introduce a new tool chain for performance evaluation and cross-layer optimization in WSNs. This requires further evaluations and improvements which we discuss next.

In addition to pure MAC frame transmissions, the 802.15.4 simulation model needs to control the RF parameters of hardware-based radio transmissions, for example: the data rate, the modulation, the transmission power, the frequency, and the transmission channel. We use a TRX firmware (cp. Sect. 4.3) that sets these RF parameters inside the source code at compile-time for initial testing. To increase the overall model accuracy we plan to adjust the TRX firmware to enable

the processing of IEEE 802.15.4 PHY data and service primitives which are already implemented in our simulation model. A more modular attempt could be to exchange those parameters in optional extension headers of the PCAPNG file format or in so-called *Radiotap*⁹ headers (like they exist for IEEE 802.11 WLAN) and implement *setter* and *getter* methods for these parameters at the target hardware. This would allow the simulation to execute all procedures and features defined in the IEEE 802.15.4 standard and provide valid RF measurements for the simulated MAC layer.

Since most of the commercially available IEEE 802.15.4 transceivers only implement a subset of the complete PHY specification in practice, we also consider the use of flexible and reconfigurable Software Defined Radio (SDR) modules as RIL gateways instead of standard-conform but functionally limited transceivers. This approach could also help to avoid the data rate bottleneck of the UART interface of the transceiver modules. This architectural change opens up interesting possibilities for rapid prototyping of cross-layer communication approaches for future transceiver chip design.

Emulation of node mobility is another practical use case for test setups. In order to model mobility or node movement patterns (e.g., a receiver node moves out of the reception range of the transmitter node) within the RIL channel emulation, we have to vary hardware attenuation values during the emulation run-time. There are still open questions regarding the limits of the given RoSeNet system architecture for the emulation of mobility due to the complexity and the radio topology dependency of the hardware allocation process as it was discussed in Sect. 4.4.

Acknowledgement. Parts of this work were funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) through the Central Innovation Programme (ZIM initiative) under Contract No. ZF4119201ED5.

References

1. Imran, M., Said, A., Hasbullah, H.: A survey of simulators, emulators and testbeds for wireless sensor networks. In: International Symposium in Information Technology (ITSim). IEEE (2010). <https://doi.org/10.1109/ITSIM.2010.5561571>
2. Kropff, M., Krop, T., Hollick, M., Mogre, P.S., Steinmetz, R.: A survey on real world and emulation testbeds for mobile ad hoc networks. In: Proceedings of the 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM), 6-pp. IEEE (2006). <https://doi.org/10.1109/tridnt.2006.1649182>
3. Göktürk, E.: Emulating ad hoc networks: differences from simulations and emulation specific problems. In: Tugcu, T., Gelenbe, E., Caglayan, M.U. (eds.) *New Trends in Computer Networks*. Advances in Computer Science and Engineering: Reports, vol. 1. Imperial College Press, October 2005
4. Wehrle, K., Güneş, M., Gross, J.: *Modeling and Tools for Network Simulation*, 1st edn. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-12331-3>

⁹ Radiotap project website: <https://www.radiotap.org/>.

5. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools), ICST, article no. 60 (2008)
6. Riley, G.F., Henderson, T.R.: The NS-3 network simulator. [4] Chapter 2, pp. 15–34. <https://doi.org/10.1007/978-3-642-12331-3>
7. OMNeT++ Manual: Function called for each event (2017). <https://omnetpp.org/doc/omnetpp/manual/#sec:simple-modules:handlemessage:overview>
8. Stehlik, M.: Comparison of simulators for wireless sensor networks. M.Sc. thesis, Faculty of Informatics, Masaryk University (2011)
9. Goldsmith, A.: Wireless Communications. Cambridge University Press, Cambridge (2005)
10. Dwivedi, A.K., Vyas, O.P.: An exploratory study of experimental tools for wireless sensor networks. *Wirel. Sens. Netw.* **3**(7), 215–240 (2011)
11. Duan, S., Wan, Y., Meng, P., Wang, Q.: Hardware-in-the-loop and parallel simulation architecture for WSN. *TELKOMNIKA* **11**(1), 103–114 (2013)
12. Mozumdar, M.M.R., Lavagno, L., Vanzago, L., Sangiovanni-Vincentelli, A.L.: HILAC: a framework for hardware in the loop simulation and multi-platform automatic code generation of WSN applications. In: International Symposium on Industrial Embedded Systems (SIES), pp. 88–97. IEEE (2010)
13. Papadopoulos, G.Z., Kritsis, K., Gallais, A., Chatzimisios, P., Noel, T.: Performance evaluation methods in ad hoc and wireless sensor networks: a literature study. *IEEE Commun. Mag.* **54**(1), 122–128 (2016)
14. Österlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN), pp. 641–648. IEEE Computer Society (2006). <https://doi.org/10.1109/LCN.2006.322172>
15. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the 29th Annual IEEE Conference on Local Computer Networks (LCN), pp. 455–462. IEEE Computer Society (2004). <https://doi.org/10.1109/LCN.2004.38>
16. Sundani, H., Li, H., Devabhaktuni, V.K., Alam, M., Bhattacharya, P.: Wireless sensor network simulators a survey and comparisons. *Int. J. Comput. Netw. (IJCN)* **2**(5), 249–265 (2011)
17. Du, W., Mieleveville, F., Navarro, D., O'Connor, I., Carrel, L.: Modeling and simulation of networked low-power embedded systems: a taxonomy. *EURASIP J. Wirel. Commun. Netw.* **2014**(1), 1–12 (2014)
18. Wehner, P., Göhringer, D.: Internet of Things simulation using OMNeT++ and hardware in the loop. In: Keramidas, G., Voros, N., Hübner, M. (eds.) *Components and Services for IoT Platforms*, pp. 77–87. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-42304-3_4
19. Kirsche, M., Schnurbusch, M.: A new IEEE 802.15.4 simulation model for OMNeT++/INET. In: Proceedings of the 1st International OMNeT++ Community Summit (OMNeT 2014), September 2014
20. IEEE Standards Association: Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). IEEE Standards Document - Revision of IEEE Std. 802.15.4TM-2006. IEEE, September 2006. <https://doi.org/10.1109/IEEESTD.2006.232110>
21. Kirsche, M.: Selected System Models - IEEE 802.15.4. [4] Chapter 12.3, pp. 276–303. <https://doi.org/10.1007/978-3-642-12331-3>

22. Böhm, S., Kirsche, M.: Looking into hardware-in-the-loop coupling of OMNeT++ and RoSeNet. In: Proceedings of the 2nd International OMNeT++ Community Summit (OMNeT 2015), September 2015
23. Tüxen, M., Rüngeler, I., Rathgeb, E.P.: Interface connecting the INET simulation framework with the real world. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques (SIMUTools 2008), ICST, Brussels, Belgium, Belgium, pp. 40:1–40:6. ICST (2008). <https://doi.org/10.1145/1416222.1416267>



Assessing Simulated Software Graphs Using Conditional Random Fields

Marlon Welter^(✉), Daniel Honsel, Verena Herbold, Andre Staedtler,
Jens Grabowski, and Stephan Waack

Institute of Computer Science, Georg-August-Universität Göttingen,
Goldschmidtstrasse 7, 37077 Göttingen, Germany
marlon.welter@informatik.uni-goettingen.de

Abstract. In the field of software evolution, simulating the software development process is an important tool to understand the reasons why some projects fail, yet others prosper. For each simulation however, there is a need to have an assessment of the simulation results. We use Conditional Random Fields, specifically a variant form based on the Ising model from theoretical physics, to assess software graph quality. Our CRF-based assessment model works on so called Software Graphs, where each node of that graph represents a software entity of the software project. The edges are determined by immediate dependencies between the pieces of software underlying the involved nodes.

Because there is a lack of reference training data for our kind of evaluation, we engineered a special training paradigm that we call the Parsimonious Homogeneity Training. This training is not dependent on reference data. Instead of that it is designed to produce the following two effects. First, homogenizing the assessment of highly interconnected regions of the software graph, Second, leaving the assessment of these regions in relative independence from one another.

The results presented demonstrate, that our assessment approach works.

Keywords: Simulating software graphs · Conditional random fields
Parsimonious homogeneity training

1 Introduction

In our previous work we devised and implemented agent-based models that simulate the evolution of software projects [4, 5]. The aim is to monitor the software quality by assessing it at every stage. In this paper we propose a method to assess the quality of simulated software projects, which can also be used to assess the current state of a real software project and predict its likely progression in the

future. The model that we created was based on the following two modelling goals:

- M_1 : **Abstraction:** The results should be comprehensible on a high level such that a project manager can easily understand them. This is not given by a simple listing of quality metrics for each file in a program.
- M_2 : **Dependencies:** The Assessment is based on the local quality of files/classes, but it should also account for dependencies between files. A bug in a key class many other classes depend upon is much worse than a bug in singular disconnected class.

Our method operates on a *software graph* at a point in time of an evolving software project. The software graph results either from a software project simulation or it is mined from a software project repository, as the case may be. The nodes of our software graphs represent software entities like classes or files. The edges represent logical dependencies between software entities. We account for dependencies between software entities that actually impact the functioning of these entities itself. Many standard approaches, that just use file metrics to assess software quality cannot account for these kind of dependencies.

In our model, each node has a *local quality label* out of {acceptable, problematic} assigned to it, which reflects the quality of the code of this class/file. In our simulation the local label is determined solely by the number of bugs of the underlying piece of software. Our method then provides a high level quality assessment, where another *final quality label* again out of {acceptable, problematic} is assigned to every node. The final quality label of a node depends on the final quality labels of those nodes adjacent to it as well as on its own local quality label. This leads to a homogenization of the final quality labels compared with the local quality labels. Entire regions of the software graph can be identified as either acceptable or problematic instead of doing so for individual software entities.

We think that such a high level assessment is more useful for project monitoring than just information about single software entities. Also note, that we use the same labels {acceptable, problematic} for the local label as well as the final label of a node. The local label of a node represents the quality of that specific piece of code, whereas the final label of a node represents its functioning and maintainability in the program including other files that influence the one in focus. If, for example, a class depends on another error-prone class, the work of that class is also flawed.

To this end, we address our assessment problem as a node classification problem for software graphs using the probabilistic data model class of conditional random fields (CRFs). For our purposes the Ising model [3] as described in Sect. 2 is particularly appropriate.

CRFs are commonly used in bioinformatics to predict protein-protein interaction sites [1, 11]. Other fields where CRFs were applied successfully are shallow parsing [7], image labeling [8], gesture recognition [12] as well as entity recognition [9]. The common ground for these application fields is the fact, that the

labeling of the neighbourhood of a node is a strong indicator of its own label. The same is true for software graphs: The function of a bug-free class that resorts to a problematic class is disturbed.

To our best knowledge, we are the first who apply CRFs in the context of software system quality assurance. We use a heuristic described in Dong et al. [2] to approximate a classification of maximum posterior probability.

It would probably be more precise to take a larger label set than {problematic, acceptable} into account, because there is often a big difference between a slightly bugged file and a very bugged file. However, we think that it is reasonable to master this simpler model first before going on to anything fancier.

In this paper we will present a new training paradigm that allows us to determine the parameters for our Ising model. A standard machine learning approach using supervised learning techniques is not tractable in this case, since there does not exist reference data for software projects, where all software entities are labelled problematic/acceptable in terms of their functioning. Note that this is not only dependent on the software entity itself, but might also be influenced by other software entities that it relies upon. Instead we focus on a training that will take the local quality and dependencies as a basis and provide the user with a reasonable degree of generalization through homogenization.

It is crucial that the software graphs are of the following kind. The set of nodes is always decomposed into highly interconnected so-called *communities* representing logical parts of the software project. The number of edges between communities, however, is much smaller. Now we can describe the two goals of our training:

- Θ_1 : **Homogenization** of regions (communities) in the software graph.
- Θ_2 : **Independence** of regions from one another.

Reverting to the modeling goals M_1 and M_2 , the training goals are designed to meet M_1 : abstraction. The homogenization of regions Θ_1 will enable a higher level assessment, since entire communities can be assessed as either acceptable or problematic. This enables an assessment on the level of communities of the software graph. The force driving the homogenization is exerted through the edges in the CRF model. The second goal Θ_2 is needed because we do not want to end up in a complete homogenisation of the entire graph. The homogenization force has to be weak enough such that sparsely connected parts of the software (different communities) do not impact each other too much. Our method therefore provides the transition from an entity-based assessment to a community-based assessment. We call this approach the Parsimonious Community Heterogeneity Training, which will be closely described in Sect. 2.

The paper is structured as follows. In Sect. 2 we describe in detail how we model software-graph-based quality assessment of software projects using conditional random fields. To this end, Subsect. 2.1 is dedicated to the Ising Model [3] applied to software graphs, whereas Subsect. 2.2 is about the objective function of our Parsimonious Community Homogeneity (PCT) Training, which in turn we need to set the parameters of the Ising model. In Subsect. 2.3 the MAP labeling

computation for our prediction is explained. In Sect. 3 we present the full PCH training procedure. Moreover, we display some graphics of our results. Finally, in Sect. 4 the method used, the results obtained and future work planned are discussed.

2 Evaluating Software Projects by Means of CRFs

The software project is modelled by a *software graph* $G = (V, E)$. Each node $v \in V$ represents a software entity (e.g. class or file) of the project under study. Two nodes are connected by an edge $e \in E$ if and only if they have been changed in a small changeset together at least twice. That is, our software graphs are *change coupling graphs* as described in [4]. We assume that joint appearance in small changesets is a good indicator for dependencies between software entities.

Each node (software entity) has a *local quality label* that is either *acceptable* represented by $+1$ or *problematic* represented by -1 . These labels are assigned by the simulation process and are dependent on the occurrence and severity of bugs and bug fixing on software entities. Once a certain threshold of weighted bugs on an entity is reached, its local label is set to problematic (see [4]), otherwise it is acceptable. Our method then uses these local quality labels, which are formally given by a function

$$\mathbf{x} : V \rightarrow \{+1, -1\},$$

and the information about the entity dependencies from the software graph to produce a new *final classification* for each entity denoted by *acceptable* ($+1$) and *problematic* (-1) as well. The final labeling is again a mapping

$$\mathbf{y} : V \rightarrow \{+1, -1\}.$$

The set of nodes V of all software graphs occurring in this paper consists of highly connected parts which we denote as *communities*. In our model these communities represent logical parts of the software project (e.g. the database connection or the user management). Homogenization of the final entity classifications in communities is our strategy to achieve a high level assessment of the quality of the software graph, where this homogenization is subject to the constraint that the dependencies between communities shall be as small as possible.

To this end, we use a graphical probabilistic model similar to the Ising Model from theoretical physics (see [3]). The Ising model is used in ferro magnetics. We chose it because this model has the potential to homogenize the classifications of connected regions in the graph, which helps us reach our modelling goal M_1 .

The variant form of the Ising model used in this paper is described in Subsect. 2.1. It assigns a probability $p(\mathbf{y} | \mathbf{x})$ to every final labeling \mathbf{y} given the local labeling \mathbf{x} . It is fully determined by two parameters. Setting these two parameters, however, has a particular intrinsic difficulty, since there does not exist labelled training data for supervised learning. Our solution of that

problem is given in Subsect. 2.2. To compute a final classification given a local one, we invoke the Maximum A Posteriori (MAP) paradigm:

$$\mathbf{y}^* \leftarrow \operatorname{argmax}_{\mathbf{y} \in \{+1, -1\}^V} p(\mathbf{y} | \mathbf{x}).$$

Details can be found in Subsect. 2.3.

2.1 The Ising Model

Let $G = (V, E)$ be a software graph, and let $\mathbf{x} : V \rightarrow \{-1, +1\}$ be an entity-wise quality labeling. We study the following *Conditional Random Field (CRF)*:

$$p(\mathbf{y} | \mathbf{x}) := \frac{1}{Z(\mathbf{x})} \exp \left(\sum_{v \in V} h \cdot \mathbf{y}_v \mathbf{x}_v + \sum_{\{v, v'\} \in E} J \cdot \mathbf{y}_v \mathbf{y}_{v'} \right), \quad (1)$$

where

$$Z(\mathbf{x}) := \sum_{\mathbf{y} \in \{+1, -1\}^V} p(\mathbf{y} | \mathbf{x}).$$

The set of all mappings from V to $\{+1, -1\}$ is denoted by $\{+1, -1\}^V$.

According to Eq. 1, our CRF has the two parameters correlation and conformity that determine the final classification.

- Conformity $h > 0$ rewards conformance of \mathbf{y}_v with \mathbf{x}_v . That way it puts pressure to classify an entity in the final classification equal to its local quality label,
- Correlation $J > 0$ smoothes the final quality labeling compared to the local one; It determines the pressure that is exerted through edges to classify neighbouring entities with the same final label.

These two parameters are working in opposite directions and setting them differently will result in fundamentally different final labelings. We want to set these two parameters in a way such that the two training goals Θ_1 and Θ_2 , described in the introduction, will be achieved. This is done by our PCH training, which is necessary because of the fact that there is no labelled training data.

2.2 Parsimonious Community Homogeneity (PCH) Training

We solved the problem of not having labelled training data by developing and implementing a training paradigm which we refer to as PCH training. This training determines the Ising parameters of our model considering the training goals mentioned earlier. Knowing that our software graphs consist of communities of nodes, the two goals of the training can be described as follows:

- Θ_1 : **Homogenization** of communities in the software graph. The final labelling of all nodes of a community shall be mostly the same.

- Θ_2 : **Independence** of communities from one another. The final labellings of the nodes in one community shall have little impact on the final labelling of nodes in another community.

Once we have broadly homogeneous classifications of communities, a comprehensible visualisation of the state of the software can be reached. At that point an entire community can be reduced to the information that it is mainly acceptable or mainly problematic and thus the complexity of the assessment decreases dramatically. The second goal is the counterpart to the first, since we would end up in complete homogenisation of the graph if we only had Θ_1 as a goal. The parsimonious homogeneity reflects our intuition that we want to homogenise parts of the graph for clarity, yet not homogenize the classification of the entire graph, as in that case we would lose too much information. We assume that problems in some part of the software (e.g. the data visualisation) does only have a small impact on other parts of the software (e.g. the database connection).

For the creation of the graphs we use the stochastic block model, so we have control of the number of communities and the level of connection between those communities. We base the total size and number of communities in the graph on real software projects described in [5] and model small software projects in the case of this paper, to master the presumably easier case first.

Let $G = (V, E)$ be a software graph, and let $\mathbf{x} : V \rightarrow \{-1, +1\}$ be an entity-wise local quality labelling. Let $\overline{G} = (V, \overline{E})$ be the graph obtained by removing all inter-community edges from the edge set E of G . We call such a graph a *community-disconnected* graph. Finally, let $\overline{\mathbf{p}}(\mathbf{y} | \mathbf{x})$ be the conditional distribution defined by Eq. 1 when taking graph \overline{G} instead of graph G .

We have a natural mapping from nodes to communities by the generative nature of the stochastic block model. Let

$$\mathbf{y}^* \leftarrow \operatorname{argmax}_{\mathbf{y} \in \{+1, -1\}^V} \mathbf{p}(\mathbf{y} | \mathbf{x}).$$

be the MAP final classification, and let

$$\overline{\mathbf{y}}^* \leftarrow \operatorname{argmax}_{\mathbf{y} \in \{+1, -1\}^V} \overline{\mathbf{p}}(\mathbf{y} | \mathbf{x}).$$

the MAP final classification for \overline{G} . The target function \mathbf{t} of our training is defined as the sum of two values:

$$\mathbf{t}(G, \overline{G}) := h_c + i_c,$$

where

$$h_c := \frac{|\sum_{v \in V} \mathbf{y}^*(v)|}{|V|},$$

is the *average community homogeneity* and

$$i_c := 1 - \frac{\sum_{v \in V} |\overline{\mathbf{y}}^*(v) - \mathbf{y}^*(v)|}{2 \cdot |V|},$$

is the *average community independence*.

The average community homogeneity is the ratio of the prevalent classification in a community and is 1 in the case of total homogenization, whereas i_c is defined as the ratio of unchanged classifications between \mathbf{y}^* and $\bar{\mathbf{y}}^*$. The average community independence reaches its maximum when the MAP final classification \mathbf{y}^* is exactly the same as the MAP final classification $\bar{\mathbf{y}}^*$ on the community-disconnected graph, which means that the influence of the final labelling of one community on the final labelling of another community is negligible and does not change the MAP final classification and we thus call it community-independence.

2.3 MAP Prediction

The basis of the generation of the MAP final labeling \mathbf{y} given the local labeling \mathbf{x} , on the basis of Eq. 1 is the computation of a node score for every node, and the computation of an edge score for every edge of the software graph $G = (V, E)$. Since the local labeling \mathbf{x} is fixed, the normalization factor $Z(\mathbf{x})$ does not matter in determining the MAP prediction. Thus we work with the term $Z(\mathbf{x})p(\mathbf{y} | \mathbf{x})$ (note that $Z(\mathbf{x})$ gets cancelled out here). Furthermore, instead of maximizing the term $Z(\mathbf{x})p(\mathbf{y} | \mathbf{x})$, it suffices to work with its logarithm, which is

$$\sum_{v \in V} h \cdot \mathbf{y}_v \mathbf{x}_v + \sum_{\{v, v'\} \in E} J \cdot \mathbf{y}_v \mathbf{y}_{v'},$$

since that gives us the same order of final labeling when ordered by their scores. We use the logarithm here, because otherwise the resulting scores will be too huge in some graphs and cannot be stored any more in the standard types of the programming languages. Then for every node $v \in V$ and every edge $e = (v, v') \in E$, the node score of v equals h if $\mathbf{x}(v)$ and $\mathbf{y}(v)$ are equal, and $-h$ otherwise. The edge score for each e is equal to J , if $\mathbf{y}(v)$ and $\mathbf{y}(v')$ are equal, and $-J$ otherwise.

With this initialisation we use the Viterbi algorithm [10], a dynamic programming scheme to find the maximum labeling for the graph. The MAP prediction for CRFs over general graphs, however, is NP-hard. In our case this means that the Viterbi algorithm is exponential. To compute an approximate solution, we applied a Viterbi-based heuristics we devised in 2014 [2].

3 Experiments and Results

3.1 Training Philosophy

As noted before, the assessment of software entities into acceptable and problematic is a subjective one and thus no reference data for a supervised learning approach exists. We overcome this issue by leaving the definition of acceptable and problematic to the user and focus on the two goals defined in Sect. 2. The first use-case for this training will be the assessment on software graphs produced by the simulation process of our group. In this process the local quality label depends on the number and severity of bugs in a specific software entity. The parameter set, that is being produced in this training, will be judged on its capability to achieve the two goals of the PCH training.

3.2 Data Creation

We used a 10×10 -grid with values ranging from 0 to 1 in 0.1 steps, where the first dimension represents the conformity value and the second dimension represents the correlation value. For each of these parameter combinations we created 100 random graphs following the stochastic block model for random graphs [6]. This model seemed to be the simplest generative random graph model that created a structure based on communities. The following parameters were used:

- $|V|$ (number of nodes) = 100
- $|C|$ (number of categories) = 4
- p_{intra} : (edge probability in the same community) = 0.15
- p_{inter} : (edge probability between different communities) = 0.01

These parameters were used to approximate the structure of small real software graphs regarding size, number of communities and connectivity degree. The analysis of real software graphs for reference is described in [4, 5]. For each graph, the local quality labels (acceptable or problematic) were assigned using the method described in Algorithm 1:

Algorithm 1. Local Quality Label Generation

```

Let  $G = (V, E)$  be a graph and  $C$  be the set of communities in that graph.
for all communities  $\gamma \in C$  do
  Draw a random number  $p$ .
  for all nodes  $n \in \gamma$  do
    Draw a random number  $p_n$ .
    if  $p_n \leq p$  then
      Assign label +1 to  $n$ .
    else
      Assign label -1 to  $n$ .
    end if
  end for
end for

```

The random number p , that is drawn for each community, can be seen as a quality estimator for the entire community. This method ensures that we create a broad spectrum of communities regarding their average local quality. This reflects our intuition, that the source code quality can vary significantly among parts of project, depending on who wrote that code, what were the circumstances under which it was written and how complex is its functionality.

In the next step we computed the average homogeneity h_c and the average community independence i_c as described in the PCH Sect. 2.2. Finally, we chose the set of parameters that had the maximum sum of h_c and i_c as the output of our training. With this procedure, the training resulted in a conformity value $h = 0.5$ and a correlation value $J = 0.4$ that achieved an average community homogeneity $h_c = 0.8666$ and an average community independence $i_c = 0.9323$.

3.3 Evaluation of the Training Results

We used the obtained parameters to create classifications on a new set of software graphs created by the stochastic block model. This software graph set consisted of 100 graphs with 3–7 communities assigned randomly and 20–30 nodes per community. This evaluation phase was done to have a visual feedback of the results of our training on several kinds of graphs created by the stochastic block model. Two examples are presented in Fig. 1. The homogenisation process can be observed and the communities or parts of them get a clearer identity as either *overall problematic* or *overall acceptable*. Please note, that a simple threshold for the number of problematic files in a community would not lead to the same results, since that would not account for the dependencies between the files.

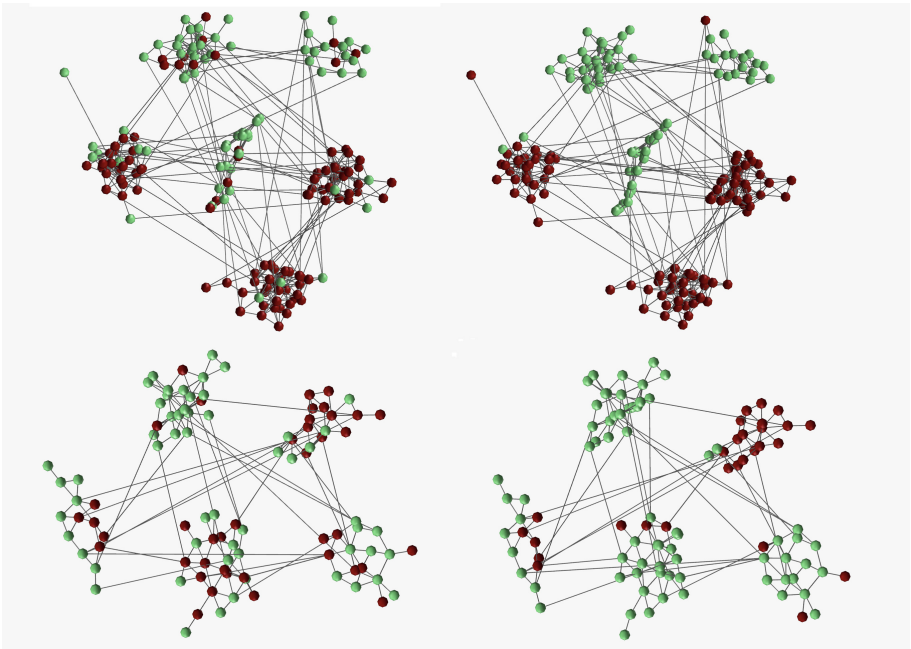


Fig. 1. Example of classifications of software graphs, where each node represents a software entity and each edge represents a dependency between two entities. The colour light green represents the label acceptable, the colour dark red represents the label problematic. The two graphs on the left side are coloured according to their local quality label. On the right side, the same two graphs are coloured according to their final classification using the parameters from our PCH training and the inference heuristic described in Dong et al. [2] to find the MAP labelling. (Color figure online)

4 Discussion

With the parameters achieved in Sect. 3, our method provides on average a broadly homogeneous classification inside communities. This is a helpful tool in judging the overall quality state of the software, since e.g. a manager, who wants to assess the state of some software project, does not need to know about the quality of every individual file. Broadly homogeneous communities provide a solid basis for such an assessment, because trends become obvious on a larger scale.

Why was the development of a new training paradigm (PCH) necessary? It was not possible to use a standard approach from machine learning, since there was no reference data for software graph quality. This implies that we could not use standard evaluation methods for parameter sets of our model, i.e. evaluation on the reference data. so we developed a new target function that would result in classifications that are easier to comprehend for human observers (community homogeneity), but not overly simplistic by homogenising the entire graph to acceptable or problematic (community independence). An advantage of this approach is in our opinion that the fine tuning (at what point will a community switch from a broadly acceptable to a broadly problematic state?) is determined by the definition of the local quality labels and therefore up to the user and her specific problem.

So far, the reference basis about the structure of software graphs is based on experience and a small sample of real software projects. Further work is needed to determine whether all software graphs can be satisfyingly modelled using a stochastic block model.

Another point for improvement might be the use of more than two local quality labels (acceptable/problematic) for each software entity. There might be a big difference for overall software quality if some entity is “a bit problematic” or “very problematic”. Our approach would generally be expendable to account for more local quality labels, but for now we decided to test the most simple modelling case first.

Some software metrics are related to code quality and in some projects it can be obvious that a project is failed, but in a general case it is very difficult to quantify the software quality state specifically. Our approach here is to rely on general guidelines (empower local quality trends to achieve higher homogeneity) and leave some tuning to the user (definition of the local quality labels).

In future research we plan to use these training results for the assessment of simulated software graphs created by Honsel et al. [4] where, so far, parameters have been manually set by the researchers.

Our second future plan is to reiterate the entire training process, but this time - instead of using artificial graphs created by the stochastic block model, we will use software graphs that were created by our simulation. In that case, the labelling will still vary, but the graph structure will be the same for all training graphs.

Acknowledgment. The authors thank the SWZ Clausthal-Göttingen (<https://www.simzentrum.de/en/>) that partially funded our work (both the former projects “Simulation-based Quality Assurance for Software Systems” and “DeSim”, and the recent project “Agent-based simulation models in support of monitoring the quality of software projects”).


References

1. Li, M.-H., Lin, L., Wang, X.-L., Liu, T.: Protein-protein interaction site prediction based on conditional random fields. *Bioinformatics* **23**(5), 597–604 (2007)
2. Dong, Z., et al.: CRF-based models of protein surfaces improve protein-protein interaction site predictions. *BMC Bioinform.* **15**, 277 (2014). <https://doi.org/10.1186/1471-2105-15-277>
3. Ising, E.: Beitrag zur Theorie des Ferromagnetismus. *Z. Phys.* **31**(1), 253–258 (1925)
4. Honsel, D., Honsel, V., Welter, M., Waack, S., Grabowski, J.: Monitoring software quality by means of simulation methods. In: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2016), Article 11, 6 pp. ACM, New York (2016). <https://doi.org/10.1145/2961111.2962617>
5. Honsel, V., Honsel, D., Herbold, S., Grabowski, J., Waack, S.: Mining software dependency networks for agent-based simulation of software evolution. In: ASE Workshop (2015)
6. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983). ISSN: 0378–8733. [https://doi.org/10.1016/0378-8733\(83\)90021-7](https://doi.org/10.1016/0378-8733(83)90021-7). (<http://www.sciencedirect.com/science/article/pii/0378873383900217>)
7. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, (NAACL 2003), vol. 1, pp. 134–141. Association for Computational Linguistics, Stroudsburg (2003). <https://doi.org/10.3115/1073445.1073473>
8. He, X., Zemel, R.S., Carreira-Perpinan, M.A.: Multiscale conditional random fields for image labeling. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 2, pp. II-695–II-702 (2004). <https://doi.org/10.1109/CVPR.2004.1315232>
9. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4 (CONLL 2003), vol. 4, pp. 188–191. Association for Computational Linguistics, Stroudsburg (2003). <https://doi.org/10.3115/1119176.1119206>
10. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **13**, 260–269 (1967)

11. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of 18th International Conference on Machine Learning, pp. 282–289 (2001). citeseer.ist.psu.edu/lafferty01conditional.html
12. Wang, S.B., Quattoni, A., Morency, L.P., Demirdjian, D., Darrell, T.: Hidden conditional random fields for gesture recognition. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), pp. 1521–1527 (2006). <https://doi.org/10.1109/CVPR.2006.132>



Elephant Against Goliath: Performance of Big Data Versus High-Performance Computing DBSCAN Clustering Implementations

Helmut Neukirchen^(✉) 

Faculty of Industrial Engineering, Mechanical Engineering
and Computer Science, School of Engineering and Natural Sciences,
University of Iceland, Reykjavik, Iceland
`helmut@hi.is`

Abstract. Data is often mined using clustering algorithms such as *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN). However, clustering is computationally expensive and thus for big data, parallel processing is required. The two prevalent paradigms for parallel processing are *High-Performance Computing* (HPC) based on *Message Passing Interface* (MPI) or *Open Multi-Processing* (OpenMP) and the newer big data frameworks such as Apache Spark or Hadoop. This paper surveys for these two different paradigms publicly available implementations that aim at parallelizing DBSCAN and compares their performance. As a result, it is found that the big data implementations are not yet mature and in particular for skewed data, the implementation's decomposition of the input data into parallel tasks has a huge influence on the performance in terms of run-time due to load imbalance.

1 Introduction

Computationally intensive problems, such as simulations, require parallel processing. Some problems are embarrassingly parallel (such as the many but rather small problems [1] resulting from the *Large Hadron Collider* (LHC) experiments) – most computational problems in simulation are, however, tightly coupled (such as, e.g., finite element modelling which may even involve model coupling [2]). The standard approach in this case is *High-Performance Computing* (HPC).

Highly praised contenders for huge parallel processing problems are big data processing frameworks such as Apache Hadoop or Apache Spark. Hence, they might be considered an alternative to HPC for distributed simulations. We have already shown [3] that for a typical embarrassingly parallel LHC simulation, an Apache Hadoop-based distributed processing approach is almost on par with the standard distributed processing approach used at LHC.

In this paper, we want to use a more tightly coupled parallel processing problem to investigate whether HPC or big data platforms are better suited

for computationally intensive problems that involve some tight coupling: clustering using the *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) [4] clustering algorithm.

DBSCAN is computationally expensive and thus for clustering big data, parallel processing is required. However, the DBSCAN algorithm has been defined as a serial, non-parallel algorithm. Hence, several parallel variants of DBSCAN have been suggested. The main contribution of this paper is to investigate the runtime performance and scalability of different publicly available parallel DBSCAN implementations running either on HPC platforms or on big data platforms such as the MapReduce-based Apache Hadoop or the *Resilient Distributed Dataset* (RDD)-based Apache Spark.

The Bible's book of Samuel and Chap. 2 of the Qur'an contain the story of the giant warrior Goliath (Jalut in Arabic): HPC clusters can be considered as such old giants. The graphical logo of the first popular big data platform, Apache Hadoop [5], is an elephant. The question whether HPC or big data is better can therefore be compared to a fight between Goliath and an elephant. – But beware: later, we encounter even ogres!

The outline of this paper is as follows: subsequent to this introduction, we provide foundations on DBSCAN, HPC and big data. Afterwards, in Sect. 3, we describe as related work other comparison of algorithms running on HPC and big data platforms as well as a comparison of non-parallel implementations of DBSCAN. In Sect. 4, we survey existing DBSCAN implementations. Those implementations that were publicly available are evaluated with respect to their run-time in Sect. 5. We conclude with a summary and an outlook in Sect. 6. This full paper is based on an extended abstract [6] and a technical report [7]. An annex with command line details can be found in a EUDAT B2SHARE persistent electronic record [8].

2 Foundations

2.1 DBSCAN

The spatial clustering algorithm *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) [4] has the nice properties that the number of clusters needs not to be known in advance, but is rather automatically determined; that it is almost independent from the shape of the clusters; and that it can deal with and filter out noise. Basically, the underlying idea is that for each data point, the neighbourhood within a given *eps* radius has to contain at least *minpts* points to form a cluster, otherwise it is considered as noise.

In the simplest implementation, finding all points which are in the *eps* neighbourhood of the currently considered point, requires to check all remaining $n - 1$ points of the input data: doing this for each of the n input points leads to a complexity of $O(n^2)$. Using spatially sorted data structures for the *eps* neighbourhood search, such as R-trees [9], R*-trees [10], or kd-trees [11], reduces the overall

complexity to $O(n \log n)$ if *eps* is reasonably small¹. The standard algorithms to populate such spatially sorted data structures cannot run in parallel and require in particular to have the entire input data available in non-distributed memory.

Even if the problem of having a distributed, parallel-processing variant of populating and using a spatially sorted data structure (in order to bring the overall complexity down to $O(n \log n)$) is solved, there are further obstacles in parallelizing DBSCAN in a way that it scales optimally.

The actual clustering can be easily parallelized by partitioning the data spatially: typically, all points that belong to the same partition or cell (=a rectangle in case of 2 dimensional data, a cube in case of 3D data, or a hypercube for $n \geq 3$ dimensions) can be processed by one thread independently from other threads that process the remaining partitions of points. Only at the border of each rectangle/cube/hypercube, points from direct neighbour rectangles/cubes/hypercubes need to be considered up to a distance of *eps*. For this, the standard approach of ghost or halo regions [12] can be applied, meaning that these points from a neighbour partition need to be accessible as well by the current thread (in case of distributed memory, this requires to copy them into the memory of the respective thread). In a final step, those clusters determined locally in each partition which form a bigger cluster spanning multiple partitions need to be merged.

To achieve a maximum speed-up, not only an efficient spatially sorted data structure and low communication overhead (e.g. for halo regions or finally merging locally obtained clusters), but also the size of the partitions is crucial: the input domain needs to be decomposed so that each thread or processor core get an equal share of the work. The simple approach of dividing the input domain into spatially equally sized chunks (for example as many chunks as processor cores are available) yields imbalanced workloads for the different cores if the input data is skewed: some partitions may then be almost empty, others very crowded. For heavily skewed data, the spatial size of each partition needs to be rather adjusted, for example in a way that each partition contains an equal number of points or the same number of comparisons are performed in each partition. If ghost/halo regions are used, then also the number of points in these regions need to be considered, because they also need to be compared by a thread processing that partition.

As shown, parallelizing DBSCAN in a scalable way beyond a trivial number of parallel threads (or processing nodes respectively) or problem size is a challenge. For example, PDBSCAN [13] is a parallelized DBSCAN, however it involves a central master node to aggregate intermediate results which can be a bottleneck with respect to scalability. In particular, when processing “big data” (i.e. n is huge), the implementation with the standard complexity of $O(n^2)$ will be too slow and rather $O(n \log n)$ algorithms are needed.

¹ If the *eps* neighbourhood contains, e.g., all data points, the complexity of DBSCAN grows obviously again to $O(n^2)$ despite these tree-based approaches.

2.2 HPC

High-Performance Computing (HPC) is tailored to CPU-bound problems. Hence, special and rather expensive hardware is used, e.g. compute nodes containing fast CPUs including many cores and large amounts of RAM, very fast interconnects (e.g. InfiniBand) for communication between nodes, and a centralised *Storage-Area Network* (SAN) with high bandwidth due to a huge *Redundant Array of Independent Disks* (RAID) and fast attachment of them to the nodes.

To make use of the multiple cores per CPU, typically shared-memory multi-threading based on *Open Multi-Processing* (OpenMP) [14] is applied. To make use of the many nodes connected via the interconnects, an implementation of the *Message Passing Interface* (MPI) [15] is used. The underlying programming model (in particular of MPI) is rather low-level: the domain decomposition of the input data, all parallel processing, the synchronisation and communication needed for tight coupling has to be explicitly programmed. Typically rather low-level, but fast programming languages such as C, C++ and Fortran are used in the HPC domain. In addition to message passing, MPI supports parallel I/O to read different file sections from the SAN in parallel into the different nodes. To this aim, parallel file systems such as Lustre [16] or the *General Parallel File System* (GPFS) [17] provide a high aggregated storage bandwidth. Typically, binary file formats such as netCDF or the *Hierarchical Data Format* (HDF) [18] are used for storing input and output data in a structured way. They come with access libraries that are tailored to MPI parallel I/O.

While the low-level approach allows fast and tightly coupled implementations, their implementation takes considerable time. Furthermore, no fault tolerance is included: a single failure on one of the many cores will cause the whole HPC job to fail which then needs to be restarted from the beginning if no checkpointing has been implemented. However, due to the server-grade hardware, hardware failures are considered to occur seldom (but still, they occur).

2.3 Big Data

The big data paradigm is tailored to process huge amounts of data, however the actual computations to be performed on this data are often not that computationally intensive. Hence, cheap commodity hardware is sufficient for most applications. Being rather I/O-bound than CPU-bound, the focus is on *High-Throughput Computing* (HTC). To achieve high-throughput, locality of data storage is exploited by using distributed file systems storing locally on each node a part of the data. The big data approach aims at doing computations on those nodes where the data is locally available. By this means, slow network communication can be minimised. (Which is crucial, because rather slow Ethernet is used in comparison to the fast interconnects in HPC.)

An example distributed file system is the *Hadoop Distributed File System* (HDFS), introduced with one of the first open-source big data frameworks, Apache Hadoop [5] which is based on the MapReduce paradigm [19]. As it is

intended for huge amounts of data, the typical block size is 64 MB or 128 MB. Hadoop has the disadvantage that only the MapReduce paradigm is supported which restricts the possible class of parallel algorithms and in particular may lead to unnecessarily storing intermediate data on disk instead of allowing to keep it in fast RAM. This weakness is overcome by Apache Spark [20] which is based on *Resilient Distributed Datasets* (RDDs) [21] which are able to store a whole data set in RAM distributed in partitions over the nodes of a cluster. A new RDD can be obtained by applying transformations in parallel on all input RDD partitions. To achieve fault tolerance, an RDD can be reconstructed by re-playing transformations on those input RDD partitions that survived a failure. The initial RDD is obtained by reads from HDFS. While RDDs are kept in RAM, required data may not be available locally in the RDD partition of a node. In this case, it is necessary to re-distribute data between nodes. Such shuffle operations are expensive, because slow network transfers are needed for them. Typically, textual file formats, such as *Comma-Separated Values* (CSV) are used that can be easily split to form the partitions on the different nodes.

High-level, but (in comparison to C/C++ or Fortran) slower languages such as Java or the even more high-level Scala or Python are used in big data frameworks. Scala has over Python the advantage that it is compiled into Java bytecode and is thus natively executed by the *Java Virtual Machine* (JVM) running the Hadoop and Spark frameworks. While the code to be executed is written as a serial code, the big data frameworks take behind the scenes care that each node applies in parallel the same code to the different partitions of the data.

Because commodity hardware is used which is more error prone than HPC server-grade hardware, big data approaches need to anticipate failures as the norm and have thus built-in fault tolerance, such as redundant data storage or restarting failed jobs.

2.4 Convergence of HPC and Big Data

Convergence of HPC and big data approaches is taking place in both directions: typically either in form of *High-Performance Data Analysis* (HPDA), meaning that HPC is used in domains that used to be the realm of big data platforms, or big data platforms are used in the realm of HPC. Alternatively, a mixture is possible: big data platforms are deployed on HPC clusters, however, sacrificing data locality-aware scheduling [22]. In this paper, we investigate how mature this convergence is by comparing DBSCAN clustering implementations for HPC and for big data platforms.

3 Related Work

HPC and big data data implementations for the same algorithms have been studied before. Jha et al. [22] compare these two parallel processing paradigms in general and introduce “Big Data Ogres” which refer to different computing problem areas with clustering being one of them. In particular, they evaluate

and compare the performance of k -means clustering [23] implementations for MPI-based HPC, for MapReduce-based big data platforms such as Hadoop and HARP (which introduces MPI-like operations into Hadoop), and for the RDD-based Spark big data platform. In their evaluation, the considered HPC MPI k -means implementation outperforms the other more big data-related implementation with the implementation based on HARP being second fastest and the implementation for Spark ranking third.

A further performance comparison of a big data implementation and a traditional parallel implementation has been done by us [3] with respect to a typical embarrassingly parallel High Energy Physics analysis: as traditional embarrassingly parallel execution platform, the *Parallel ROOT Facility* (PROOF) [24] has been compared to using Apache Hadoop for this analysis: while the Hadoop-based implementation is slightly slower, it offers fault tolerance.

The influence of data storage locality as exploited by Spark and other big data platforms compared to centralized HPC SAN storage has been investigated by Wang et al. [25]. They use data intensive Grep search and compute intensive logistic regression as case study and come to the conclusion that even with a fast 47 GB/s bandwidth with centralized Lustre SAN, data locality matters for Grep and thus accesses to local SSDs are faster. However, for the logistic regression, locality of I/O does not matter.

Kriegel et al. [26] stresses the scientific value of benchmarking and evaluates in particular the run-time of serial, non-parallel implementations of DBSCAN. Concerning different programming languages, they show that a C++ implementation is one order of magnitude faster than a comparable Java implementation. They also observed a four orders of magnitude speed difference between different serial implementations of DBSCAN.

4 Survey of Parallel DBSCAN Implementations

This section contains a survey of the considered DBSCAN implementations. To be able to compare their run-time performance on the same hardware and using the same input, only open-source implementations have been considered.

For comparison, we also used also ELKI 0.7.1 [27], an *Environment for DeveLoping KDD-Applications supported by Index-Structures*. ELKI is an optimised serial open-source DBSCAN implementation in Java which employs R*-trees to achieve $O(n \log n)$ performance. By default, ELKI reads space- or comma-separated values and it supports arbitrary dimensions of the input data.

4.1 HPC DBSCAN Implementations

A couple of DBSCAN implementations for HPC platforms exist (as, for example, listed by Patwaryat et al. [28] or Götz et al. [29]). To our knowledge, only for two of them, the source is available: PDSDBSCAN and HPDBSCAN. Thus, we restrict in the following to these two. Both support arbitrary data dimensions.

PDSDBSCAN by Patwary et al. [28] (C++ source code available on request from the PDSDBSCAN first author [30]) makes use of parallelization either based on shared memory using OpenMP or based on distributed memory using MPI. For their OpenMP variant, the the input data needs to fit into the available RAM; for the MPI variant, a pre-processing step is required to partition the input data onto the distributed memory. Details of this pre-processing step are not documented as the authors do not consider this step as part of their algorithm and thus, it is neither parallelized nor taken into account when they measure their running times. Input data is read via the netCDF I/O library.

HPDBSCAN by Götz et al. [29] (C++ source code available from repository [31]) makes use of parallelization based on shared memory and/or distributed memory: besides a pure OpenMP and pure MPI mode, also a hybrid mode is supported. This is practically relevant, because an HPC cluster is a combination of both memory types (each node has RAM shared by multiple cores) and thus, a hybrid mode is most promising to achieve high performance. For the domain decomposition and to obtain a spatially sorted data structure with $O(\log n)$ access complexity, the arbitrary ordered input data is first indexed in a parallel way and then re-distributed so that each parallel processor has points in the local memory which belong to the same spatial partition. It is the only implementation that sizes the partitions using a cost function that is based on the number of comparisons (=number of pairs for which the distance function needs to be calculated) to be made for the resulting partition size: this obviously also includes points in adjacent ghost/halo regions. The command line version of the implementation reads the input data via the HDF I/O library.

4.2 Spark DBSCAN Implementations

Even though our search for Apache Spark big data implementations of DBSCAN² was restricted to JVM-based Java or Scala³ candidates, we found several parallel open-source⁴ implementations of DBSCAN: Spark DBSCAN,

² Remarkably, the machine learning library MLlib which is a part of Apache Spark does not contain DBSCAN implementations.

³ Note that also purely serial Scala implementations of DBSCAN are available, for example GSBSCAN from the Nak machine learning library [32]. However, these obviously make not use of Apache Spark parallel processing. But still, they can be used from within Apache Spark code to call these implementations in parallel, however each does then cluster unrelated data sets [33].

⁴ There is another promising DBSCAN implementation for Spark by Han et al. [34]: A kd-tree is used to obtain $O(n \log n)$ complexity. Concerning the partitioning, the authors state “We did not partition data points based on the neighborhood relationship in our work and that might cause workload to be unbalanced. So, in the future, we will consider partitioning the input data points before they are assigned to executors.” [34]. However, it was not possible to benchmark it as is not available as open-source.

RDD-DBSCAN, Spark_DBSCAN, and DBSCAN On Spark. They are described in the following.

Spark DBSCAN by Litouka (source code via repository [35]) is declared as experimental and being not well optimised. For the domain decomposition, the data set is considered initially as a large box full of points. This box is then along its longest dimension split into two parts containing approximately the same number of points. Each of these boxes is then split again recursively until the number of points in a box becomes less than or equal to a threshold, or a maximum number of levels is reached, or the shortest side of a box becomes smaller than $2\textit{eps}$ [35]. Each such a box becomes a record of an RDD which can be processed in parallel, thus yielding a time complexity of $O(m^2)$ for that parallel processing step with m being the number of points per box [36].

RDD-DBSCAN by Cordova and Moh [37] (source code via repository [38]) is loosely based on MR-DBSCAN [39]. Just as the above Spark DBSCAN by Litouka, the data space is split into boxes that contain roughly the same amount of data points until the number of points in a box becomes less than a threshold or the shortest side of a box becomes smaller than $2\textit{eps}$. R-trees are used to achieve an overall $O(n \log n)$ complexity [37].

Spark_DBSCAN (source code via repository [40]) is a very simple implementation (just 98 lines of Scala code) and was not considered any further, because of its complexity being $O(n^2)$ [36].

DBSCAN On Spark by Raad (source code via repository [41]) uses for domain decomposition a fixed grid independent from how many points are contained in each resulting grid cell. Furthermore, to reduce the complexity, no Euclidian distance function is used (which would be a circle with $2\textit{eps}$ diameter), but the square box grid cells (with $2\textit{eps}$ edge length) themselves are rather used to decide concerning neighbourhood (see function `findNeighbors` in [41]). So, while it is called “DBSCAN On Spark” it implements only an approximation of the DBSCAN algorithm and does in fact return wrong clustering results.

Common Features and Limitations of the Spark Implementations. All the considered implementations of DBSCAN for big data platforms assume the data to be in CSV or space-separated format.

All the Apache Spark DBSCAN implementations (except for the closed-source DBSCAN by Han et al. [34]) work only on 2D data: On the one hand, the partitioning schemes used for decomposition are based on rectangles instead of higher-dimensional hyper-cubes. On the other hand, for calculating the distance between points, most implementations use a hard-coded 2D-only implementation of calculating the Euclidian distance⁵.

⁵ Note that spheric distances of longitude/latitude points should in general not be calculated using Euclidian distance in the plane. However, as long as these points are sufficiently close together, clustering based on the simpler and faster to calculate Euclidian distance is considered as appropriate.

Most Spark-based implementations aim at load balancing by having an approximately equal number of data points in each partition and a partition may not get smaller than 2 *eps*.

4.3 MapReduce DBSCAN Implementations

For further comparison, it would have been interesting to evaluate in addition also MapReduce-based DBSCAN implementations for the Apache Hadoop platform; candidates found to be worthwhile (because they claim to be able to deal with skewed data) were MR-DBSCAN [39, 42] by He et al. and DBSCAN-MR [43] by Dai and Lin. However, none of these implementations were available as open-source and e-mail requests to the respective first authors to provide their implementations either as source code or as binary were not answered. Hence, it is impossible to validate the performance claims made by these authors.

5 Evaluation of Parallel DBSCAN Implementations

Typically, comparisons between HPC and big data implementations are difficult as the implementations run on different cluster hardware (HPC hardware versus commodity hardware) or cannot exploit underlying assumptions (such as missing local data storage when deploying big data frameworks at run-time on HPC clusters using a SAN).

5.1 Hardware and Software Configuration

In this paper, the same hardware is used for HPC and Spark runs: the cluster JUDGE at Jülich Supercomputing Centre. JUDGE was formerly used for HPC and has been turned into a big data cluster. It consists of IBM System x iDataPlex dx360 M3 compute nodes each comprising two Intel Xeon X5650 (Westmere) 6-core processors running at 2.66 GHz. For the big data evaluation, we were able to use 39 executor nodes, each having 12 cores or 24 virtual cores with hyper-threading enabled (=936 virtual cores) and 42 GB of usable RAM per node and local hard disk.

In the HPC configuration, a network-attached GPFS storage system, the Jülich Storage cluster JUST, was used to access data (measured peak performance of 160 GB/s), and the CPU nodes were connected via an Infiniband interconnect. The big data configuration relies on local storage provided on each node by a Western Digital WD2502ABYS-23B7A hard disk (with peak performance of 222.9 MB/s per disk, corresponding to 8.7 GB/s total bandwidth if all 39 nodes read their local disk in parallel). 200 GB on each disk were dedicated to HDFS using a replication factor of 2 and 128 MB HDFS block size. The CPU nodes were connected via Ethernet network connections.

The software setup in the HPC configuration was SUSE Linux SLES 11 with kernel version 2.6.32.59-0.7. The MPI distribution was MPICH2 in version

1.2.1p1. For accessing HDF5 files, the HDF group’s reference implementation version 1.8.14 was used. The compiler was gcc 4.9.2 using optimisation level O3.

The big data software configuration was deployed using the Cloudera CDH 5.8.0 distribution providing Apache Spark version 1.6.0 and Apache Hadoop (including HDFS and YARN which was used as resource manager) version 2.6.0 running on a 64-Bit Java 1.7.0_67 VM. The operating system was CentOS Linux release 7.2.1511.

5.2 Input Data

Instead of using artificial data, a real data set containing skewed data was used for evaluating the DBSCAN implementations: geo-tagged tweets from a rectangle around the United Kingdom and Ireland (including a corner of France) in the first week of June 2014. The data was obtained by Junjun Yin from the National Center for Supercomputing Application (NCSA) using the Twitter streaming API. This data set contains 3 704 351 longitude/latitude points and is available at the scientific data storage and sharing platform B2SHARE [44]. There, the data is contained in file `twitterSmall.h5.h5`. A bigger Twitter data set `twitter.h5.h5` from the same B2SHARE location covers whole of June 2014 containing of 16 602 137 data points, some of them are bogus artefacts though (Twitter spam) – still we used it to check whether implementations are able to cope with bigger data sets; a 3D point cloud data set for the city of Bremen is also provided there, however it was not usable for benchmarking the surveyed DBSCAN implementations for Spark which typically support only 2D data.

The original file of the small Twitter data set is in HDF5 format and 57 MB in size. To be readable by ELKI and the Spark DBSCAN implementations, it has been converted using the `h5dump` tool (available from the HDF group) into a 67 MB CSV version and into an 88 MB *Space-Separated Values* (SSV) version that contains in the first column an increasing integer number as point identifier (expected by some of the evaluated DBSCAN implementations). The size of these two data sets is summarised in Table 1.

Table 1. Size of the used Twitter data sets

| | Data points | HDF5 size | CSV size | SSV with Ids |
|---------------|-------------|-----------|----------|--------------|
| Twitter Small | 3 704 351 | 57 MB | 67 MB | 88 MB |
| Twitter Big | 16 602 137 | 254 MB | 289 MB | 390 MB |

For all runs, $eps = 0.01$ and $minpts = 40$ were used as parameters of DBSCAN. The detailed command line parameters can be found in EUDAT [8].

5.3 DBSCAN Implementation Versions

The dates of the used DBSCAN implementation source code versions and their repository are provided in Table 2. The source code was used unmodified except

Table 2. Used open source repository versions

| Implementation | Repository | Version date |
|-----------------|--|--------------|
| HPDBSCAN | bitbucket.org/markus.goetz/hpdbscan | 2015-09-10 |
| Spark DBSCAN | github.com/alitouka/spark_dbscan | 22 Feb 2015 |
| RDD DBSCAN | github.com/irvingc/dbscan-on-spark | 14 Jun 2016 |
| DBSCAN on Spark | github.com/mraad/dbscan-spark | 30 Jan 2016 |

for one change: by default, Spark makes each HDFS block of the input file a separate partition of the input RDD. With the above file sizes of the small Twitter data set being lower than the used HDFS block size of 128 MB, the initial RDD would contain just a single partition located on the node storing the corresponding HDFS block. In this case, no parallelism would be used to process the initial RDD. Therefore, if the Spark DBSCAN implementations did not anyway allow to specify the number of partitions to be used, the implementations were changed so that it is possible to specify the number of partitions to be used for the initial file read. While this means that non-local reads will occur, the overhead of these non-local reads is negligible in particular since it leads to a better degree of parallel processing.

5.4 Measurements

Comparing C++ implementations (PDSDBSCAN and HPDBSCAN) to JVM-based DBSCAN implementations for Spark is somewhat comparing apples and oranges. Hence, we used as a further comparison a Java implementation, the pure serial ELKI (see Sect. 4) with `-db.index "tree.metrical.covertree.SimplifiedCoverTree$Factory"` spatial indexing option running just on one of the cluster cores. The times were measured using the POSIX command `time`.

As usual in Spark, the Spark DBSCAN implementations create the output in parallel resulting in one file per parallel RDD output partition. If a single output file is intended, it can be merged afterwards, however this time is not included in the measurement. The reported times were taken from the “Elapsed” line of the application’s entry in the Spark web user interface for the completed run.

For the number of experiments that we did, we could not afford to re-run all of them multiple times to obtain averages or medians. However, for one scenario (RDD-DBSCAN, 233 executors, each using 4 cores with 912 initial partitions running on the small Twitter data set), we repeated execution four times. The observed run-times are shown in Table 3. For these 9–10 min jobs, deviations of

Table 3. Deviation of RDD-DBSCAN run-times for the same configuration

| Run | 1. | 2. | 3. | 4. |
|----------|-----|-----|-----|-----|
| Time (s) | 653 | 546 | 553 | 560 |

up to almost 2 min occurred. The fact that the first run was slower than the subsequent runs might be attributed to caching of the input file. In all of our experiments, we had exclusive use of the assigned cores.

Preparatory Measurements. In addition to the DBSCAN parameters *eps* and *minpts*, the parallel Spark-based implementations are influenced by a couple of parallelism parameters which were determined first as described below.

Spark uses the concepts of *executors* with a certain number of threads per executor process. A couple of sample measurements using a number of threads per executor ranging from 3 to 22 have been performed and the results ranging from 626 s to 775 s are within the above deviations, hence the influence of threads per executor is not considered significant. (Most of the following measurements have been made with 8 threads per executor – details can be found in EUDAT [8].)

Parallelism in Spark is influenced by the number of partitions into which an RDD is divided. Therefore, measurements with varying initial partition sizes have been made (in subsequent RDD transformations, the number of partitions may however change depending on the DBSCAN implementations). Measurement for RDD-DBSCAN running on the small Twitter data set on the 932 core cluster (not all cores were assigned to executors to leave cores available for cluster management) have been made for a number of initial number of input partitions ranging from 28 to 912. The observed run-times were between 622 s and 736 s which are all within the above deviation range. Hence, these experiments do not give observable evidence of an optimal number of input partitions. However, in the remainder, it is assumed that making use of the available cores already from the beginning is optimal and hence 912 was used as the initial number of input partitions.

After the input data has been read, the DBSCAN implementations aim at distributing the read data points based on spatial locality : as described in Sect. 4.2, most Spark DBSCAN implementations aim at recursively decomposing the input domain into spatial rectangles that contain approximately an equal number of data points and they stop doing so as soon as a rectangle contains only a certain number of points; however, a rectangle becomes never smaller than $2\textit{eps}$ edge length. Assuming that the subsequent clustering steps are also based on 912 partitions, the 3 704 351 points of the small Twitter data set divided by 912 partitions yield 4061 points per partition as a decomposition into an equal number of points. However, due to the fact a rectangle becomes never smaller than $2\textit{eps}$ edge length, some rectangles of that size still contain more points (e.g. in the dense-populated London area, some of these $2\textit{eps}$ rectangle contain up to 25 000 data points) and thus, the domain decomposition algorithm terminates with some rectangles containing a rather high number of points.

Experiments have been made with different numbers of points used as threshold for the domain decomposition by the Spark-based implementations. As shown in Table 4, a threshold of a minimum of 25 000 data points per rectangle promises fastest execution. This number is also the lowest number that avoids the domain

Table 4. Influence of number of points used in domain decomposition

| No. of points threshold | 4 061 | 9 000 | 20 000 | 25 000 | 50 000 |
|-------------------------|-------|-------|--------|--------|--------|
| Times (s) | 1 157 | 823 | 867 | 675 | 846 |

decomposition to terminate splitting rectangles because of reaching the 2ϵ s edge length limit: a naïve explanation would be that all rectangles contain an approximately equal number of points thus leading to load balancing. However, later findings (Sect. 5.5) show a significant load imbalance.

Run-Time Measurements on Small Data Set. After the above parameters have been determined, a comparison of the run-times of the different implementations was made when clustering the small Twitter data set while increasing the number of cores and keeping the problem size fixed (“strong scaling”).

Table 5 shows results using a lower number of cores in parallel. The C++ implementation HPDBSCAN (running in MPI only mode) performs best in all cases and scales well: even with just one core, only 114s are needed to cluster the small Twitter data set. Second in terms of run-time is C++ PDSDBSCAN (MPI variant), however, the scalability beyond 8 cores is already limited.

Even though the Java ELKI is optimised for serial execution, it is much slower than the parallel C++ implementations using a single core only. All implementations for Spark are much slower when using a single core only. (Spark DBSCAN was not measured using a low number of cores, because already with a high number of cores it was very slow.) When running on many cores, the Spark-based implementations beat the serial ELKI but are still by one (DBSCAN on Spark) or two (RDD-DBSCAN) orders of magnitude slower than HPDBSCAN and do not scale as well. While DBSCAN on Spark is faster than RDD-DBSCAN, it does only implement a simple approximation of DBSCAN and thus delivers completely different (=wrong) clusters than correct DBSCAN implementations.

Table 6 shows results using a higher number of cores. (No measurements of any of the two DBSCAN HPC implementations on the small Twitter data set have been made, as we can already see from Table 5 that using a higher number

Table 5. Run-time (in seconds) on small Twitter data set vs. number of cores

| Number of cores | 1 | 2 | 4 | 8 | 16 | 32 |
|------------------------------|-------|-------|-------|-------|-----|-----|
| HPDBSCAN MPI | 114 | 59 | 30 | 16 | 8 | 6 |
| PDSDBSCAN MPI | 288 | 162 | 106 | 90 | 85 | 88 |
| ELKI | 997 | – | – | – | – | – |
| RDD-DBSCAN | 7 311 | 3 521 | 1 994 | 1 219 | 889 | 832 |
| DBSCAN on Spark ^a | 1 105 | 574 | 330 | 174 | 150 | 147 |

^aDoes only implement an approximation of the DBSCAN algorithm.

Table 6. Run-time (in seconds) on small Twitter data set vs. number of cores

| Number of cores | 58 | 116 | 232 | 464 | 928 |
|------------------------------|-----|-----|-----|-----|-------|
| Spark DBSCAN | – | – | – | – | 2 406 |
| RDD-DBSCAN | 622 | 707 | 663 | 624 | 675 |
| DBSCAN on Spark ^a | 169 | 167 | 173 | 183 | 208 |

^aDoes only implement an approximation of the DBSCAN algorithm.

of cores does not give any gains on this small data set – measurements with many cores for HPDBSCAN running on the bigger Twitter data set are presented later). For Spark DBSCAN, an initial experiment has been made using 928 cores, but as it was rather slow, so no further experiments have been made for this implementation. For RDD-DBSCAN, no real speed-up can be observed when scaling the number of cores (run-times are more or less independent from the number of used cores and constant when taking into account the measurement deviations to be expected). The same applies to the DBSCAN on Spark implementation.

As pointed out in Sect. 4.3, it would have been interesting to compare the running time of MapReduce-based implementations using the same data set and hardware. Han et al. [34] who tried as well without success to get the implementations of MR-DBSCAN and DBSCAN-MR, developed for comparison reasons their own MapReduce-based implementation and observed a 9 to 16 times slower performance of their MapReduce-based DBSCAN implementation in comparison to their implementation for Spark.

Run-Time Measurements on Big Data Set. While the previous measurements were made using the smaller Twitter data set, also the bigger one containing 16 602 137 points was used in experiments in order to investigate some sort of “weak scaling”. While HPDBSCAN can easily handle it, the Spark implementations have problems with this 289 MB CSV file.

When running any of the Spark DBSCAN implementations while making use of all available cores of our cluster, we experienced out-of-memory exceptions⁶. Even though each node in our cluster has 42 GB RAM, this memory is shared by 24 virtual cores of that node. Hence, the number of cores used on each node had to be restricted using the `--executor-cores` parameter, thus reducing the parallelism, but leaving each thread more RAM (which was adjusted using the `--executor-memory` parameter).

The results for the big Twitter data set are provided in Table 7. HPDBSCAN (in the hybrid version using OpenMP within each node and MPI between nodes) scales well. Spark DBSCAN failed throwing the exception `java.lang.`

⁶ Despite these exceptions, we did only encounter once during all measurements a re-submissions of a failed Spark tasks – in this case, we did re-run the job to obtain a measurement comparable to the other, non failing, executions.

Table 7. Run-time (in seconds) on big Twitter data set vs. number of cores

| Number of cores | 1 | 384 | 768 | 928 |
|------------------------------|--------|-----|-----|-----------|
| HPBDBSCAN hybrid | 2 079 | 10 | 8 | – |
| ELKI | 15 362 | – | – | – |
| Spark DBSCAN | – | – | – | Exception |
| RDD-DBSCAN | – | – | – | 5 335 |
| DBSCAN on Spark ^a | – | – | – | 1 491 |

^aDoes only implement an approximation the DBSCAN algorithm.

Exception: Box for point Point at (51.382, -2.3846); id = 6618196; box = 706; cluster = -2; neighbors = 0 was not found. DBSCAN on Spark did not crash, but returned completely wrong clusters (it anyway does not cluster according to the original DBSCAN idea). RDD-DBSCAN took almost one and a half hour. Due to the long run-times of the DBSCAN implementations for Spark already with the maximum number of cores, we did not perform measurements with a lower number of cores.

5.5 Discussion of Results

Big data approaches aim at avoiding “expensive” network transfer of initial input data by moving computation where the data is available on local storage. In contrast, in HPC systems, the initial input data is not available locally, but via an external SAN storage system. Due to the fact that big data approaches aim at minimising network transfers, the fact that the Infiniband interconnection of CPU nodes used in the HPC configuration is faster than the Ethernet-based big data configuration, should not matter that much. In addition, because DBSCAN is not that I/O bound, but rather CPU bound, the I/O speed and file formats do not matter as much as the used programming languages and clever implementations in particular with respect to the quality of the domain decomposition for an effective load balancing of parallel execution.

HPBDBSCAN outperforms all other considered implementations. Even the optimised serial ELKI is slower than a serial run of HPBDBSCAN. This can be attributed to C++ code being faster than Java and to the fact that HPBDBSCAN uses the fast binary HDF5 file format, whereas all other implementations have to read and parse a textual input file (and respectively create and write a textual output file). Having a closer look at the scalability reveals furthermore that HPBDBSCAN scales (“strong scaling”) for the small data set very well up to 16 cores, but some saturation becomes visible with 32 cores (Table 5): Amdahl’s law [45] suggests that sequential parts and overheads start then to dominate.

For the given skewed data set, scalability of RDD-DBSCAN is only given for a low number of cores (the run-time difference between 16 and 32 cores is

within to be expected measurement deviations), but not beyond⁷. An analysis of the run-time behaviour reveals that in the middle of execution, only one long running task of RDD-DBSCAN is being executed by Spark: while one core is busy with this task, all other cores are idle and the subsequent RDD transformations cannot yet be started as they rely on the long running task. This means, the load is due to bad domain decomposition not well balanced and explains why RDD-DBSCAN does not scale beyond 58 cores: adding more cores just means adding more idle cores (while one core executes the long running task, the remaining 57 cores are enough to handle the workload of all the other parallel tasks). In fact, the serial ELKI using just one core is faster than RDD-DBSCAN using up to 8 cores and even beyond, RDD-DBSCAN is not that much faster and which does not really justify using a high number of cores. DBSCAN on Spark delivers completely wrong clusters, hence it has to be considered useless and it is pointless that it is faster than RDD-DBSCAN.

The comparison between HPDBSCAN and the Spark-based implementations shows that HPDBSCAN does a much better load balancing: the Spark-based implementations typically try to balance the number of points per partition before enlarging the partitions by *eps* on each side to add the ghost/halo regions which adds further extra points (which can be a significant amount in dense areas). Also, partitions cannot get smaller than $2\textit{eps}$. In contrast, HPDBSCAN balances the number of comparisons to be performed (i.e. calculating the Euclidean distance) and takes to this aim also comparisons of points inside the partition with points in the ghost/halo regions of that partition into account. Furthermore, partitions can get smaller than $2\textit{eps}$ which is also important to balance heavily skewed data. As a result, HPDBSCAN is able to balance the computational costs much better in particular for skewed data.

While the HPC implementations are much faster than the big data implementations, it is at least in favour of the big data implementations that HPC implementations require much more lines of code (=more development efforts) than the more abstract Scala implementations for Spark. Also, the big data platforms provide fault tolerance which is not given on HPC platforms.

6 Summary and Outlook

We surveyed existing parallel implementations of the spatial clustering algorithm DBSCAN for *High-Performance Computing* (HPC) platforms and big data platforms, namely Apache Hadoop and Apache Spark. For those implementations that were available as open-source, we evaluated and compared their performance in terms of run-time. The result is devastating: none of the evaluated implementations for Apache Spark is anywhere near to the HPC implementations. In particular on bigger (but still rather small) data sets, most of them fail completely and do not even deliver correct results.

⁷ Remarkably, the authors of RDD-DBSCAN [37] performed scalability studies only up to 10 cores.

As typical HPC hardware is much more expensive than commodity hardware used in most big data applications, one might be tempted to say that it is obvious that the HPC DBSCAN implementations are faster than all the evaluated Spark DBSCAN implementations. Therefore, in this case study, the same hardware was used for both platforms: HPC hardware. – But using commodity hardware instead would not change the result: while the HPC implementations of DBSCAN would then not benefit from the fast HPC I/O, a closer analysis reveals that typical big data considerations such as locality of data are not relevant in this case study, but rather proper parallelization such as decomposition into load balanced parallel tasks matters. The Spark implementations of DBSCAN suffer from a unsuitable decomposition of the input data. Hence, the used skewed input data leads to tasks with extremely imbalanced load on the different parallel cores.

It can be speculated that in HPC, parallelization needs to manually implemented and thus gets more attention in contrast to the high-level big data approaches where the developer gets not in touch with parallelization. Another reason to prefer HPC for compute-intensive tasks is that already based on the used programming languages, run-time performance of the JVM-based Spark platform can be expected to be one order of magnitude slower than C/C++. While RDDs support a bigger class of non-embarrassingly parallel problems than MapReduce, Spark still does not support as tight coupling as OpenMP and MPI used in HPC which might however be required for, e.g., simulations.

To conclude our story of Goliath and the elephant: if you do not even get the parallelization and load balancing right, it does matter whether you are Goliath or an elephant. Or – looking at it the other way around – if you want to take for your big data a fast DBSCAN algorithm off-the-shelf, you are currently better off if you take HPDBSCAN [29].

However, it has to be said that in general, the big data platforms such as Apache Spark offer resilience (such as re-starting crashed sub-tasks) and a higher level of abstraction (reducing time spent implementing an algorithm) in comparison to the low-level HPC approach.

As future work, it would be interesting to investigate the performance of the different implementations on other data sets – e.g. for non-skewed data, the load imbalance can be expected to disappear (but still, the C/C++ HPC implementations can be expected to be faster than the Java big data implementation). Furthermore, it is worthwhile to transfer the parallelization concepts of HPDBSCAN to a Spark implementation, in particular the domain decomposition below rectangles/hypercubes smaller than 2ϵ and the load balancing cost function of considering the number of comparisons of a partition including comparisons to points in the ghost/halo regions of that partition (in contrast to the Spark-based implementations considering only the number of points in a partition without taking ghost/halo regions into account). This would give the end user faster DBSCAN clustering on big data platforms. (Having more or less the same parallelization ideas implemented on both platforms would also allow better assessment of the influence of C/C++ versus Java/Scala and of MPI versus

the RDD approach of Spark.) Also, the scientific binary HDF5 data file format can currently not be processed by Hadoop or Spark in a way that data storage locality is exploited. Hence, the binary file had to be converted to a text-based input file format and one might argue that this I/O issue slowed down the Spark implementations. But in fact, the load imbalance of the CPU load contributes much more to the run-time than any I/O. As soon as the algorithms become better load balanced, less CPU bound and instead more I/O bound, data locality matters. A simple approach to be able to exploit the harder to predict locality of binary formats is to create some sort of “street map” in an initial and easily to parallelize run and use later-on the data locality information contained in this “street map” to send jobs to those nodes where the data is locally stored. We have successfully demonstrated this approach [3] for processing with Apache Hadoop the binary file formats used in the LHC experiments; the same approach should also be applicable to HDF5 files.

Acknowledgment. The author likes to thank all those who provided implementations of their DBSCAN algorithms. Special thanks go to the division Federated Systems and Data at the Jülich Supercomputing Centre (JSC), in particular to the research group High Productivity Data Processing and its head Morris Riedel. The author gratefully acknowledges the computing time granted on the supercomputer JUDGE at Jülich Supercomputing Centre (JSC).

The HPDBSCAN implementation of DBSCAN will be used as pilot application in the research project DEEP-EST (Dynamical Exascale Entry Platform – Extreme Scale Technologies) which receives funding from the European Union Horizon 2020 – the Framework Programme for Research and Innovation (2014–2020) under Grant Agreement number 754304.

References

1. Schmelling, M., Britsch, M., Gagunashvili, N., Gudmundsson, H.K., Neukirchen, H., Whitehead, N.: RAVEN – boosting data analysis for the LHC experiments. In: Jónasson, K. (ed.) PARA 2010. LNCS, vol. 7134, pp. 206–214. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28145-7_21
2. Memon, S., Vallot, D., Zwinger, T., Neukirchen, H.: Coupling of a continuum ice sheet model and a discrete element calving model using a scientific workflow system. In: Geophysical Research Abstracts. European Geosciences Union (EGU) General Assembly 2017, Copernicus, vol. 19 (2017). EGU2017-8499
3. Glaser, F., Neukirchen, H., Rings, T., Grabowski, J.: Using MapReduce for high energy physics data analysis. In: 2013 International Symposium on MapReduce and Big Data Infrastructure. IEEE (2013/2014). <https://doi.org/10.1109/CSE.2013.189>
4. Ester, M., Kriegel, H., Sander, J., Xu, X.: Density-based spatial clustering of applications with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press (1996)
5. Apache Software Foundation: Apache Hadoop (2017). <http://hadoop.apache.org/>
6. Neukirchen, H.: Performance of big data versus high-performance computing: some observations. In: Clausthal-Göttingen International Workshop on Simulation Science, Göttingen, Germany, 27–28 April 2017 (2017). Extended Abstract

7. Neukirchen, H.: Survey and performance evaluation of DBSCAN spatial clustering implementations for big data and high-performance computing paradigms. Technical report VHI-01-2016, Engineering Research Institute, University of Iceland (2016)
8. Neukirchen, H.: Elephant against Goliath: Performance of Big Data versus High-Performance Computing DBSCAN Clustering Implementations. EUDAT B2SHARE record (2017). <https://doi.org/10.23728/b2share.b5e0bb9557034fe087651de9c263000c>
9. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, vol. 14, issue 2. ACM (1984). <https://doi.org/10.1145/602259.602266>
10. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, vol. 19, issue 2. ACM (1990). <https://doi.org/10.1145/93597.98741>
11. Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**(9), 509–517 (1975). <https://doi.org/10.1145/361002.361007>
12. Kjolstad, F., Snir, M.: Ghost cell pattern. In: 2nd Annual Conference on Parallel Programming Patterns (ParaPLoP), 30–31 March 2010, Carefree, AZ. ACM (2010). <https://doi.org/10.1145/1953611.1953615>
13. Xu, X., Jäger, J., Kriegel, H.P.: A fast parallel clustering algorithm for large spatial databases. *Data Min. Knowl. Disc.* **3**(3), 263–290 (1999). <https://doi.org/10.1023/A:1009884809343>
14. Dagum, L., Menon, R.: OpenMP: an industry standard API for shared-memory programming. *IEEE Comput. Sci. Eng.* **5**(1), 46–55 (1998). <https://doi.org/10.1109/99.660313>
15. MPI Forum: MPI: A Message-Passing Interface Standard. Version 3.0, September 2012. <http://mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf>
16. OpenSFS and EOFS: Lustre (2017). <http://lustre.org/>
17. IBM: General Parallel File System Knowledge Center (2017). <http://www.ibm.com/support/knowledgecenter/en/SSFKCN/>
18. Folk, M., Heber, G., Koziol, Q., Pourmal, E., Robinson, D.: An overview of the HDF5 technology suite and its applications. In: Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pp. 36–47. ACM (2011). <https://doi.org/10.1145/1966895.1966900>
19. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, Berkeley, CA, USA. USENIX Association (2004). <https://doi.org/10.1145/1327452.1327492>
20. Apache Software Foundation: Apache Spark (2017). <http://spark.apache.org/>
21. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association (2012)
22. Jha, S., Qiu, J., Luckow, A., Mantha, P., Fox, G.C.: A tale of two data-intensive paradigms: applications, abstractions, and architectures. In: 2014 IEEE International Congress on Big Data, pp. 645–652. IEEE (2014). <https://doi.org/10.1109/BigData.Congress.2014.137>

23. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, University of California Press, Berkeley, California, pp. 281–297 (1967)
24. Ganis, G., Iwazkiewicz, J., Rademakers, F.: Data analysis with PROOF. In: Proceedings of XII International Workshop on Advanced Computing and Analysis Techniques in Physics Research. Number PoS(ACAT08)007 in Proceedings of Science (PoS) (2008)
25. Wang, Y., Goldstone, R., Yu, W., Wang, T.: Characterization and optimization of memory-resident MapReduce on HPC systems. In: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, pp. 799–808. IEEE (2014). <https://doi.org/10.1109/IPDPS.2014.87>
26. Kriegel, H.P., Schubert, E., Zimek, A.: The (black) art of runtime evaluation: are we comparing algorithms or implementations? *Knowl. Inf. Syst.* **52**(2), 341–378 (2016). <https://doi.org/10.1007/s10115-016-1004-2>
27. Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K.A., Zimek, A.: A framework for clustering uncertain data. *PVLDB* **8**(12), 1976–1979 (2015). <https://doi.org/10.14778/2824032.2824115>
28. Patwary, M.M.A., Palsetia, D., Agrawal, A., Liao, W.k., Manne, F., Choudhary, A.: A new scalable parallel DBSCAN algorithm using the disjoint-set data structure. In: International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–11. IEEE (2012). <https://doi.org/10.1109/SC.2012.9>
29. Götz, M., Bodenstern, C., Riedel, M.: HPDBSCAN: highly parallel DBSCAN. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, held in conjunction with SC15: The International Conference for High Performance Computing, Networking, Storage and Analysis. ACM (2015). <https://doi.org/10.1145/2834892.2834894>
30. Patwary, M.M.A.: PDSDBSCAN source code (2017). <http://users.eecs.northwestern.edu/~mpatwary/Software.html>
31. Götz, M.: HPDBSCAN source code. Bitbucket repository (2016). <https://bitbucket.org/markus.goetz/hpdbscan>
32. Baldridge, J.: ScalaNLP/Nak source code. GitHub repository (2015). <https://github.com/scalanlp/nak>
33. Busa, N.: Clustering geolocated data using Spark and DBSCAN. O’Reilly (2016). <https://www.oreilly.com/ideas/clustering-geolocated-data-using-spark-and-dbscan>
34. Han, D., Agrawal, A., Liao, W.K., Choudhary, A.: A novel scalable DBSCAN algorithm with Spark. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 1393–1402. IEEE (2016). <https://doi.org/10.1109/IPDPSW.2016.57>
35. Litouka, A.: Spark DBSCAN source code. GitHub repository (2017). https://github.com/alitouka/spark_dbscan
36. Stack Overflow: Apache Spark distance between two points using squaredDistance. Stack Overflow discussion (2015). <http://stackoverflow.com/a/31202037>
37. Cordova, I., Moh, T.S.: DBSCAN on Resilient Distributed Datasets. In: 2015 International Conference on High Performance Computing and Simulation (HPCS), pp. 531–540. IEEE (2015). <https://doi.org/10.1109/HPCSim.2015.7237086>
38. Cordova, I.: RDD DBSCAN source code. GitHub repository (2017). <https://github.com/irvingc/dbscan-on-spark>

39. He, Y., Tan, H., Luo, W., Feng, S., Fan, J.: MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data. *Front. Comput. Sci.* **8**(1), 83–99 (2014). <https://doi.org/10.1007/s11704-013-3158-3>
40. aizook: Spark_DBSCAN source code. GitHub repository (2014). <https://github.com/aizook/SparkAI>
41. Raad, M.: DBSCAN On Spark source code. GitHub repository (2016). <https://github.com/mraad/dbscan-spark>
42. He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., Fan, J.: MR-DBSCAN: an efficient parallel density-based clustering algorithm using MapReduce. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS), pp. 473–480. IEEE (2011). <https://doi.org/10.1109/ICPADS.2011.83>
43. Dai, B.R., Lin, I.C.: Efficient map/reduce-based DBSCAN algorithm with optimized data partition. In: 2012 IEEE Fifth International Conference on Cloud Computing, pp. 59–66. IEEE (2012). <https://doi.org/10.1109/CLOUD.2012.42>
44. Bodenstein, C.: HPDBSCAN Benchmark test files. EUDAT B2SHARE record (2015). <https://doi.org/10.23728/b2share.7f0c22ba9a5a44ca83cdf4fb304ce44e>
45. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: AFIPS Conference Proceedings Volume 30: 1967 Spring Joint Computer Conference, pp. 483–485. American Federation of Information Processing Societies (1967). <https://doi.org/10.1145/1465482.1465560>

Author Index

- Ahlbrecht, Tobias 161
Albert, Sebastian 3
- Begau, Christoph 112
Böhm, Sebastian 219
Bracht, Uwe 39
Brenner, Gunther 176
Bufe, Alexander 176
Burmeister, Sascha 77
- D'Ambrogio, Andrea 193
Dahlbeck, Mirko 39
Dix, Jürgen 161
Durak, Umut 193
- Feinauer, Julian 128
Fiekas, Niklas 161
Fischer, Anja 39
Friedrich, Markus 24
Furat, Orkun 145
- Ganesan, Hariprasath 112
Gerlach, Torsten 193
Grabowski, Jens 161, 176, 239
Grosch, Thorsten 97
Gu, Feng 97
- Hallmann, Corinna 77
Hartl, Maximilian 24
Hartmann, Sven 193
Herbold, Verena 161, 239
Hlushkou, Dzmitry 145
Holzer, Lorenz 145
Honsel, Daniel 161, 239
- Kirsche, Michael 219
Köhler, Christian 176
- Kolonko, Michael 97
Korte, Fabian 176
Kraus, Philipp 3
Krüger, Thomas 39
Kuchler, Klaus 128
- Magg, Christoph 24
Mahmoodi, Somaye 193
Miettinen, Kaisa 60
Müller, Jörg P. 3
- Neukirchen, Helmut 251
Neumann, Matthias 145
- Rausch, Andreas 208
- Schmidt, Volker 128, 145
Schöbel, Anita 3
Sindhya, Karthik 60
Staedtler, Andre 239
Suhl, Leena 77
Sutmann, Godehard 112
- Tallarek, Ulrich 145
- Waack, Stephan 161, 239
Welter, Marlon 161, 239
Westhoff, Daniel 128
Wieder, Philipp 176
Wissing, Michaela 77
Wittek, Stefan 208
- Yang, Zhixing 97
- Zhou-Kangas, Yue 60