Bart Verheij
Marco Wiering (Eds.)

Communications in Computer and Information Science 823

Artificial Intelligence

29th Benelux Conference, BNAIC 2017
Groningen, The Netherlands, November 8–9, 2017
Revised Selected Papers

Springer

Communications in Computer and Information Science 823

Commenced Publication in 2007
Founding and Former Series Editors:
Alfredo Cuzzocrea, Xiaoyong Du, Orhun Kara, Ting Liu, Dominik Ślęzak, and Xiaokang Yang

Editorial Board

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

Bart Verheij · Marco Wiering (Eds.)

Artificial Intelligence

29th Benelux Conference, BNAIC 2017
Groningen, The Netherlands, November 8–9, 2017
Revised Selected Papers

Springer

Editors
Bart Verheij
Artificial Intelligence
University of Groningen
Groningen
The Netherlands

Marco Wiering
Artificial Intelligence
University of Groningen
Groningen
The Netherlands

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-319-76891-5 ISBN 978-3-319-76892-2 (eBook)
https://doi.org/10.1007/978-3-319-76892-2

Library of Congress Control Number: 2018934359

Printed on acid-free paper

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

Preface

BNAIC is the annual Benelux Conference on Artificial Intelligence. In 2017, the 29th edition of BNAIC was organized by the Institute of Artificial Intelligence and Cognitive Engineering (ALICE), University of Groningen, under the auspices of the Benelux Association for Artificial Intelligence (BNVKI) and the Dutch Research School for Information and Knowledge Systems (SIKS).

BNAIC 2017 took place in Het Kasteel, Groningen, The Netherlands, on November 8–9, 2017. BNAIC 2017 included invited speakers, research presentations, posters, demonstrations, a deep learning workshop (organized by our sponsor NVIDIA) and a research and business session. Some 160 participants visited the conference.

The four BNAIC 2017 keynote speakers were:

- Marco Dorigo, Université Libre de Bruxelles
  "Swarm Robotics: Current Research Directions at IRIDIA"
- Laurens van der Maaten, Facebook AI Research
  "From Visual Recognition to Visual Understanding"
- Luc Steels, Institute for Advanced Studies (ICREA), Barcelona
  "Digital Replicants and Mind-Uploading"
- Rineke Verbrugge, University of Groningen
  "Recursive Theory of Mind: Between Logic and Cognition"

Three FACt talks (FACulty focusing on the FACts of Artificial Intelligence) were scheduled:

- Bert Bredeweg, Universiteit van Amsterdam
  "Humanly AI: Creating Smart People with AI"
- Eric Postma, Tilburg University
  "Towards Artificial Human-Like Intelligence"
- Geraint Wiggins, Queen Mary University of London/Vrije Universiteit Brussel
  "Introducing Computational Creativity"

Authors were invited to submit papers on all aspects of artificial intelligence. This year we received 68 submissions in total. Of the 30 submitted Type A regular papers, 11 (37%) were accepted for oral presentation, and 14 (47%) for poster presentation. Five (17%) were rejected. Of the 19 submitted Type B compressed contributions, 17 (89%) were accepted for oral presentation, and two (11%) for poster presentation. None were rejected. All six submitted Type C demonstration abstracts were accepted. Of the submitted 13 Type D thesis abstracts, five (38%) were accepted for oral presentation, and eight (62%) for poster presentation. None were rejected. The selection was made using peer review. Each submission was assigned to three members of the Program Committee, and their expert reviews were the basis for our decisions.

All submissions accepted for oral or poster presentations and all demonstration abstracts appeared in the electronic preproceedings, made available on the conference website during the conference (http://bnaic2017.ai.rug.nl/). All 11 Type A regular papers accepted for oral presentation (37%) appear in these proceedings, in the Springer CCIS series.

The BNAIC 2017 conference would not have been possible without the support and efforts of many. We thank the members of the Program Committee for their constructive and scholarly reviews. We are grateful to Elina Sietsema, Carlijne de Vries, and Sarah van Wouwe, members of the administrative staff at the Institute of Artificial Intelligence and Cognitive Engineering (ALICE), for their tireless and reliable support. We thank our local organization team Luca Bandelli, Abe Brandsma, Tomasz Darmetko, Mingcheng Ding, Ana Dugeniuc, Joel During, Ameer Islam, Siebert Looije, René Mellema, Michaela Mrázková, Annet Onnes, Benjamin Shafrey, Sjaak ten Caat, Albert Thie, Jelmer van der Linde, Luuk van Keeken, Paul Veldhuyzen, Randy Wind, and Galiya Yeshmagambetova, all of them students in our BSc and MSc Artificial Intelligence programs, for enthusiastically volunteering to help out in many ways. We thank Annet Onnes for preparing the proceedings, Jelmer van der Linde for developing the website, Randy Wind for designing the program leaflet, and Albert Thie for coordinating the local organization.

We are grateful to our sponsors for their generous support of the conference:

- Target Holding
- NVIDIA Deep Learning Institute
- Anchormen
- Quint
- The Netherlands Research School for Information and Knowledge Systems (SIKS)
- SIM-CI
- Textkernel
- LuxAI
- IOS Press
- Stichting Knowledge-Based Systems (SKBS)
- SSN Adaptive Intelligence

Two awards were presented during the conference. The BNAIC 2017 SNN Best Paper Award was won by Mathijs Pieters and Marco Wiering for their paper “Comparison of Machine Learning Techniques for Multi-label Genre Classification.” The BNAIC 2017 SKBS Best Demo Award was won by Denis Steckelmacher, Hèléne Plisnier, Diederik M. Roijers, and Ann Nowé for their demonstration entitled “Hierarchical Reinforcement Learning for a Robotic Partially Observable Task.”

Organizing the 2017 edition of the annual BNAIC conference series was an honor and a pleasure. We hope to meet you at a future edition.

February 2018

Bart Verheij
Marco Wiering

Organization

General Chairs

Bart Verheij	University of Groningen, The Netherlands
Marco Wiering	University of Groningen, The Netherlands

Contact and Administration

Elina Sietsema	University of Groningen, The Netherlands
Carlijne de Vries	University of Groningen, The Netherlands
Sarah van Wouwe	University of Groningen, The Netherlands

Local Organization

Luca Bandelli	University of Groningen, The Netherlands
Abe Brandsma	University of Groningen, The Netherlands
Tomasz Darmetko	University of Groningen, The Netherlands
Mingcheng Ding	University of Groningen, The Netherlands
Ana Dugeniuc	University of Groningen, The Netherlands
Joel During	University of Groningen, The Netherlands
Ameer Islam	University of Groningen, The Netherlands
Siebert Looije	University of Groningen, The Netherlands
René Mellema	University of Groningen, The Netherlands
Michaela Mràzková	University of Groningen, The Netherlands
Annet Onnes	University of Groningen, The Netherlands
Benjamin Shafrey	University of Groningen, The Netherlands
Sjaak ten Caat	University of Groningen, The Netherlands
Albert Thie	University of Groningen, The Netherlands
Jelmer van der Linde	University of Groningen, The Netherlands
Luuk van Keeken	University of Groningen, The Netherlands
Paul Veldhuyzen	University of Groningen, The Netherlands
Randy Wind	University of Groningen, The Netherlands
Galiya Yeshmagambetova	University of Groningen, The Netherlands

Program Committee

Stylianos Asteriadis	Maastricht University, The Netherlands
Reyhan Aydogan	Delft University of Technology, The Netherlands
Floris Bex	Utrecht University, The Netherlands
Michael Biehl	University of Groningen, The Netherlands
Mauro Birattari	Université Libre de Bruxelles, Belgium

Hendrik Blockeel	Katholieke Universiteit Leuven, Belgium
Hans L. Bodlaender	Utrecht University, The Netherlands
Sander Bohte	Centrum Wiskunde & Informatica (CWI), The Netherlands
Peter Bosman	Centrum Wiskunde & Informatica (CWI), The Netherlands
Tibor Bosse	Vrije Universiteit Amsterdam, The Netherlands
Bruno Bouzy	Paris Descartes University, France
Frances Brazier	Delft University of Technology, The Netherlands
Bert Bredeweg	University of Amsterdam, The Netherlands
Tristan Cazenave	Université Paris Dauphine, France
Tom Claassen	Radboud University Nijmegen, The Netherlands
Walter Daelemans	University of Antwerp, Belgium
Gregoire Danoy	University of Luxembourg, Luxembourg
Mehdi Dastani	Utrecht University, The Netherlands
Patrick De Causmaecker	Katholieke Universiteit Leuven, Belgium
Martine De Cock	University of Washington Tacoma, USA
Mathijs De Weerdt	Delft University of Technology, The Netherlands
Benoît Depaire	Hasselt University, Belgium
Frank Dignum	Utrecht University, The Netherlands
Virginia Dignum	Delft University of Technology, The Netherlands
Kurt Driessens	Maastricht University, The Netherlands
Madalina Drugan	Technical University of Eindhoven, The Netherlands
Sjur Dyrkolbotn	Western Norway University of Applied Sciences, Norway
Jason Farquhar	Radboud University Nijmegen, The Netherlands
Ad Feelders	Utrecht University, The Netherlands
Bart Goethals	University of Antwerp, Belgium
Pascal Gribomont	University of Liège, Belgium
Perry Groot	Radboud University Nijmegen, The Netherlands
Marc Gyssens	Universiteit Hasselt, Belgium
Frank Harmsen	Ernst & Young Advisory, The Netherlands
Maaike Harbers	Rotterdam University of Applied Science, The Netherlands
Tom Heskes	Radboud University Nijmegen, The Netherlands
Koen Hindriks	Delft University of Technology, The Netherlands
Arjen Hommersom	Open University of the Netherlands, The Netherlands
Mark Hoogendoorn	Vrije Universiteit Amsterdam, The Netherlands
Gerda Janssens	Katholieke Universiteit Leuven, Belgium
Maurits Kaptein	Eindhoven University of Technology, The Netherlands
Uzay Kaymak	Eindhoven University of Technology, The Netherlands
Walter Kosters	Leiden University, The Netherlands
Johan Kwisthout	Radboud University Nijmegen, The Netherlands
Tom Lenaerts	Université Libre de Bruxelles, Belgium
Marco Loog	Delft University of Technology, The Netherlands
Peter Lucas	Radboud University, The Netherlands
Elena Marchiori	Radboud University Nijmegen, The Netherlands
Wannes Meert	Katholieke Universiteit Leuven, Belgium
John-Jules Meyer	Utrecht University, The Netherlands
Peter Novák	Delft University of Technology, The Netherlands

Ann Nowé	Vrije Universiteit Brussel, Belgium
Aske Plaat	Leiden University, The Netherlands
Eric Postma	Tilburg University, The Netherlands
Henry Prakken	University of Utrecht, University of Groningen, The Netherlands
Jan Ramon	Inria, France
Silja Renooij	Utrecht University, The Netherlands
Nico Roos	Maastricht University, The Netherlands
Stefan Schlobach	Vrije Universiteit Amsterdam, The Netherlands
Pierre-Yves Schobbens	University of Namur, Belgium
Johannes Scholtes	Maastricht University, The Netherlands
Lambert Schomaker	University of Groningen, The Netherlands
Martijn Schut	Universiteit van Amsterdam, The Netherlands
Evgueni Smirnov	Maastricht University, The Netherlands
Matthijs T. J. Spaan	Delft University of Technology, The Netherlands
Gerasimos Spanakis	Maastricht University, The Netherlands
Jennifer Spenader	University of Groningen, The Netherlands
Ida Sprinkhuizen-Kuyper	Radboud University Nijmegen, The Netherlands
Thomas Stützle	Université Libre de Bruxelles, Belgium
Johan Suykens	Katholieke Universiteit Leuven, Belgium
Niels Taatgen	University of Groningen, The Netherlands
Annette Ten Teije	Vrije Universiteit Amsterdam, The Netherlands
Dirk Thierens	Utrecht University, The Netherlands
Karl Tuyls	University of Liverpool, UK
Antal van den Bosch	Radboud University Nijmegen, The Netherlands
Egon L. van den Broek	Utrecht University, The Netherlands
Jaap van den Herik	Leiden University, The Netherlands
Linda van der Gaag	Utrecht University, The Netherlands
Peter van der Putten	Leiden University, Pegasystems, The Netherlands
Leon van der Torre	University of Luxembourg, Luxembourg
Natalie van der Wal	Vrije Universiteit Amsterdam, The Netherlands
Tim van Erven	Leiden University, The Netherlands
Marcel van Gerven	Radboud University Nijmegen, The Netherlands
Frank van Harmelen	Vrije Universiteit Amsterdam, The Netherlands
Martijn van Otterlo	Vrije Universiteit Amsterdam, The Netherlands
M. Birna van Riemsdijk	Delft University of Technology, The Netherlands
Maarten van Someren	University of Amsterdam, The Netherlands
Marieke van Vugt	University of Groningen, The Netherlands
Menno van Zaanen	Tilburg University, The Netherlands
Joaquin Vanschoren	Eindhoven University of Technology, The Netherlands
Marina Velikova	Radboud University Nijmegen, The Netherlands
Remco Veltkamp	Utrecht University, The Netherlands
Joost Vennekens	Katholieke Universiteit Leuven, Belgium
Katja Verbeeck	Technologiecampus Gent, Belgium
Rineke Verbrugge	University of Groningen, The Netherlands

Michel Verleysen	Université Catholique de Louvain, Belgium
Sicco Verwer	Delft University of Technology, The Netherlands
Arnoud Visser	Universiteit van Amsterdam, The Netherlands
Willem Waegeman	Ghent University, Belgium
Martijn Warnier	Delft University of Technology, The Netherlands
Gerhard Weiss	Maastricht University, The Netherlands
Floris Wiesman	Academic Medical Center Amsterdam, The Netherlands
Jef Wijsen	University of Mons, Belgium
Mark H. M. Winands	Maastricht University, The Netherlands
Radboud Winkels	Universiteit van Amsterdam, The Netherlands
Cees Witteveen	Delft University of Technology, The Netherlands
Marcel Worring	University of Amsterdam, The Netherlands
Yingqian Zhang	Eindhoven University of Technology, The Netherlands

Contents

Learning-Based Diagnosis and Repair 1
Nico Roos

Competition Between Cooperative Projects. 16
Gleb Polevoy and Mathijs de Weerdt

Refining a Heuristic for Constructing Bayesian Networks from Structured Arguments 32
Remi Wieten, Floris Bex, Linda C. van der Gaag, Henry Prakken, and Silja Renooij

Reciprocation Effort Games 46
Gleb Polevoy and Mathijs de Weerdt

Get Your Virtual Hands Off Me! – Developing Threatening IVAs Using Haptic Feedback 61
Linford Goedschalk, Tibor Bosse, and Marco Otte

Tracking Perceptual and Memory Decisions by Decoding Brain Activity 76
Marieke van Vugt, Armin Brandt, and Andreas Schulze-Bonhage

The Origin of Mimicry: Deception or Merely Coincidence? 86
Bram Wiggers and Harmen de Weerd

Feature Selection in High-Dimensional Dataset Using MapReduce 101
Claudio Reggiani, Yann-Aël Le Borgne, and Gianluca Bontempi

Simultaneous Ensemble Generation and Hyperparameter Optimization for Regression 116
David Roschewitz, Kurt Driessens, and Pieter Collins

Comparison of Machine Learning Techniques for Multi-label Genre Classification 131
Mathijs Pieters and Marco Wiering

Learning to Play Donkey Kong Using Neural Networks and Reinforcement Learning. 145
Paul Ozkohen, Jelle Visser, Martijn van Otterlo, and Marco Wiering

Author Index 161

Learning-Based Diagnosis and Repair

Nico Roos(✉)

Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands
roos@maastrichtuniversity.nl

Abstract. This paper proposes a new form of diagnosis and repair based on reinforcement learning. Self-interested agents learn locally which agents may provide a low quality of service for a task. The correctness of learned assessments of other agents is proved under conditions on exploration versus exploitation of the learned assessments.

Compared to collaborative multi-agent diagnosis, the proposed learning-based approach is not very efficient. However, it does not depend on collaboration with other agents. The proposed learning based diagnosis approach may therefore provide an incentive to collaborate in the execution of tasks, and in diagnosis if tasks are executed in a suboptimal way.

1 Introduction

Diagnosis is an important aspect of systems consisting of autonomous and possibly self-interested agents that need to collaborate [4–12,14–30,32–34,37]. Collaboration between agents may fail because of malfunctioning agents, environmental circumstances, or malicious agents. Diagnosis may identify the cause of the problem and the agents responsible [31]. Efficient multi-agent diagnosis of collaboration failures also requires collaboration and requires sharing of information. Agents responsible for collaboration failures may be reluctant in providing the correct information. Therefore it is important to have an incentive to provide the right information. The ability to learn an assessments of other agents without the need to exchange information, may provide such an incentive.

This paper addresses the learning of a diagnosis in a network of distributed services. In such a network, tasks are executed by multiple agents where each agent does a part of the whole task. The execution of a part of a task will be called a service.

The more than 2000 year old silk route is an example of a distributed network of services. Local traders transported silk and other goods over a small part of the route between China and Europe before passing the goods on to other traders. A modern version of the silk route is a multi modal transport, which can consists of planes, trains, trucks and ships. Another example of distributed services is the computational services on a computer network. Here, the processing of data are the distributed services. In smart energy networks, consumers of energy may also be producers of energy. The energy flows have to be routed dynamically through

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 1–15, 2018.
https://doi.org/10.1007/978-3-319-76892-2_1

the network. A last example of a distributed service is Industry 4.0. In Industry 4.0, the traditional sequential production process is replaced by products that know which production steps (services) are required in their production. Each product selects the appropriate machine for the next production step and tells the machine what is should do.

To describe a network of distributed services such that diagnosis can be performed, we propose a directed graph representation. An arc of the graph represents the provision of a service by some agent. The nodes are the points where a task[1] is transferred from one agent to another. Incorrect task executions are modeled as transitions to special nodes.

The assumption that agents are self-interested and no agent has a global view of the network, limits the possibility of diagnosis and repair. We will demonstrate that it is still possible to learn which agents are reliable w.r.t. the quality of service that they provide.

The remainder of the paper is organized as follows. In the next section, we will present our graph-based model of a network of distributed services. Section 3 presents an algorithm for locally learning the reliability of agents providing services. Section 4 presents the experimental results and Sect. 5 concludes the paper.

2 The Model

We wish to model a network of services provided by a set of agents. The services provided by the agents contribute to the executions of tasks. The order of the services needed for a task need not be fixed, nor the agents providing the services. This suggests that we need a model in which services cause state transitions, and in each state there may be a choice between several agent-service combinations that can provide the next service. The service that is provided by an agent may be of different quality levels. We can model this at an abstract level by different state transitions. If we also abstract from the actual service descriptions, then we can use a graph based representation.

We model a network of services provided by a set of agents Ag using a graph $G = (N, A)$, where the N represents a set of nodes and $A = \{(n_i, n'_i, ag_i) \mid \{n_i, n'_i\} \subseteq N, ag_i \in Ag\}_{i=1}^{|A|}$ set of arcs. Each arc $(n, n', ag) \in A$ represents a service (n, n') that is provided by an agent $ag \in Ag$. We allow for multiple services between two nodes provided that the associated agents are different; i.e., several agents may provide the same service.

A set of tasks T is defined by pairs of nodes $(s, d) \in T$. Any path between the source s and the destination d of a task $(s, d) \in T$; i.e., a path $(a_1, \ldots, a_k)$ with $a_i = (n_i, n_{i+1}, ag_i)$, $n_1 = s$ and $n_{k+1} = d$, represents a correct execution of the task.

An incorrect execution of a task $(s, d) \in T$ is represented by a path that ends in a node d' not equal to d; i.e., a path $(a_1, \ldots, a_k)$ with $a_i = (n_i, n_{i+1}, ag_i)$, $n_1 = s$ and $n_{k+1} = d' \neq d$. A special node f is used to denote the complete

[1] In smart energy networks the tasks are the directions in which energy must flow.

failure of a service provided by an agent. No recovery from f is possible and no information about this failure is made available.

To describe a sub-optimal execution of a task $(s, d) \in T$, we associate a set of special nodes with each destination node d. These nodes indicate that something went wrong during the realization of the task. For instance, goods may be damaged during the execution of a transport task. The function $D : N \rightarrow 2^N$ will be used for this purpose. Beside the nodes denoting suboptimal executions, we also include the normal execution; i.e., $d \in D(d)$. Moreover, $f \in D(d)$.

To measure the quality of the execution of a task $(s, d) \in T$, we associate a utility with every possible outcome of the task execution: $U(d', d)$ for every $d' \in D(d)$. Here, $U(f, d) \leq U(d', d) < U(d, d)$ for every $d' \in D(d) \backslash d$.

The possible results of a service provided by agent ag in node n for a task $t = (s, d)$ with destination d, will be specified by the function $E(n, d, ag)$. This function $E : N \times N \times Ag \rightarrow 2^N$ specifies all nodes that may be reached by the provided service. The function must satisfy the following requirements:

- $E(n, d, ag) \subseteq \{n'' \mid (n, n'', ag) \in A\}$

We also define a probability distribution $e : N \times N \times Ag \times N \rightarrow [0, 1]$ over $E(n, d, ag)$, describing the probability of every possible outcome of the provided service; i.e.,

- $e(n, d, ag, n') = P(n' \mid n, d, ag)$
  where $n' \in E(n, d, ag)$ and $\sum_{n' \in E(n,d,ag)} e(n, d, ag, n') = 1$.

There may be several agents in a node n that can provide the next service for a task $t = (s, d)$ with destination d. The function $succ : N \times N \rightarrow 2^{Ag}$ will be used to denote the set of agents $succ(n, d) = \{ag_1, \ldots, ag_k\}$ that can provide the next service.

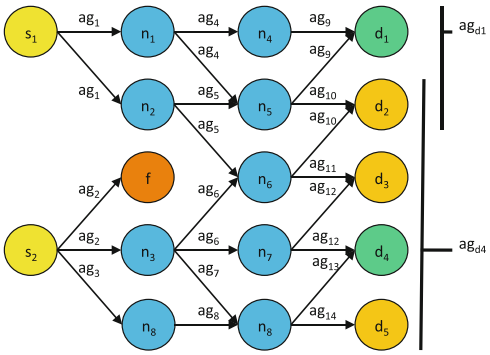


Fig. 1. An example network.

Figure 1 gives an illustration of a network of services represented as a graph. The network shows two starting nodes for tasks, s_1 and s_2, two successful destination nodes for tasks, d_1 and d_4, two unsuccessful destination nodes for tasks, d_2 and d_3, the failure node f and seven intermediate nodes.

3 Distributed Learning of Agent Reliability

Agents may learn locally diagnostic information using feedback about the result of a task execution. The diagnostic information learned by each agent may enable it to pass on a task in a node to a next agent such that the task is completed in the best possible way. So, an agent must learn the reputation of the agents to which it passes on tasks. This reputation may depend on the node in which the coordination with the next agent takes place as well as on all future agents that will provide services for the task.

We could view our model of a network of services provided by agents as a Markov Decision Process (MDP) [1,13]. In this markov decision process the nodes in $D(d)$ given the task (s, d), are absorbing states. Only when reaching a node in $D(d)$ a reward is received. All other rewards are 0. The transition probabilities are given by $e(n, d, ag, n')$. If these probabilities do not depend on the destination; i.e., $e(n, d, ag, n') = P(n' \mid n, ag)$, then we have a standard markov decision process for which the optimal policy can be learned using Q-learning [35,36]. However, Q-learning requires that an agent providing a service knows the Q-values of the services the next agent may provide. This implies that we have a Decentralized MDP [2,3] in which collaboration is needed to learn the optimal Q-values of services. If agents are willing to collaborate, it is, however, more efficient to use the traditional forms of diagnosis [31]. Therefore, in this section, we assume the agents are self-interested and do not collaborate.

To enable local learning of the agents' reputations, we assume that for every task $t = (s, d) \in T$ one and the same agent ag_d is associated with all nodes in $D(d) \backslash f$. Moreover, we assume that each agent that provided a service for the task execution, has added its signature to the task. The incentive for adding a signature is the payment for the provided service. The agent ag_d uses these signatures to make the payments and to inform the agents that provided a service about the success of the whole task execution. The latter information enables each service agent to assess the quality of the agents to which it passes on tasks. If the payments depend on the quality of service of the whole chain, the agents providing services will have an incentive to provide the best possible service and to pass on a task to a next agent such that the quality of the next services is maximized.

An agent ag that provided a service must pass on task $t = (s, d) \in T$ to the next agent if the task is not yet finished. There may be k agents that can provide the next service: $ag_1, \ldots, ag_k$. Assuming that agent ag can identify the current node n and thereby the quality of its own service, ag would like to learn which of the k agents is the most suited to provide the next service for the task.

The agent ag_d associated with the destination d of task $t = (s, d) \in T$ will inform agent ag about the actual quality $d' \in D(d)$ that is realized for the task. This feedback enables agent ag to evaluate the quality of the whole chain of services starting with a next agent ag_i. So, even if agent ag_i is providing a high quality service, it may not be a good choice if subsequent agents are failing.

An agent ag can learn for each combination of a task destination (the node d) a next agent ag' and the current node n, the probability that the remainder

of the task execution will result in the quality $d' \in D(d) \backslash f$. The probability estimate is defined as:

$$pe(d' \mid d, ag', n, i) = \frac{C_{d' \mid i}}{i}$$

where i is the number of times that a task t with destination d is passed on to agent ag' in the node n, and $C_{d' \mid i}$ is the number of times that agent ag_d gives the feedback of d' for task t with destination d.

Agent ag may not receive any feedback if the execution of task t ended in a complete failure, unless agent ag_d knows about the execution of t. In the absence of feedback, agent ag can still learn the probability estimate of a complete failure:

$$pe(f \mid d, ag', n, i) = \frac{C_{f \mid i}}{i}$$

where $C_{f \mid i}$ is the number of times that no feedback is received from agent ag_d. An underlying assumption is that agent ag_d always gives feedback when a task is completed, and that the communication channels are failure free.

Estimating the probability is not enough. The behavior of future agents may change over time thereby influencing the probability estimates $pe(d' \mid d, ag', n, i)$. Assuming that the transition probabilities $e(n, n', ag, n'')$ of provided services do not change over time, the coordination between agents when passing on tasks is the only factor influencing the probability estimate $pe(d' \mid d, ag', n, i)$. Since agents have an incentive to select the best possible next agent when passing on a task, we need to address the effect of this incentive on the probability estimates. *First, however, we will investigate the question whether there exist an optimal policy for passing on a task to a next agent and a corresponding probability* $P(d' \mid d, ag', n, i)$.

To answer the above question, utilities of task executions are important. With more than two possible outcomes for a task execution, i.e., $|D(d)| > 2$, the expected utility of a task execution needs to be considered. Therefore, we need to know the utility $U(d', d)$ of all outcomes $d' \in D(d)$. We assume that either this information is global knowledge or that agent ag_d provides this information in its feedback.

Using the utilities of task outcomes, we can prove that there exists an optimal policy for the agents, and corresponding probabilities.

Proposition 1. *Let ag be an agent that has to choose a next agent ag' to provide a service for the task $t = (s, d) \in T$ in node n. Moreover, let $P(d' \mid ag', d, n)$ be the probability of reaching d' given the policies of the succeeding agents.*

The utility $U(d, ag, n)$ an agent ag can realize in node n for a task t with destination d, is maximal *if* every *agent ag chooses a next agent ag^* in* every *node n in which it can provide a service, such that the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.*

Proof. Given a task $t = (s, d) \in T$ we wish to maximize the expected utility agent ag can realize in node n by choosing the proper next agent to provide a service for the task.

$$\begin{aligned} U(d, ag, n) &= \sum_{d' \in D(d)} P(d' \mid d, n) \cdot U(d', d) \\ &= \sum_{d' \in D(d)} \sum_{ag'} P(d' \mid ag', d, n) \cdot P(ag' \mid d, n) \cdot U(d', d) \\ &= \sum_{ag'} P(ag' \mid d, n) \cdot \sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d) \end{aligned}$$

Here $P(ag' \mid d, n)$ is the probability that agent ag chooses ag' to be the next agent.

Suppose that the term $\sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d)$ is maximal for $ag' = ag^*$. Then $U(d, ag, n)$ is maximal if agent ag chooses ag^* to be the next agent with probability 1; i.e., $P(ag^* \mid d, n) = 1$. Therefore,

$$U(d, ag, n) = \sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$$

We can rewrite this equation as:

$$\begin{aligned} U(d, ag, n) &= \sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d) \\ &= \sum_{d' \in D(d)} \sum_{n' \in E(n, d, ag^*)} P(d' \mid d, n') \cdot P(n' \mid ag^*, d, n) \cdot U(d', d) \\ &= \sum_{n' \in E(n, d, ag^*)} P(n' \mid ag^*, d, n) \cdot \sum_{d' \in D(d)} P(d' \mid d, n') \cdot U(d', d) \\ &= \sum_{n' \in E(n, d, ag^*)} e(n, d, ag^*, n') \cdot U(d, ag', n') \end{aligned}$$

Here $P(n' \mid ag^*, d, n)$ is the transition probability of the service provided by agent ag^*, and $U(d, ag^*, n') = \sum_{d' \in D(d)} P(d' \mid d, n') \cdot U(d', d)$ is the expected utility agent ag^* can realize in node n' by choosing the proper next agent to provide a service.

We can now conclude that to maximize $U(d, ag, n)$, agent ag must choose the agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d)$ is maximal, and agent ag^* ensures that $U(d, ag^*, n')$ is maximized. This result enables us to prove by induction to the maximum distance to a node $d' \in D(d)$ that for every agent ag, $U(d, ag, n)$ is *maximal* if every agent ag chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.

– *Initialization step* Let the current node be $d' \in D(d)$. Then the maximum distance is 0 and the current agent is the agent ag_d receiving the result of the task. So, $U(d, ag_d, d') = U(d', d)$.

– *Induction step* Let $U(d, ag_d, n')$ be maximal for all distances less than k. Let n be a node such that the maximum distance to a node in $D(d)$ is k. Then according to the above result, $U(d, ag, n)$ is maximal if agent ag chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal, and for every $n' \in E(n, d, ag^*)$, $U(d, ag^*, n')$ is maximal. The former condition holds according to the prerequisites mentioned in the proposition. The latter condition holds according to the *induction hypothesis*. Therefore, the proposition holds.

□

The proposition shows that there exists an optimal policy for the agents, namely choosing the next agent for which the expected utility is maximized. *The next question is whether the agent can learn the information needed to make this choice.* That is, for every possible next agent, the agent must learn the probabilities of every value in $D(d)$ for a task $t = (s, d) \in T$ with destination d. Since these probabilities depend on the following agents that provide services, the optimal probabilities, denoted by the superscript $*$, can only be learned if these agent have learned to make an optimal choice. So, each agent needs to balance *exploration* (choosing every next agent infinitely many times in order to learn the optimal probabilities) and *exploitation* (choosing the best next agent). We therefore propose the following requirements

– Every agent ag uses a probability $P_i(ag' \mid d, n)$ to choose a next agent ag' for the task with destination d. The index i denotes that this probability depends on the number of times this choice has been made till now.
– The probability $P_i(ag' \mid d, n)$ that agent ag will choose agent ag' of which the till now learned expected utility is sub-optimal, approximates 0 if $i \to \infty$.
– $\sum_{i \to \infty} P_i(ag' \mid d, n) = \infty$

The first requirement states that we use a probabilistic exploration. The second requirement ensures that the agent will eventually only exploit what it has learned. The third requirement ensures that the agent will select every possible next agent infinitely many times in order to learn the correct probabilities.

A policy meeting the requirements is the policy in which the agent ag chooses the currently optimal next agent ag' with probability $1 - \frac{1}{(k-1)i}$. Here, k is the number of agents that can perform the next service for a task with destination d, and i is the number of times agent ag has to choose one of these k agents for a task with destination d. The agents that are currently not the optimal choice are chosen with probability $\frac{1}{(k-1)i}$.

We can prove that any approach meeting the above listed requirements will enable agents to learn the optimal policy.

Theorem 1. *Let every agent ag meet the above listed requirements for the probability $P_i(ag' \mid d, n)$ of choosing the next agent. Moreover, let $P^*(d' \mid ag, d, n)$ be the optimal probability of reaching the node $d' \in D(d)$ if every agent chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.*

Then, every agent ag learns $P^*(d' \mid ag', d, n)$ through $pe(d' \mid ag', d, n, i)$ if the number of tasks with destination d for which agent ag has to choose a next agent ag', denoted by i, goes to infinity.

Proof. We have to prove that: $\lim_{i\to\infty} pe(d' \mid ag', d, n, i) = P^*(d' \mid ag', d, n)$.

We can rewrite $\lim_{i\to\infty} pe(d' \mid ag, d, n, i)$ as:

$$\begin{aligned}
\lim_{i\to\infty} pe(d' \mid ag', d, n, i) &= \lim_{i\to\infty} \frac{C_{d' \mid i}}{i} \\
&= \lim_{i\to\infty} \sum_{n'\in E(n,d,ag')} \frac{C_{n' \mid i}}{i} \cdot \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}} \\
&= \lim_{i\to\infty} \sum_{n'\in E(n,d,ag')} pe(n' \mid ag', d, n, i) \cdot \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}} \\
&= \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, n) \cdot \lim_{i\to\infty} \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}}
\end{aligned}$$

We will prove that $C_{n' \mid i} \to \infty$ if $i \to \infty$ and $P(n' \mid ag', d, n) > 0$. That is, for every $x \in \mathbb{N}$, $\lim_{i\to\infty} P(C_{n' \mid i} > x) = 1$.

$$\begin{aligned}
\lim_{i\to\infty} P(C_{n' \mid i} > x) &= \lim_{i\to\infty} 1 - P(C_{n' \mid i} \le x) \\
&= 1 - \lim_{i\to\infty} \sum_{j=0}^{x} (P(n' \mid ag', d, n))^j \cdot (1 - P(n' \mid ag', d, n))^{i-j} \\
&= 1
\end{aligned}$$

So, $C_{n' \mid i} \to \infty$ if $i \to \infty$. Therefore,

$$\lim_{i\to\infty} pe(d' \mid ag', d, n, i) = \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, n) \cdot \lim_{j\to\infty} pe(d' \mid d, n', j)$$

The estimated probability $pe(d' \mid d, n', j)$ depends on the probability of choosing the next agent. This probability is a function of the j-th time agent ag' must choose a next agent ag'' for a task with destination d in node n'.

$$\lim_{j\to\infty} pe(d' \mid d, n', j) = \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n') \cdot \frac{C_{d' \mid ag'',j}}{C_{ag'' \mid j}}$$

where $C_{ag'' \mid j}$ is the number of times that agent ag'' was chosen to be the next agent, and $C_{d' \mid ag'',j}$ is the number of times that subsequently node d' was reached.

We will prove that $C_{ag'' \mid j} \to \infty$ if $j \to \infty$ and $P_j(ag'' \mid d, n) > 0$ for every j. That is, for every $x \in \mathbb{N}$, $\lim_{i\to\infty} P(C_{ag'' \mid j} > x) = 1$. A complicating factor is that $P_j(ag' \mid d, n)$ can be different for every value of j. Let y be the index of

the last time agent ag'' is chosen, and let p_x be the probability of all possible sequences till index y. Then we can formulate:

$$\begin{aligned}\lim_{j\to\infty} P(C_{ag''\mid j} > x) &= \lim_{j\to\infty} 1 - P(C_{ag''\mid j} \leq x)\\ &= 1 - p_x \cdot \lim_{j\to\infty} \prod_{k=y+1}^{j} (1 - P_k(ag'' \mid d, n))\\ &= 1 - e^{\ln(p_x)+\sum_{k=y+1}^{\infty} \ln(1-P_k(ag''\mid d,n))}\end{aligned}$$

According to the Taylor expansion of $\ln(\cdot)$: $\ln(1 - P_k(ag'' \mid d, n)) < -P_k(ag' \mid d, n)$. Therefore,

$$\begin{aligned}\lim_{j\to\infty} P(c_{ag''\mid j} > x) &= 1 - e^{\ln(p_x)-\sum_{k=y+1}^{\infty} P_k(ag''\mid d,n)}\\ &= 1 - e^{\ln(p_x)-\infty} \;=\; 1\end{aligned}$$

The above result implies:

$$\lim_{j\to\infty} pe(d' \mid d, n', j) = \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n') \cdot \lim_{k\to\infty} pe(d' \mid ag'', d, n', k)$$

We can now prove the theorem by induction to the maximum distance to a node $d' \in D(d)$.

- *Initialization step.* Let the current node be $d' \in D(d)$. The maximum distance is 0 and the current agent is the agent ag_d receiving the result of the task. So, $\lim_{i\to\infty} pe(d' \mid ag_d, d, d', i) = P^*(d' \mid ag_d, d, d') = 1$.
- *Induction step.* Let $\lim_{j\to\infty} pe(d' \mid ag', d, n', j) = P^*(d' \mid ag', d, n')$ be maximal for all distances less than k. Moreover, let the maximum distance from n to d' be k.
  Then, the expected utility of agent $ag'' \in succ(n', d)$ is:

$$\begin{aligned}\lim_{j\to\infty} U_j(ag'', d, n') &= \lim_{j\to\infty} \sum_{d'\in D(d)} pe(d' \mid ag'', d, n', j) \cdot U(d', d)\\ &= \sum_{d'\in D(d)} P^*(d' \mid ag', d, n') \cdot U(d', d) = U^*(ag'', d, n')\end{aligned}$$

According to the requirement,

$$\lim_{j\to\infty} P_j(ag_j^* \mid d, n') = 1 \text{ for } ag_j^* = \text{argmax}_{ag''} U_j(ag'', d, n')$$

So,

$$\begin{aligned}ag^* &= \lim_{j\to\infty} ag_j^*\\ &= \lim_{j\to\infty} \text{argmax}_{ag''} U_j(ag'', d, n')\\ &= \text{argmax}_{ag''} U^*(ag'', d, n')\end{aligned}$$

This implies:

$$\begin{aligned}\lim_{j\to\infty} pe(d' \mid d, n', j) &= \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n) \cdot \lim_{k\to\infty} pe(d' \mid ag'', d, n', k) \\ &= \sum_{ag''\in succ(n',d)} P^*(d' \mid ag'', d, n') \cdot \lim_{j\to\infty} P_j(ag'' \mid d, n) \\ &= P^*(d' \mid ag^*, d, n') \;\; = \;\; P^*(d' \mid d, n')\end{aligned}$$

Therefore,

$$\begin{aligned}\lim_{i\to\infty} pe(d' \mid ag', d, n, i) &= \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, d) \cdot \lim_{j\to\infty} pe(d' \mid d, n', j) \\ &= \sum_{n'\in E(n,d,ag')} e(n, d, ag', n') \cdot P^*(d' \mid d, n') \\ &= P^*(d' \mid ag', d, n)\end{aligned}$$

□

The theorem shows us that each agent can learn which next agent results in an expected high or low quality for the remainder of a task. In order to learn this assessment, the agents must explore all possible choices for a task infinitely many times. At the same time the agents may also exploit what they have learned sofar. In the end the agents will only exploit what they have learned. Hence, the learning-based approach combines *diagnosis and repair*.

An advantage of the learning-based approach is that intermitting faults can be addressed and that no collaboration between service agents is required. A disadvantage is that making diagnosis requires information about many executions of the same task. However, as we will see in the next section, a repair is learned quickly at the price that correctly functioning agents may be ignored.

Agents learn an assessment for each possible destination. In special circumstances, they need not consider the destination, and can focus on the next agent that can provide a service for a task. First, the quality of service provided by an agent does not depend on the destination of the task. Second, we do not use utilities for the result of a task and only identify whether a task execution is successful. If these conditions are met, an agent can learn for every next agent the probability that the task execution will be successful.

4 Experiments

To determine the applicability of the theoretical results of the previous section, we ran several experiments. For the experiments, we used a network of n^2 normal nodes organized in n layers of n nodes. Every normal node in a layer, except the last layer, is connected to two normal nodes in the next layer. Moreover, from every normal node in de first layer, every normal node in the last layer can be reached. With every transition a different agent is associated. To model

that these agents may provide a low quality of service, for every transition from normal node n to normal node n' representing the correct execution of a service by an agent, there is also a transition from n to an abnormal node n'' representing the incorrect execution of the service. Here, the abnormal node n'' is a duplicate of the normal node of n'. For every normal node except the nodes in the first layer, there is a duplicate abnormal node denoting the sub-optimal execution of a service. In this model, no recovery is possible. Figure 2 show a 4 by 4 network. The normal nodes that can be used for a normal execution of tasks are shown in yellow, blue and green. The duplicate abnormal nodes representing a sub-optimal execution are shown in orange. The transitions to the latter nodes and the transitions between the latter nodes are not shown in the figure.

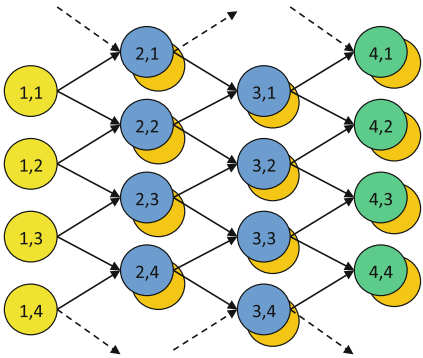


Fig. 2. The network used in the experiments. Note that the dashed arrows denote transitions from nodes (1,4), (2,1) and (3,4) to nodes (2,1), (3,4) and (4,1) respectively.

In our first experiment we determined how often a randomly chosen service is executed in 10000 randomly chosen tasks. We used a network of 10 by 10 nodes in this experiment. Figure 3 shows the cumulative results as a function of the number of processed task. Figure 4 shows in which experiment the service is used.

In the second experiment we used the same network. A fault probability of 0.1 was assigned to the randomly chosen service. Again, we measured how often a service is executed in 10000 randomly chosen tasks. Figure 5 shows the cumulative results as a function of the number of experiments, and Fig. 6 shows in which task the service is executed. We clearly see that the agents learn to avoid the agent that provides a low quality of service.

The results show that each agent learns to avoid passing on a task to an agent that may provide a low quality of service. An agent uses the estimated probabilities of a successful completion of a task when passing on the task to the next agent. Nevertheless, as shown in Fig. 6, the agents still try the low quality service, but with an increasingly lower probability. This exploration is necessary to learn the correct probabilities.

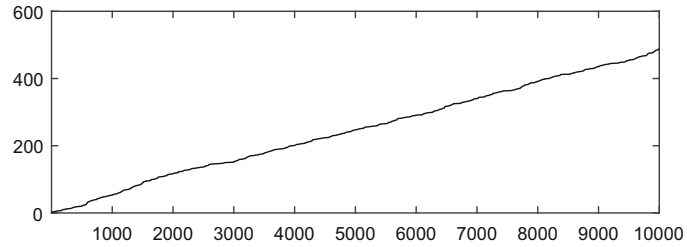


Fig. 3. The number of times a selected service is chosen as a function of the number of processed task.

Fig. 4. The tasks in which a selected service is chosen.

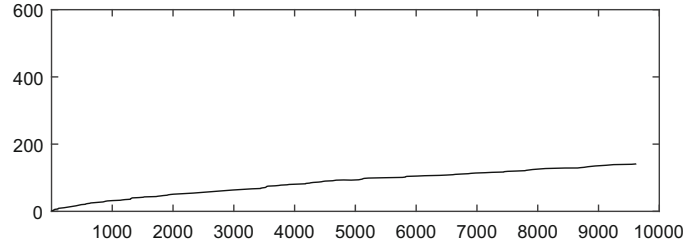


Fig. 5. The number of times a selected service is chosen as a function of the number of processed task.

Fig. 6. The tasks in which a selected service is chosen.

Inspection of the learned probabilities shows that the learning process is slow w.r.t. the total number of executed tasks. Figure 7 shows the learning of the probability that choosing an agent in a node n will result in a good quality of service for a task with a specific destination d. The probability that must be learned is 0.5. The agents only learn when they provided a service for a task with destination d. In Fig. 7, the service is executed only 4 times for tasks with destination d of 10000 executions of randomly chosen task. Although the learning process is slow, it is not a problem for the behavior of the network of distributed

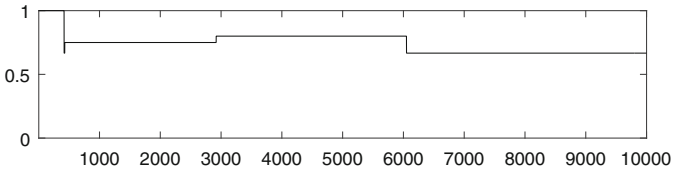


Fig. 7. Learning of the service success probability given a destination.

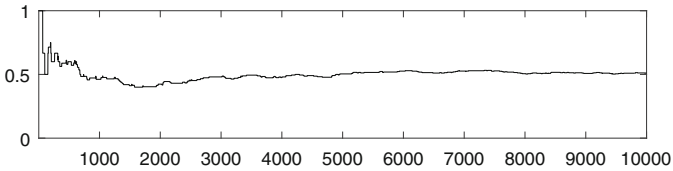


Fig. 8. Learning of the service success probability ignoring the destination.

services. However, it does result in avoiding the services provided by some agents while there is no need for it.

In the third experiment we learned the probability that choosing an agent will result in a good quality of service for a task, independent of the destination of the task. Figure 8 shows the result of the learning process. Again the probability that must be learned is 0.5. The learning process is much faster. However, as discussed at the end of the previous section, ignoring the destination of a task is only possible if the quality of service does not depend on the destination, and if we only identify whether a task is successful.

5 Conclusions

This paper presented a model for describing a network of distributed services for task executions. Each service is provided by an autonomous, possibly self-interested agent. The model also allows for the description of sub-optimal and failed services.

When a task is completed with a low quality, we would like to determine which service was of insufficient quality, which agent was responsible for the provision of this service, and how we can avoid agents that might provide a low quality of service. To answer these questions, the paper investigated an approach for learning in a distributed way an assessment of other agents. The learned information can be exploited to maximize the quality of a task execution. The correctness of the learned diagnosis an repair approach is proved, and demonstrated through experiments.

An important aspect of the distributed learning approach is that agents do not have to collaborate. Since diagnosis of distributed services is about identifying the agents that are to blame for a low quality of service, this is an important

property. It provides an incentive for being honest if agents make a diagnosis in a collaborative setting. Systematic lying will be detected eventually.

This research opens up several lines of further research. First, other policies that balance exploration and exploitation could be investigated. Second, more special cases in which the learning speed can be improved should be investigated. The topology might, for instance, be exploited to improve the learning speed. Third, since agents learn to avoid services of low quality before accurately learning the corresponding probabilities, we may investigate whether we can abstract from the actual probabilities. Fourth, as mentioned in the Introduction and above, the learned assessments provide an incentive for honesty when agents make a collaborative diagnosis. Is this incentive sufficient for agents to collaborate if traditional diagnostic techniques are used?

References

1. Bellman, R.: A markovian decision process. J. Math. Mech. **6**, 679–684 (1957)
2. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: Sequential optimality and coordination in multiagent systems. Math. Oper. Res. **6**(4), 819–840 (2002)
3. Boutilier, C.: Sequential optimality and coordination in multiagent systems. In: IJCAI, pp. 478–485 (1999)
4. Console, L., Torasso, P.: Hypothetical reasoning in causal models. Int. J. Intell. Syst. **5**, 83–124 (1990)
5. Console, L., Torasso, P.: A spectrum of logical definitions of model-based diagnosis. Comput. Intell. **7**, 133–141 (1991)
6. Davis, R.: Diagnostic reasoning based on structure and behaviour. Artif. Intell. **24**, 347–410 (1984)
7. de Jonge, F., Roos, N.: Plan-execution health repair in a multi-agent system. In: PlanSIG 2004 (2004)
8. de Jonge, F., Roos, N., Aldewereld, H.: MATES 2007. LNCS, vol. 4687. Springer, Heidelberg (2007)
9. de Jonge, F., Roos, N., Aldewereld, H.: Temporal diagnosis of multi-agent plan execution without an explicit representation of time. In: BNAIC-07 (2007)
10. de Jonge, F., Roos, N., van den Herik, J.: Keeping plan execution healthy. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 377–387. Springer, Heidelberg (2005). https://doi.org/10.1007/11559221_38
11. de Jonge, F., Roos, N., Witteveen, C.: Diagnosis of multi-agent plan execution. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2006. LNCS (LNAI), vol. 4196, pp. 86–97. Springer, Heidelberg (2006). https://doi.org/10.1007/11872283_8
12. de Jonge, F., Roos, N., Witteveen, C.: Primary and secondary plan diagnosis. In: The International Workshop on Principles of Diagnosis, DX-06 (2006)
13. Howard, R.A.: Dynamic Programming and Markov Processes. MIT Press, New York (1960)
14. Kalech, M., Kaminka, G.A.: On the design of social diagnosis algorithms for multi-agent teams. In: IJCAI-03, pp. 370–375 (2003)
15. Kalech, M., Kaminka, G.A.: Diagnosing a team of agents: scaling-up. In: AAMAS 2005, pp. 249–255 (2005)

16. Kalech, M., Kaminka, G.A.: Towards model-based diagnosis of coordination failures. In: AAAI 2005, pp. 102–107 (2005)
17. Kalech, M., Kaminka, G.A.: On the design of coordination diagnosis algorithms for teams of situated agents. Artif. Intell. **171**, 491–513 (2007)
18. Kalech, M., Kaminka, G.A.: Coordination diagnostic algorithms for teams of situated agents: scaling up. Comput. Intell. **27**(3), 393–421 (2011)
19. de Kleer, J., Mackworth, A.K., Reiter, R.: Characterizing diagnoses and systems. Artif. Intell. **56**, 197–222 (1992)
20. de Kleer, J., Williams, B.C.: Diagnosing with behaviour modes. In: IJCAI 89, pp. 104–109 (1989)
21. Micalizio, R.: A distributed control loop for autonomous recovery in a multi-agent plan. In: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, (IJCAI-09), pp. 1760–1765 (2009)
22. Micalizio, R.: Action failure recovery via model-based diagnosis and conformant planning. Comput. Intell. **29**(2), 233–280 (2013)
23. Micalizio, R., Torasso, P.: On-line monitoring of plan execution: a distributed approach. Knowl. Based Syst. **20**, 134–142 (2007)
24. Micalizio, R., Torasso, P.: Plan diagnosis and agent diagnosis in multi-agent systems. In: Basili, R., Pazienza, M.T. (eds.) AI*IA 2007. LNCS (LNAI), vol. 4733, pp. 434–446. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74782-6_38
25. Micalizio, R., Torasso, P.: Team cooperation for plan recovery in multi-agent systems. In: Petta, P., Müller, J.P., Klusch, M., Georgeff, M. (eds.) MATES 2007. LNCS (LNAI), vol. 4687, pp. 170–181. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74949-3_15
26. Micalizio, R., Torasso, P.: Monitoring the execution of a multi-agent plan: dealing with partial observability. In: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08), pp. 408–412. IOS Press (2008)
27. Micalizio, R., Torasso, P.: Cooperative monitoring to diagnose multiagent plans. J. Artif. Intell. Res. **51**, 1–70 (2014)
28. Micalizio, R., Torta, G.: Explaining interdependent action delays in multiagent plans execution. Auton. Agent. Multi-Agent Syst. **30**(4), 601–639 (2016)
29. Raiman, O., de Kleer, J., Saraswat, V., Shirley, M.: Characterizing non-intermittent faults. In: AAAI 91, pp. 849–854 (1991)
30. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**, 57–95 (1987)
31. Roos, N., ten Teije, A., Witteveen, C.: Reaching diagnostic agreement in multi-agent diagnosis. In: AAMAS 2004, pp. 1254–1255 (2004)
32. Roos, N., Witteveen, C.: Diagnosis of plan execution and the executing agent. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698, pp. 161–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11551263_14
33. Roos, N., Witteveen, C.: Diagnosis of plan structure violations. In: Petta, P., Müller, J.P., Klusch, M., Georgeff, M. (eds.) MATES 2007. LNCS (LNAI), vol. 4687, pp. 157–169. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74949-3_14
34. Roos, N., Witteveen, C.: Models and methods for plan diagnosis. J. Auton. Agents Multi-Agent Syst. **19**, 30–52 (2008)
35. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis Cambridge University (1989)
36. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**(3), 279–292 (1992)
37. Witteveen, C., Roos, N., van der Krogt, R., de Weerdt, M.: Diagnosis of single and multi-agent plans. In: AAMAS 2005, pp. 805–812 (2005)

Competition Between Cooperative Projects

Gleb Polevoy[1(✉)] and Mathijs de Weerdt[2]

[1] University of Amsterdam, Amsterdam, The Netherlands
g.polevoy@uva.nl
[2] Delft University of Technology, Delft, The Netherlands
m.m.deweerdt@tudelft.nl

Abstract. A paper needs to be good enough to be published; a grant proposal needs to be sufficiently convincing compared to the other proposals, in order to get funded. Papers and proposals are examples of cooperative projects that compete with each other and require effort from the involved agents, while often these agents need to divide their efforts across several such projects. We aim to provide advice how an agent can act optimally and how the designer of such a competition (e.g., the program chairs) can create the conditions under which a socially optimal outcome can be obtained. We therefore extend a model for dividing effort across projects with two types of competition: *a quota* or *a success threshold*. In the quota competition type, only a given number of the best projects survive, while in the second competition type, only the projects that are better than a predefined success threshold survive. For these two types of games we prove conditions for equilibrium existence and efficiency. Additionally we find that competitions using a success threshold can more often have an efficient equilibrium than those using a quota. We also show that often a socially optimal Nash equilibrium exists, but there exist inefficient equilibria as well, requiring regulation.

1 Introduction

Cooperative projects often compete with each other. For example, a paper needs to have a certain quality, or to be among a certain number of the best papers to be published, and a grant needs to be one of the best to be awarded. Either the projects that achieve a certain *minimum level*, or those that are among a certain *quota* of the best projects attain their value. Agents endowed with a resource budget (such as time) need to divide this resource across several such projects. We consider so-called public projects where agents contribute resources to create something together. If such a project survives the competition, its rewards are typically divided among the contributors based on their individual investments.

Agents often divide effort across competing projects. In addition to co-authoring articles or books [6,7,10] and research proposals, examples include

G. Polevoy—Most of this work was done at Delft University of Technology.

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 16–31, 2018.
https://doi.org/10.1007/978-3-319-76892-2_2

participating in crowdsensing projects [8] and online communities [9]. Examples of quotas for successful projects include investing effort in manufacturing several products, where the market becomes saturated with a certain number of products. Examples of success thresholds are investing in start-ups, where a minimum investment is needed to survive, or funding agencies contributing to social projects, where a minimum contribution is required to make the project succeed. Another example is students investing effort in study projects.

The ubiquity and the complexity of such competing projects calls for a decision-support system, helping agents to divide their efforts wisely. Assuming rationality of all the others, an agent needs to know how to behave given the behavior of the others, and the designer of the competition would like to know which rules lead to better results. In the terms of non-cooperative game theory, the objective of this work is to find the equilibria and their efficiency.

Analyzing the NE and their efficiency helps characterizing the influence of a quota or a success threshold on how efficient the stable strategies are for the society and thus increase the efficiency of investing time in the mentioned enterprises. For example, Batchelor [4] suggests increasing the publication standards. However, in addition to maximizing the total value of the published papers, he considers goals such as reducing the noise (number of low quality publications).

To make things clear, we employ this running example:

Example 1. Consider scientists investing time from their time budget in writing papers. A paper attains its value (representing the acknowledgment and all the related rewards) if it stands up to the competition with other papers. The competition can mean either being one of the q best papers, or achieving at least the minimum level of δ, depending on the circumstances. A scientist is rewarded by a paper by becoming its co-author if she has contributed enough to that paper.

Here, the submitters need to know how to split their efforts between the papers, and the conference chairs need to properly organize the selection process, e.g. by defining the quota or threshold on the papers to get accepted.

There were several studies of contributing to projects but the projects did not compete. For example, in the all-pay auction model, only one contributor benefits from the project, but everyone contributes. Its equilibria are analyzed in [5], etc. A famous example is the colonel Blotto game with two players [14], where these players spread their forces among the battlefields, winning a battle if allocating it more forces than the opponent does. The relative number of won battles determines the player's utility. Anshelevich and Hoefer [2] model two-player games by an undirected graph where nodes contribute to the edges. A project, being an edge, obtains contributions from two players. They study minimum-effort projects, proving the existence of an NE and showing that the price of anarchy (PoA)[1] is at most 2.

[1] The social welfare is the sum of the utilities of all the players. The price of anarchy [11,12] is the ratio of the minimum social welfare in an NE to the maximum possible social welfare. The price of stability [1,15] is the ratio of the maximum social welfare in an NE to the maximum possible social welfare.

The effort-dividing model [13] used the model of a shared effort game [3], where each player has a budget to divide among a given set of projects. The game possesses a contribution threshold θ, and the project's value is equally shared among the players who invest above this threshold. They analyzed Nash equilibria (NE) and their price of anarchy (PoA) and stability (PoS) for such games. However, they ignored that projects may compete for survival. We fill this gap, extending their model by allowing the projects only to obtain their modeled value if they stand up to a competition. To conclude, we study the yet unanswered question of strategic behavior with multiple competing projects.

Compared to the contribution in [10], we model contributing to multiple projects by an agent, and concentrate on the competition, rather than on sharing a project's utility. Unlike devising division rules to make people contribute properly, studied in cooperative game theory (see *Shapley value* [16] for a prominent example), we model given division rules and analyze the obtained game, using non-cooperative game theory.

We formally define the following models:

1. Given a *quota* q, only q projects receive their value. This models the limit on the number of papers to be accepted to a conference, the number of politicians in a city council, the lobbyists being the agents and the politicians being the projects, or the number of projects an organization can fund.
2. There exists a *success threshold* δ, such that only the projects that have a value of at least δ actually receive their value. This models a paper or proposal acceptance process that is purely based on quality.

Our contributions are as follows: We analyze existence and efficiency of NE in these games. In particular, we demonstrate that introducing a quota or a success threshold can sometimes kill existing equilibria, but sometimes allow for new ones. We study how adjusting a quota or a success threshold influences the contribution efficiency, and thereby the social welfare of the participants. We derive that competitions using a success threshold have efficient equilibria more often than those with a quota. We also prove that characterizing the existence of an NE would require more parameters than just the quota or the threshold and the number of the agents and the projects.

We formalize our models in Sect. 2, analyze the Nash equilibria of the first model and their efficiency in Sect. 3, and analyze the second model in Sect. 4. Theorems 2, 3, 5 and 6 are inspired by the existence and efficiency results for the model without competition. Having analyzed both models of competition between projects, Sect. 5 compares their characteristics, the possibility to influence the authors' behavior through tuning the acceptance criteria, and draws further conclusions. Some proofs are deferred to the Appendix A.

2 Model

We build our model on that from [13], since that is a model of investment in common projects with a general threshold. We first present their model for

shared effort games, which also appears in [3]. From Definition 1 on, we introduce competition among the projects.

There are n players $N = \{1, \ldots, n\}$ and a set Ω of m projects. Each player $i \in N$ can contribute to any of the projects in Ω_i, where $\emptyset \subsetneq \Omega_i \subseteq \Omega$; the contribution of player i to project $\omega \in \Omega_i$ is denoted by $x^i_\omega \in \mathbb{R}_+$. Each player i has a budget $B_i > 0$, so that the strategy space of player i (i.e., the set of her possible actions) is defined as $\left\{x^i = (x^i_\omega)_{\omega \in \Omega_i} \in \mathbb{R}_+^{|\Omega_i|} \middle| \sum_{\omega \in \Omega_i} x^i_\omega \leq B_i\right\}$. Denote the strategies of all the players except i by x^{-i}.

The next step to define a game is defining the utilities. Let us associate each project $\omega \in \Omega$ with its *project function*, which determines its *value*, based on the total contribution $x_\omega = (x^i_\omega)_{i \in N}$ that it receives; formally, $P_\omega(x_\omega) \colon \mathbb{R}^n_+ \to \mathbb{R}_+$. The assumption is that every P_ω is increasing in every parameter. The increasing part stems from the idea that receiving more effort does not make a project worse off. When we write a project function as a function of a single parameter, like $P_\omega(x) = \alpha x$, we assume that project functions P_ω depend only on the $\sum_{i \in N}(x^i_\omega)$, which is denoted by x_ω as well, when it is clear from the context. The project's value is distributed among the players in $N_\omega \triangleq \{i \in N | \omega \in \Omega_i\}$ according to the following rule. From each project $\omega \in \Omega_i$, each player i gets a share $\phi^i_\omega(x_\omega) \colon \mathbb{R}^n_+ \to \mathbb{R}_+$ with free disposal:

$$\forall \omega \in \Omega : \sum_{i \in N_\omega} \phi^i_\omega(x_\omega) \leq P_\omega(x_\omega). \tag{1}$$

We assume the sharing functions are non-decreasing. The non-decreasing assumption fits the intuition that contributing more does not get the players less.

Denote the vector of all the contributions by $x = (x^i_\omega)^{i \in N}_{\omega \in \Omega}$. The utility of a player $i \in N$ is defined to be

$$u^i(x) \triangleq \sum_{\omega \in \Omega_i} \phi^i_\omega(x_\omega).$$

Consider the numerous applications where a minimum contribution is required to share the revenue, such as paper co-authorship and homework. To analyze these applications, define a specific variant of a shared effort game, called a *θ-sharing mechanism*. This variant is relevant to many applications, including co-authoring papers and participating in crowdsensing projects. For any $\theta \in [0, 1]$, the players who get a share are defined to be $N^\theta_\omega \triangleq \left\{i \in N_\omega | x^i_\omega \geq \theta \cdot \max_{j \in N_\omega} x^j_\omega\right\}$, which are those who bid at least θ fraction of the maximum bid size to ω. Define the θ-equal sharing mechanism as equally dividing the project's value between all the users who contribute to the project at least θ of the maximum bid to the project.

The *θ-equal sharing mechanism*, denoted by M^θ_{eq}, is

$$\phi^i_\omega(x_\omega) \triangleq \begin{cases} \frac{P_\omega(x_\omega)}{|N^\theta_\omega|} & \text{if } i \in N^\theta_\omega, \\ 0 & \text{otherwise.} \end{cases}$$

Let us consider θ-equal sharing, where all the project functions are linear, i.e. $P_\omega(x_\omega) = \alpha_\omega(\sum_{i \in N} x^i_\omega)$. W.l.o.g., $\alpha_m \geq \alpha_{m-1} \geq \ldots \geq \alpha_1$. We denote the number of projects with the largest coefficient project functions by $k \in \mathbb{N}$, i.e. $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$. We call those projects *steep*. Assume w.l.o.g. that $B_n \geq \ldots \geq B_2 \geq B_1$.

A project that receives no contribution in a given profile is called a *vacant* project. A player is *dominated* at a project ω, if it belongs to the set $D_\omega \triangleq N_\omega \setminus N^\theta_\omega$. A player is *suppressed* at a project ω, if it belongs to the set $S_\omega \triangleq \left\{i \in N_\omega : x^i_\omega > 0\right\} \setminus N^\theta_\omega$. That is, a player who is contributing to a project but is dominated there.

We now depart from [13] and model competition in two different ways.

Definition 1. *In the* quota model*, given a natural number $q > 0$, only the q highest valued projects actually obtain a value to be divided between their contributors. The rest obtain zero. In the case of ties, all the projects that would have belonged to the highest q under some tie breaking rule receive their value; therefore, more than q projects can receive their value in this case. Formally, project ω is* in the quota *if $|\{\omega' \in \Omega | P_{\omega'}(x_{\omega'}) > P_\omega(x_\omega)\}| < q$, and ω is* out of the quota *otherwise, and, effectively, $P_\omega(x_\omega) = 0$.*

The second model is called the success threshold model.

Definition 2. *In the* success threshold model*, given a threshold δ, only the projects with value at least δ, meaning that $P_\omega(x_\omega) \geq \delta$, obtain a value, while if $P_\omega(x_\omega) < \delta$, then, effectively, $P_\omega(x_\omega) = 0$.*

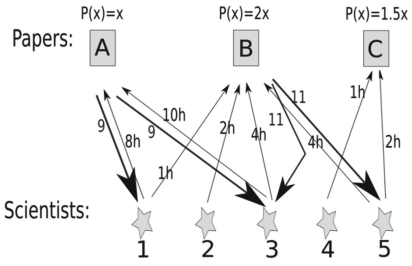


Fig. 1. Scientists contribute time to papers (arrows up), and share the value of the accepted ones (arrows down).

Example 1 (Continued). Figure 1 depicts a success threshold model, where paper C does not make it to the success threshold, and is, therefore, unpublished. The other two papers are above the success threshold, and get published; such a paper's recognition is equally divided between the contributors who contribute at least θ of the maximum contribution to the paper, and become co-authors.

3 The Quota Model

In this section, we study the equilibria of shared effort games with a quota and their efficiency. We first give an example of an NE, and generalize it to a sufficiency theorem. Then, we provide equilibrium existence and efficiency theorems for the quota model. Finally, we show that no simple setting of parameters guarantees the existence of an equilibrium or the lack thereof.

Intuitively, introducing a quota can make previously unstable profiles become NE, by making deviations non-profitable. This would increase the price of stability but decrease the price of anarchy. On the other hand, a profile that is an NE without a quota can cease being so in our model, since some projects may obtain no value because of the quota.

Having a quota can lead to counter-intuitive results. In the following example, there can be an NE where no steep project obtains a contribution. The idea is that any deviation from the project where everyone contributes is non-profitable, because it would still leave the other projects out of quota.

Example 2. Given projects 1 and 2, such that $\alpha_2 > \alpha_1$, assume that all the players contribute all their budgets to project 1. If $\alpha_2 B_n < \alpha_1 \sum_{i=1}^{n-1} B_i$ and $q = 1$, then no player can deviate to project 2, as this would still leave that project out of the quota, and therefore, this profile is an NE.

In this NE, the social welfare is equal to $\alpha_1 \sum_{i \in N} B_i$. The optimal social welfare, achieved if and only if all the players contribute all their budgets to project 2, is equal to $\alpha_2 \sum_{i \in N} B_i$. The ratio between the social welfare in this NE and the optimal one is $\frac{\alpha_1}{\alpha_2}$. That ratio is an upper bound on the price of anarchy of this game. In addition, since the optimal profile is also an NE, the price of stability is 1.

The price of anarchy is smaller than $\frac{\alpha_1}{\alpha_2}$ if and only if some agents do not contribute all their budgets. This can only happen in an NE if θ is positive, and if this is the case, then we can have arbitrarily low price of anarchy, down to the case when only agent n contributes, if $\theta B_n > B_{n-1}$, and then, $\text{PoA} = \frac{\alpha_1 B_n}{\alpha_2 \sum_{i \in N} B_i}$.

We now generalize these ideas to the following theorem about possible NE.

Theorem 1. *Consider a θ-equal sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$ (the order is w.l.o.g.), $0 < \theta < 1$, linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g.), and quota q.*

This game has a pure strategy NE, if $q = 1$ and $B_n < \sum_{i=1}^{n-1} B_i$. Additionally, $\text{PoA} = \frac{\alpha_l \sum_{i:\ B_i \geq \theta B_n} B_i}{\alpha_m \sum_{i \in N} B_i} \leq \frac{\alpha_l}{\alpha_m}$, *for* $l \triangleq \min \left\{ j \in \Omega : \alpha_m B_n < \alpha_j \sum_{i=1}^{n-1} B_i \right\}$, $\text{PoS} = 1$.

Proof. If all the players contribute to any single project $j \geq l$, then since $B_n < \sum_{i=1}^{n-1} B_i$, no player can deviate to any project, because this would still leave this project out of the quota. Therefore, this profile is an NE.

In particular, when all the players invest all their budgets in project m, it is an NE, and thus, $\text{PoS} = 1$.

To find the price of anarchy, notice that the worst equilibrium for the social welfare is when everyone contributes to the least profitable possible project, i.e. l, and only those who have a reason to do so contribute. Having an incentive means being not below the threshold amount, θB_n. This equilibrium yields $\alpha_l \sum_{i:\ B_i \geq \theta B_n} B_i$. ■

This theorem, in accord with the intuition above, shows that reducing the quota can either facilitate an optimal NE, or a very inferior NE. Actually, every efficiency of the form $\frac{\alpha_i}{\alpha_m}$ is possible at equilibrium, which brings us to the question of equilibria appearing and disappearing, which we treat next.

Example 3. A game with NE can cease having equilibria after introducing a quota. For example, consider $\theta \in (0,1)$, 2 players with budgets $B_1 = \theta B_2$ and 2 projects with the coefficients $\alpha, (1-\epsilon)\alpha$. This game has an NE if no quota exists, by Theorem 3 from [13]. However, introducing the quota of $q = 1$ implies there is no NE. Indeed, the only candidate profile for an equilibrium is both agents contributing everything to the same project or when both projects obtain the same value. In the former case, agent 2 would like to deviate, to avoid sharing, since the projects are close for small enough an ϵ. In the latter case, agent 2 contributes to both projects, since for small enough an ϵ, agent 1 alone would make the project be out of quota. Since there exists at least one project where agent 1 contributes less than θB_2, say project i, agent 2 would benefit from contributing to that project all its budget. This is because she would gain at least $(1-\epsilon)\alpha x_\omega^2$ while losing at most $\alpha(B_1 + x_\omega^2)/2$, which is smaller, for small enough ϵ and θ.

A game can also start having equilibria after introducing a quota. For instance, consider a game with two players, $B_2 = B_1$, $\alpha_m = 1.9\alpha_{m-1}$. Then Theorem 3 from [13] implies that no NE exists, but if we introduce the quota of 1, then both agents contributing to project m is an NE, since a deviator would be out of quota.

We now present an existence theorem. The theorem presents possible equilibria, providing advice on possible stable states. Afterwards, we study efficiency.

Theorem 2. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$ (the order is w.l.o.g.), $0 < \theta < 1$, linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g.), and quota q.*

This game has a pure NE if one of the following holds.[2]

1. $B_1 \geq k\theta B_n$, $k \leq q$ *and* $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$,
2. $B_1 \geq q\theta B_n$, $k \geq q$ *and* $B_n < \sum_{j=1}^{n-1} B_i/q$;
3. $B_{n-1} < \frac{\theta}{|\Omega|} B_n$ *and all the project functions are equal, i.e.* $\alpha_m = \alpha_1$.

[2] If α_{m-k} does not exist, consider the containing condition to be vacuously true.

The proof provides a profile and shows that no deviation is profitable.

Proof. To prove part 1, distinguish between the case where $k \leq q$ and $k > q$. If $k \leq q$, then the profile where all the players allocate $1/k$th of their respective budgets to each of the steep projects is an NE for the same reasons that were given for the original model, since here, the quota's existence can only reduce the motivation to deviate.

As for the part 2, consider the profile where all the players allocate $1/q$th of their respective budgets to each of the q steep projects $m, m-1, \ldots, m-q+1$. This is an NE, since the only deviation that is possibly profitable, besides reallocating between the non vacant projects, is a player moving all of her contributions from some projects to one or more of the vacant projects. This cannot bring profit, because these previously vacant projects will be outside of the quota, since $B_n < \sum_{j=1}^{n-1} B_i/q$. As for reallocating between the non-vacant projects, this is not profitable, since $B_1 \geq q\theta B_n$ means that suppressing is impossible. Therefore, this is an NE.

We now prove part 3. Let every player divide her budget equally among all the projects. No player wants to deviate, for the following reasons. All the projects obtain equal value, and therefore are in the quota. Player n suppresses all the rest and obtains her maximum possible profit, $\alpha_m(\sum_{i \in N} B_i)$. The rest obtain no profit, since they are suppressed whatever they do. ■

We now prove an efficiency result, based on Theorem 2.

Theorem 3. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$, $0 < \theta < 1$ (the order is w.l.o.g.), linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g.).(see footnote 2), and quota q.*

1. *If at least one of the following holds.*
   (a) $B_1 \geq k\theta B_n$, $k \leq q$ and $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$,
   (b) $B_1 \geq q\theta B_n$, $k \geq q$ and $B_n < \sum_{j=1}^{n-1} B_i/q$;
   Then, there exists a pure strategy NE and there holds: $\mathrm{PoS} = 1$.
2. *Assume $B_{n-1} < \frac{\theta}{|\Omega|} B_n$ and all the project functions are equal, i.e. $\alpha_m = \alpha_1$. Then, there exists a pure strategy NE and the following holds:* $\mathrm{PoS} = 1, \mathrm{PoA} = \frac{B_n}{\sum_{i \in \{1,2,\ldots,n\}} B_i}$.

Proof. We first prove part 1a and 1b. According to the proof of parts 1 and 2 of Theorem 2, equally dividing all the budgets among $\min\{k, q\}$ steep projects is an NE. Therefore, $\mathrm{PoS} = 1$.

For part 2, recall that in the proof of part 3 of Theorem 2, we show that everyone equally dividing the budgets between all the projects is an NE. This is optimal for the social welfare, and so $\mathrm{PoS} = 1$. We turn to find the price of anarchy now. If player n acts as just mentioned, while the other players do not contribute anything, then this is an NE, since all the projects are equal and therefore, in the quota, and players $1, \ldots, n-1$ will be suppressed at any contribution. An NE cannot have a lower social welfare, since n gets at least

$\alpha_m B_n$ in any NE, since this is obtainable alone. Therefore, the fraction between the two social welfare values, namely $\frac{\alpha_m B_n}{\alpha_m \sum_{i\in\{1,2,\dots,n\}} B_i}$, is the PoA. ■

The condition "$k \geq q$ and $B_n < \sum_{j=1}^{n-1} B_i/q$" in Theorem 3 does not hold if the largest budget can be much larger than the rest, implying that we shall ask whether our optimum NE is guaranteed by part 1a, which requires that the quota has to be at least k. When there are many equally glorious projects to contribute to, meaning that k is large, this constraint becomes non-trivial to implement. The condition "$k \leq q$ and $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$" in Theorem 3 does not hold if the difference between the two largest projects is not big enough, and then the quota has to be at most k if one wants our optimum NE to follow from part 1b. This is non-trivial when we have few most glorious (steep) projects.

We do not know a full characterization of the existence of equilibria; we do know that it would require many parameters. We prove now that the quota with the number of agents and projects do not determine existence.

Proposition 1. *For any quota $q \geq 1$, any number of agents $n \geq 2$ and projects $m \geq 2$, there exists a game which possesses an NE, and a game which does not.*

The proof engineers games with the given parameters with and without NE.

Proof. A game that satisfies the conditions of Theorem 2 provides evidence for the existence.

To find a game without an NE, we first treat the case of $q = 1$. Let all the project coefficient be equal to one another and let

$$B_n > \sum_{i=1}^{n-1} B_i, \tag{2}$$

$$\text{and } B_n > \frac{\sum_{i=1}^{n} B_i}{|\{i \in N : B_i \geq \theta B_n\}|}. \tag{3}$$

Because of the equality of all the project coefficients and of (2), in an equilibrium, all the agents with budgets at least θB_n will be together with n. Then, (3) implies agent n will deviate, contradictory to having an equilibrium.

For quota $q \geq 2$, let $B_{n-1} < \frac{\theta}{m} B_n$. In any NE, agent n dominates all the rest in the sense that it invests (strictly) more than $B_{n-1}\theta$ in any project that is in the quota, because otherwise, the other agents could get a share at some projects, and assuming $\alpha_l(x + \frac{x}{\theta}) > \alpha_m(\frac{x}{\theta})$ for every project l, agent n would prefer to suppress that. However, if $\alpha_m > \alpha_{m-1}$, n would always prefer to move a bit more contribution to project m, contradictory to the assumption of an NE. ■

4 The Success Threshold Model

In this section, we consider the NE of shared effort games with a success threshold. We allow success thresholds δ be at most the sum of all the budgets times α_m,

to let at least one project to obtain its value, in at least one strategy profile. We begin with an example, which inspires a theorem, and then we study existence and efficiency with a given success threshold.

In a profile, we call a project that has a value of at least the threshold an *accepted* project, and we call it *unaccepted* otherwise. In Example 1, the accepted papers are A and B.

Success threshold can cause counter-intuitive results, as follows.

Example 4. Given the projects 1 and 2, such that $\alpha_2 > \alpha_1$, assume that all the players contribute all their budgets to project 1. If $\delta > \alpha_2 B_n$, then no player can deviate to project 2, as this would leave that project unaccepted, and therefore, this profile is an NE.

The conclusions about the prices of anarchy and stability are the same as in Example 2, besides that the price of anarchy can be even zero if $\alpha_1 \sum_{i=1}^{n} B_i < \delta$.

The exemplified ideas yield the following theorem.

Theorem 4. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$ (the order is w.l.o.g.), $0 < \theta < 1$, linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g.), and success threshold δ.*

This game has a pure NE, if $\alpha_m B_n < \delta$. In addition, $\text{PoA} \leq \frac{\alpha_1}{\alpha_m}$ *and* $\text{PoS} = 1$. *If $\alpha_1 \sum_{i=1}^{n} B_i < \delta$, then* $\text{PoA} = 0$.

Proof. If all the players contribute to any single project, then since $\alpha_m B_n < \delta$, no player can deviate to any project, because this would still leave that project unaccepted. Therefore, this profile is an NE.

In particular, when all the players invest all their budgets in project m, it is an NE, and thus, $\text{PoS} = 1$. When all the players invest in 1, it also is an NE, showing that $\text{PoA} \leq \frac{\alpha_1}{\alpha_m}$, and if $\alpha_1 \sum_{i=1}^{n} B_i < \delta$, then $\text{PoA} = 0$. ■

This theorem, in accord with the intuition above, shows that increasing the success threshold can either facilitate an optimal NE, or an inferior NE. Actually, every efficiency of the form $\frac{\alpha_j}{\alpha_m}$, for $j \geq \min\{i : \alpha_i \sum_{l=1}^{n} B_l \geq \delta\}$, is possible at an equilibrium.

Example 5 (Introducing a success threshold can kill or create new NE). The game with $\theta \in (0, 0.5)$, 2 players with budgets $B_1 = 2\theta B_2$ and 2 projects with the coefficients α, α has an NE if no success threshold exists, by Theorem 3 from [13]. If we introduce the success threshold of αB_2, then in any NE both agents have to contribute to the same project. Then, agent 2 will deviate. For an emerging NE, consider a game with two players, $B_2 = B_1$, $\alpha_m = 1.9\alpha_{m-1}$. Then Theorem 3 from [13] implies that no NE exists, but if we introduce the success threshold of $2B_1\alpha_m$, then both agents contributing to project m constitute an NE, since a deviator would be at a project below the success threshold.

Next, we provide sufficient conditions for the existence of an NE.

Theorem 5. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$ (the order is w.l.o.g.), $0 < \theta < 1$, linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g.), and success threshold δ.*

This game has a pure NE, if one of the following holds (see Footnote 2). Define $p \triangleq \left\lfloor \frac{\alpha_m \sum_{i \in N} B_i}{\delta} \right\rfloor$; intuitively, it is the number of the projects that can be accepted.

1. *$B_1 \geq k\theta B_n$, $k \leq p$ and $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$,*
2. *$B_1 \geq p\theta B_n$, $k \geq p \geq 1$ and $\alpha_m B_n < \delta$;*
3. *$B_{n-1} < \frac{\theta}{|\Omega|} B_n$, all the project functions are equal, i.e. $\alpha_m = \alpha_1$.*

Proof. We first prove part 1. The profile where all the players allocate $1/k$th of their respective budgets to each of the steep projects is an NE for the same reasons that were given for the original model, since here, the requirement to be not less than the success threshold can only reduce the motivation to deviate.

In part 2, consider the profile where all the players allocate $1/p$th of their respective budgets to each of the p steep projects $m, m-1, \ldots, m-p+1$. This is an NE, since the only deviation that is possibly profitable, besides moving budgets between the non vacant projects, is a player moving all of her contributions from some projects to one or more of the vacant projects. This cannot bring profit, because these previously vacant projects will be unaccepted, since $\alpha_m B_n < \delta$. Additionally, any reallocating between the non-vacant projects is not profitable, since $B_1 \geq p\theta B_2$ means that suppressing is impossible. Therefore, the current profile is an NE.

We now prove part 3. We distinguish between the case where the condition $p \geq |\Omega|$ holds or not. If $p \geq |\Omega|$, then the proof continues as in the case of part 3 of Theorem 2, where every player divides her budget equally among all the projects. All the projects are accepted, so no new deviations become profitable.

In the case that $p < |\Omega|$, consider the profile where all the players allocate $1/p$th of their respective budgets to each of the p projects $m, m-1, \ldots, m-p+1$. This is an NE, since the only deviation that is possibly profitable is some player $j < n$ moving all her budget to a vacant project. However, this is not profitable, since the project would be unaccepted, because $B_j \leq B_{n-1} < \frac{\theta}{|\Omega|} B_n < \theta\delta/\alpha_m \leq \delta/\alpha_m$. The penultimate inequality stems from $p < |\Omega| \iff \frac{\alpha_m \sum_{i \in N} B_i}{|\Omega|} < \delta$. Therefore, this is an NE. ■

We now provide an efficiency result, proven in the appendix.

Theorem 6. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$, $0 < \theta < 1$ (the order is w.l.o.g.), linear project functions with coefficients $\alpha_m = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g) (see Footnote 2), and success threshold δ. Define $p \triangleq \left\lfloor \frac{\alpha_m \sum_{i \in N} B_i}{\delta} \right\rfloor$, as in Theorem 5.*

1. *If at least one of the following holds.*
   (a) $B_1 \geq k\theta B_n$, $k \leq p$ *and* $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$,
   (b) $B_1 \geq p\theta B_n$, $k \geq p \geq 1$ *and* $\alpha_m B_n < \delta$;
   Then, there exists a pure NE and there holds: $\mathrm{PoS} = 1$.
2. *Assume* $B_{n-1} < \frac{\theta}{|\Omega|} B_n$, *all the project functions are equal, i.e.* $\alpha_m = \alpha_1$.
   Then, there exists a pure NE and $\mathrm{PoS} = 1$. *If, an addition,* $\alpha_m B_n \geq \delta$, *then* $\mathrm{PoA} = \frac{B_n}{\sum_{i \in \{1,2,\ldots,n\}} B_i}$.

Condition 1a of Theorem 6 implies that if the second best project is close to a best one, then the threshold should be big enough, for condition 1b to guarantee our optimum NE. The contrapositive of the condition 1b implies that if the biggest player is able to make a project succeed on her own, then the threshold should be small enough so that p is at least the number of the most profitable projects, for our optimum NE to be guaranteed by condition 1a.

There exists no simple characterization for the NE existence when $\delta \leq \alpha_m B_n$.

Proposition 2. *For any success threshold* $\delta \in [0, \alpha_m B_n]$ *and any number of agents* $n \geq 2$ *and projects* $m \geq 2$, *there exists a game which possesses an NE, and a game which does not.*

The proof appears in Appendix A.

5 Conclusions and Further Research

We analyze the stable investments in projects, where a project has to comply to certain requirements to obtain its value. This models paper co-authorship, investment in firms, etc. The goal is to advise which investments are individually and socially preferable. Each agent freely divides her budget of time or effort between the projects. A project that succeeds in the competition obtains a value, which is divided between the contributors who have contributed at least a given fraction of the maximum contribution to the project. We model succeeding in a competition either by a quota of projects that actually obtain their value, or by a success threshold on the value of projects that do.

For purposes like organizing a conference, we ask which quota or success threshold would make the behavior of the players better for the social welfare. Theorem 1 implies that if no player has a budget as large as the total budget of all the other players times the ratio between the least and the largest project coefficient, then the quota of 1 makes many equilibria, including an optimal one, possible. Theorem 4 promises the same by choosing a success threshold that disables any player to make a project successful on her own. The first problem of this approach is that it also allows very inefficient profiles constitute equilibria, asking for some coordination. The second problem is that the discussed equilibria have all the players investing in the same project, which is understandable because of the linear project functions but practically unreasonable in conferences, though possible in other applications, such as sponsorship of large projects like Uber, Lyft, Facebook and VKontakte.

Comparing these models, we see from Theorems 1 and 4 that the success threshold allows ensuring that there exists a socially optimal equilibrium while the quota requires also assuming that the largest effort budget is less than the sum of the other ones times the ratio of the least to the most profitable project coefficients. In addition, comparing Theorems 3 and 6 shows that provided the smallest budget is at least a certain fraction of the largest one, choosing large enough a threshold or small enough a quota guarantees that an optimal profile will be an equilibrium. Unlike in the described cases, where success threshold seems stronger than quota, we notice that the second part of Theorem 6 actually contains an additional condition, relatively to the second part of Theorem 3, but since the second parts of these theorems refer to the case of a single agent being able to dominate everyone everywhere and all the projects being equally rewarding, this is less practical. To conclude the comparison, sometimes, choosing success threshold has more power, since choosing quota needs to assume an additional relation between the budgets, in order to guarantee that socially optimal equilibria exist. Intuitively, this stems from a quota needing an assumption on what the players are able to do to increase their utility, given the quota, while providing a success threshold can be done already with the budgets in mind.

Both a quota and a success threshold have a concentrating effect: equilibria where the agents contribute to less projects than without any of these conditions.

Many directions to expand the research exist. First, some common projects like papers and books have an upper bound on the maximal number of participants. Also a person has an upper bound on the maximal number of projects she can contribute to. The model should account for these bounds. Second, competition can be of many sorts. For instance, a project may need to have a winning coalition of contributors, in the sense of cooperative games. The fate of the projects that fail the competition can also vary; for example, their value can be distributed between the winning projects. We have extended the sufficiency results for existence from [13], and proven the necessity to be harder for analytical analysis. Simulations or other analytical approaches may be tried to delineate the set of Nash equilibria more clearly. Naturally, project functions do not have to be linear, so there is a clear need to model various non-linear functions. Such a more general model will make the conclusions on scientific investments, paper co-authorship, and the many other application domains more precise, and enable us to further improve the advice to participants as well as organizers. We can look at submitting a paper to a highly-ranked conference and reducing the conference level till the paper gets accepted as on a series of shared effort games with various quotas, success thresholds and participants. If we model the cost of each submission, then the question is to which conference to submit first.

Acknowledgments. This work has been supported by the project SHINE, the flagship project of DIRECT (Delft Institute for Research on ICT at Delft University of Technology). We thank anonymous reviewers for their comments.

A Omitted Proofs

We now prove Theorem 6.

Theorem 6. *Consider an equal θ-sharing game with $n \geq 2$ players with budgets $B_n \geq \ldots \geq B_2 \geq B_1$, $0 < \theta < 1$ (the order is w.l.o.g.), linear project functions with coefficients $\alpha_m = \alpha_{m-1} = \ldots = \alpha_{m-k+1} > \alpha_{m-k} \geq \alpha_{m-k-1} \geq \ldots \geq \alpha_1$ (the order is w.l.o.g) (see Footnote 2), and success threshold δ. Define $p \triangleq \left\lfloor \frac{\alpha_m \sum_{i \in N} B_i}{\delta} \right\rfloor$, as in Theorem 5.*

1. *If at least one of the following holds.*
   (a) $B_1 \geq k\theta B_n$, $k \leq p$ and $\frac{1}{n}\alpha_{m-k+1} \geq \alpha_{m-k}$,
   (b) $B_1 \geq p\theta B_n$, $k \geq p \geq 1$ and $\alpha_m B_n < \delta$;
   Then, there exists a pure NE and there holds: $\text{PoS} = 1$.
2. *Assume $B_{n-1} < \frac{\theta}{|\Omega|} B_n$, all the project functions are equal, i.e. $\alpha_m = \alpha_1$. Then, there exists a pure NE and* $\text{PoS} = 1$. *If, in addition, $\alpha_m B_n \geq \delta$, then* $\text{PoA} = \frac{B_n}{\sum_{i \in \{1,2,\ldots,n\}} B_i}$.

Proof. We first prove parts 1a and 1b. According to proof of parts 1 and 2 in Theorem 5, equally dividing all the budgets among $\min\{k, p\}$ steep projects is an NE. Therefore, $\text{PoS} = 1$.

Part 2 is proven as follows. Since all the players dividing their budgets equally between any $\min\{p, m\}$ projects constitutes an NE, we have $\text{PoS} = 1$.

To treat the PoA, we define the number of projects player n can make accepted on her own, $r \triangleq \left\lfloor \alpha_m \frac{B_n}{\delta} \right\rfloor$, and distinguish between the case where $m \leq r$ and $m > r$. If $m \leq r$, consider the profile where player n divides her budget equally between all the projects, while the other players contribute nothing at all. This is an NE, because all the projects are accepted, player n cannot increase her profit and any other player will be suppressed, if she contributes anything anywhere. On the other hand, if $m > r$, consider the profile where player n divides her budget equally between $m, m-1, \ldots, m-r+1$, while the other players contribute nothing at all. The only possible deviation is player $j < n$ contributing to a vacant project. However, we have $B_j \leq B_{n-1} < \frac{\theta}{|\Omega|} B_n < \theta\delta/\alpha_m \leq \delta/\alpha_m$. This means that the project would be unaccepted. Therefore, this is an NE.

Therefore, $\text{PoA} \leq \frac{\alpha_m B_n}{\alpha_m (\sum_{i \in N} B_i)}$. Since $\alpha_m B_n \geq \delta$, in any NE, player n receives at least $\alpha_m B_n$, and therefore, $\text{PoA} = \frac{B_n}{\sum_{i \in \{1,2,\ldots,n\}} B_i}$. ■

We finally prove Proposition 2.

Proposition 2. *For any success threshold $\delta \in [0, \alpha_m B_n]$ and any number of agents $n \geq 2$ and projects $m \geq 2$, there exists a game which possesses an NE, and a game which does not.*

Proof. For $\delta = 0$, which means for no threshold, the theorem follows from Theorem 3 from [13]. Therefore, we assume henceforth a positive success threshold.

A game that satisfies the conditions of Theorem 5 provides an example of the existence. Notice that the p they define is positive, since $\delta \leq \alpha_m B_N$.

To find a game that does not possess an equilibrium, let $\alpha_m = \alpha_1$ and let

$$B_n > \sum_{i=1}^{n-1} B_i, \tag{4}$$

$$B_1 = \ldots = B_{n-1} = \theta B_n \text{ and } \delta = \alpha B_n. \tag{5}$$

Because of the equality of all the project coefficients, of (4) and of the choice of the success threshold, in an equilibrium, all the agents with budgets at least θB_n (which are $1, \ldots, B_{n-1}$ here) will be together with n on the same single project. Then, agent n will deviate, contradictory to being in an equilibrium. ■

References

1. Anshelevich, E., DasGupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, 2004, pp. 295–304, October 2004
2. Anshelevich, E., Hoefer, M.: Contribution games in social networks. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6346, pp. 158–169. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15775-2_14
3. Bachrach, Y., Syrgkanis, V., Vojnovic, M.: Efficiency and the redistribution of welfare. Technical report, Microsoft Reserach, May 2012
4. Batchelor, G.K.: Preoccupations of a journal editor. J. Fluid Mech. **106**, 1–25 (1981)
5. Baye, M.R., Kovenock, D., de Vries, C.G.: The all-pay auction with complete information. Econ. Theory **8**(2), 291–305 (1996)
6. Bernstein, P.A., DeWitt, D., Heuer, A., Ives, Z., Jensen, C.S., Meyer, H., Özsu, M.T., Snodgrass, R.T., Whang, K.Y., Widom, J.: Database publication practices. In: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005, pp. 1241–1245. VLDB Endowment (2005)
7. Douceur, J.R.: Paper rating vs. paper ranking. SIGOPS Oper. Syst. Rev. **43**(2), 117–121 (2009)
8. Ganti, R., Ye, F., Lei, H.: Mobile crowdsensing: current state and future challenges. IEEE Commun. Mag. **49**(11), 32–39 (2011)
9. Harper, F.M., Xin Li, S., Chen, Y., Konstan, J.A.: Social comparisons to motivate contributions to an online community. In: de Kort, Y., IJsselsteijn, W., Midden, C., Eggen, B., Fogg, B.J. (eds.) PERSUASIVE 2007. LNCS, vol. 4744, pp. 148–159. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77006-0_20
10. Kleinberg, J., Oren, S.: Mechanisms for (mis)allocating scientific credit. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 529–538. ACM, New York (2011)
11. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49116-3_38

12. Papadimitriou, C.: Algorithms, games, and the internet. In: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, STOC 2001, pp. 749–753. ACM, New York (2001)
13. Polevoy, G., Trajanovski, S., de Weerdt, M.M.: Nash equilibria in shared effort games. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2014, pp. 861–868. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2014)
14. Roberson, B.: The colonel blotto game. Econ. Theory **29**, 1–24 (2006)
15. Schulz, A.S., Moses, N.S.: On the performance of user equilibria in traffic networks. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2003, pp. 86–87. Society for Industrial and Applied Mathematics, Philadelphia (2003)
16. Shapley, L.S.: A value for n-person games. Contribution to the theory of games. Ann. Math. Stud. **2**, 28 (1953)

Refining a Heuristic for Constructing Bayesian Networks from Structured Arguments

Remi Wieten[1(✉)], Floris Bex[1], Linda C. van der Gaag[1], Henry Prakken[1,2], and Silja Renooij[1]

[1] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
g.m.wieten@uu.nl

[2] Faculty of Law, University of Groningen, Groningen, The Netherlands

Abstract. Recently, a heuristic was proposed for constructing Bayesian networks (BNs) from structured arguments. This heuristic helps domain experts who are accustomed to argumentation to transform their reasoning into a BN and subsequently weigh their case evidence in a probabilistic manner. While the underlying undirected graph of the BN is automatically constructed by following the heuristic, the arc directions are to be set manually by a BN engineer in consultation with the domain expert. As the knowledge elicitation involved is known to be time-consuming, it is of value to (partly) automate this step. We propose a refinement of the heuristic to this end, which specifies the directions in which arcs are to be set given specific conditions on structured arguments.

Keywords: Bayesian Networks · Structured argumentation

1 Introduction

In recent years, efforts have been made to gain a better understanding of the relation between different normative frameworks for evidential reasoning, such as argumentative and probabilistic approaches [9]. Argumentative approaches are particularly suited for adversarial settings, where arguments for and against a specific conclusion are constructed from evidence. The inferences which are used to draw conclusions from evidence are generally defeasible, in that the conclusion of an argument does not universally hold given the evidence. Arguments can be attacked by other arguments; it can then be established which arguments are accepted and which are rejected. In current argumentative approaches, however, there is no emphasis on incorporating graded uncertainty.

In contrast, probabilistic approaches are well suited for handling graded uncertainty. In particular, Bayesian networks (BNs) [2,3] are powerful tools to this end. BNs are compact graphical models of joint probability distributions, which allow for evidence evaluation by calculating the probability of the truth

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 32–45, 2018.
https://doi.org/10.1007/978-3-319-76892-2_3

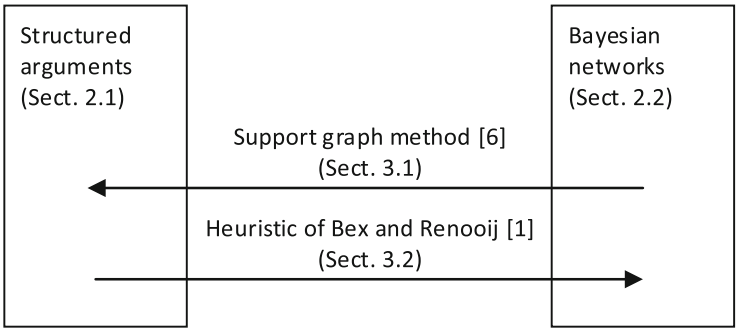


Fig. 1. Outline of Sects. 2 and 3 of this paper.

of a proposition of interest. However, BNs are generally difficult to construct; in fact, they are often constructed by modelers with the relevant mathematical background, called BN engineers, in consultation with a domain expert.

Recently, a heuristic for constructing BNs from structured arguments was proposed by Bex and Renooij [1]; in this paper, the heuristic will be referred to as the BR heuristic. The heuristic helps domain experts who are more accustomed to argumentation to transform their reasoning into a BN (cf. Fig. 1) and subsequently weigh their case evidence in a probabilistic manner. The focus of the BR heuristic lies on obtaining the graphical structure of the BN, called the BN graph, which captures the independence relations between the domain variables. While the underlying undirected graph, or skeleton, of the BN graph can be automatically constructed by following the BR heuristic, the heuristic prescribes that the arc directions should be set manually by a BN engineer in consultation with a domain expert. Although the heuristic further suggests that the commonly used notion of causality be taken as a guiding principle [3], the resulting graph still has to be verified and refined in terms of the independence relations it represents. This type of knowledge elicitation is known to be time-consuming [7], however, and moreover needs to be repeated for every adjustment to the original arguments. As a consequence, letting arc directions be set by a BN engineer is practically infeasible in investigative contexts such as police investigations, where evidence changes dynamically. It is, therefore, of value to investigate whether the process of setting arc directions can be (partly) automated.

Accordingly, in this paper we propose a refinement of the BR heuristic, which specifies the directions in which the arcs should be set in a BN graph under specific conditions on structured arguments. These conditions are identified by applying a method called the support graph method [6]. This method essentially works in the opposite direction of the BR heuristic, in that structured arguments are constructed from BNs (cf. Fig. 1). By applying the support graph method to BN graphs obtained with the BR heuristic, it is determined whether and under which conditions the original arguments are re-obtained. If the original arguments are not re-obtained from the thus constructed BN graph, it may be

concluded that this graph represents the original arguments in a different, possibly incorrect, way. Our refinement of the BR heuristic now ensures that BN graphs from which the original arguments are not returned by the support graph method are not constructed.

The paper is structured as follows. Sections 2 and 3 provide some preliminaries on structured argumentation, BNs, the support graph method and the BR heuristic. In Sect. 4, our refinement to the BR heuristic is proposed, based on observations from applying the support graph method. In Sect. 5, our findings are summarized and possible directions for future research are discussed.

2 Preliminaries

In this section, structured argumentation and BNs are briefly reviewed.

2.1 Structured Argumentation

A simplified version of the ASPIC+ framework for structured argumentation [4] is assumed throughout this paper. Let $\mathcal{L}$ be a non-empty propositional literal language with the unary negation symbol $\neg$. Informally, $\mathcal{L}$ contains the basic elements which can be argued about. Given a knowledge base $\mathcal{K} \subseteq \mathcal{L}$ of premises, arguments are constructed by chaining inference rules. These rules are defined over $\mathcal{L}$ and are defeasible, in that the conclusion of a defeasible rule does not universally hold given the premises, in contrast with the strict inferences of classical logic. Let $\mathcal{R}$ be a set of defeasible inference rules of the form $d\colon \phi_1, \ldots, \phi_n \Rightarrow \phi$, where $\phi_1, \ldots, \phi_n$ and ϕ are meta-variables ranging over well-formed formulas in $\mathcal{L}$. An argument A is then either: (1) ϕ if $\phi \in \mathcal{K}$, where the conclusion of the argument A, denoted by $\textsc{Conc}(A)$, is equal to ϕ; or (2) $A_1, \ldots, A_n \Rightarrow \phi$ with $\phi \in \mathcal{L} \setminus \mathcal{K}$, where $A_1, \ldots, A_n$ are arguments such that there exists a rule $\textsc{Conc}(A_1), \ldots, \textsc{Conc}(A_n) \Rightarrow \phi$ in $\mathcal{R}$. In the first case, $\textsc{Conc}(\mathrm{A})$ is an element from the knowledge base, while in the second case, $\textsc{Conc}(\mathrm{A})$ follows by applying a defeasible rule to the conclusion(s) of arguments $A_1, \ldots, A_n$, which are called the immediate sub-arguments of A. Generally, a sub-argument of an argument A is either A itself or an argument that is (iteratively) used to construct A. The smallest set of finite arguments which can be constructed from $\mathcal{L}$, $\mathcal{K}$ and $\mathcal{R}$ is denoted by $\mathcal{A}$. An argument graph of $\mathcal{A}$ then graphically displays the arguments in $\mathcal{A}$ and their sub-arguments. Figure 3a shows an example of an argument graph.

The general ASPIC+ framework further includes the notion of attack. Informally, an argument in $\mathcal{A}$ is attacked on one of its non-premise sub-arguments by another argument in $\mathcal{A}$ with the opposite conclusion of that sub-argument. Due to space limitations, the focus of the current paper lies on argument structures without attack relations.

2.2 Bayesian Networks

BNs [3] are graphical probabilistic models which are being applied in many different fields, including medicine and law [2]. A BN is a compact representation

of a joint probability distribution $\Pr(\mathbf{V})$ over a finite set of discrete random variables $\mathbf{V}$. The random variables are represented as nodes in a directed acyclic graph G, where each node[1] can take one of a number of mutually exclusive and exhaustive values; in this paper, we assume all nodes to be Boolean. A node A is a parent of another node B, called the child, in G if G contains an arc from A to B. The BN further includes, for each node, a conditional probability table, or CPT, given its parents; this table specifies the probabilities of the values of the node itself conditioned on the possible joint value combinations of its parents. A node is called instantiated iff it is fixed in a specific value. Given a set of instantiated nodes, conditional probability distributions over the other nodes in the network can be computed using probability calculus [3].

The BN graph captures the independence relations between its variables. Let a chain be defined as a simple path in the underlying undirected graph, or skeleton, of a BN graph. A node V is called a head-to-head node on a chain c if it has two incoming arcs on c. A chain c is blocked iff it includes a node V such that (1) V is an uninstantiated head-to-head node on c without instantiated descendants; (2) V is not a head-to-head node on c and is instantiated. In addition, instantiated end-points of the chain c, that is, instantiated nodes with at most one incoming or outgoing arc on c, serve to block the chain [5]. A chain is inactive if it is blocked; otherwise it is called active. Two nodes $A \neq B$ are called d-separated by a set of nodes $\mathbf{Z}$ if no active chains exist between A and B given instantiations of nodes in $\mathbf{Z}$. If two nodes are d-separated by $\mathbf{Z}$, then they are considered conditionally independent given $\mathbf{Z}$. We note that conditional independence thereby depends on the set of instantiated nodes [8].

An immorality in a BN graph is defined as a triple of nodes (A, B, C), where A and C are parents of B that are not directly connected by an arc. Two BNs are said to be Markov equivalent iff they share the same skeleton and immoralities. Markov equivalent networks constitute an equivalence class, for which Verma and Pearl [10] proved that any two elements represent the same independence relations over the variables involved. Arcs between nodes that are not involved in an immorality can thus be reversed without changing the represented independence relations as long as no new immoralities arise. Immoralities derive their importance from providing for intercausal reasoning [11]. Specifically, if the head-to-head node involved in an immorality is instantiated, an active chain arises between the parents of the node. These parents can be seen as different causes of the same effect modeled by the head-to-head node. If one of the causes is now observed, then the probability of the other cause being present as well can either increase, decrease or stay the same upon updating, depending on the probabilities in the CPT of the head-to-head node.

[1] The terms 'node' and 'variable' are used interchangeably.

3 Two Methods for Translating Between Structured Arguments and Bayesian Networks

In this section, the support graph method [6] and the BR heuristic [1] are reviewed; the support graph method is used to build structured arguments from BNs, while the BR heuristic is used to construct BN graphs from structured arguments.

3.1 The Support Graph Method

The support graph method, proposed by Timmer and colleagues [6], is a two-phase method for constructing argument structures from BNs. The method allows domain experts who are not familiar with BNs but are accustomed to argumentation to understand the knowledge and reasoning patterns captured by a BN. To this end, the method summarizes all reasoning chains from a set of evidence to a conclusion in a given BN.

In the first phase of the method, a directed graph called the support graph (SG) is constructed from a BN given a variable of interest V^*; in this SG, all reasoning chains in the BN ending in V^* are captured. The SG does not depend on specific instantiations, and can thus be re-used to build argument structures for different evidence. An SG is iteratively constructed, starting with a graph containing only V^*. New parents are added to existing nodes in the SG as new inference steps are identified in the BN. Three types of inference step are distinguished: (1) an inference step along an arc from a parent to a child; (2) an inference step along an arc from a child to a parent; and (3) an inference step between two parents in an immorality. The last type directly accommodates intercausal reasoning steps which occur between the parents of an immorality, and summarizes the inference from one parent of an immorality to another parent via the common child. In the constructed SG, V^* is the only node without children; every other node in the SG is an ancestor of V^*.

In the second phase of the support graph method, arguments are constructed from the SG for a given set of node instantiations. Given this evidence, the SG is pruned such that only paths remain that start in an instantiated node. From the thus pruned graph, arguments are constructed as follows. The logical language $\mathcal{L}$ is taken to consist of all literals which correspond to the values of the nodes in the BN; two literals $\phi, \psi \in \mathcal{L}$ negate each other iff ϕ and ψ correspond with the different values of the same node. Given the evidence, the knowledge base $\mathcal{K}$ consists of those literals in $\mathcal{L}$ that correspond with the values of the instantiated nodes. The defeasible rules in $\mathcal{R}$ are of the form $(N_1, o_1), \ldots, (N_k, o_k) \Rightarrow (N, o)$, where $N_1, \ldots, N_k$ are parents of the node N in the pruned SG and $o_1, \ldots, o_k, o$ are values of these nodes. From $\mathcal{L}$, $\mathcal{K}$, and $\mathcal{R}$, a set of arguments $\mathcal{A}$ is then constructed.

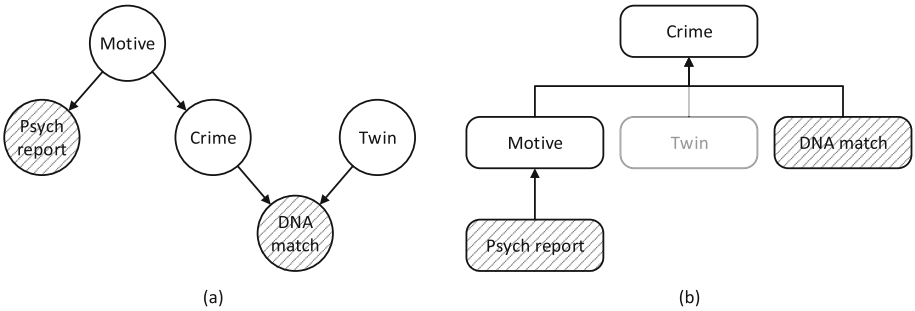


Fig. 2. A BN graph (a) and the corresponding SG for the variable of interest *Crime* (b); *Twin* is pruned from the SG as only *Psych report* and *DNA match* are instantiated.

Example 1. An example by Timmer and colleagues [6] from the legal domain is reviewed to demonstrate the support graph method. In the example, the BN graph from Fig. 2a[2] is constructed for a criminal case, in which we are interested in whether the suspect committed the crime, that is, whether *Crime* = *true*. Evidence for this possible conclusion would be the existence of a motive, which may be mentioned in a psychological report. A match between the suspect's DNA and DNA found at the crime scene would further support the proposition that the suspect committed the crime. This finding might also be explained, however, if the suspect had an identical twin. For the variable of interest *Crime*, the SG of Fig. 2b is obtained; the node *Twin* is directly added as a parent of *Crime*, as the triplet (*Crime*, *DNA match*, *Twin*) is an immorality in the BN graph. The literals in $\mathcal{L}$ are the possible values of all nodes in the BN graph, that is, $\mathcal{L}$ contains *crime*, $\neg$*crime*, *motive*, $\neg$*motive*, ... Now, if we assume that *Psych report* and *DNA match* are instantiated with the value *true* conform available evidence, and *Twin* is not instantiated, then the path starting at the node *Twin* is pruned from the SG. The knowledge base $\mathcal{K}$ then consists of *psych report* and *dna match*. Among the defeasible rules extracted from the pruned SG are d_1: *psych report* $\Rightarrow$ *motive* and d_2: *dna match*, *motive* $\Rightarrow$ *crime*. The arguments A_1: *psych report*, A_2: *dna match*, A_3: $A_1 \Rightarrow$ *motive*, and A_4: $A_2, A_3 \Rightarrow$ *crime* can then be constructed. Also the rules d_3: *psych report* $\Rightarrow \neg$*motive* and d_4: *dna match*, $\neg$*motive* $\Rightarrow \neg$*crime* are extracted from the SG, from which arguments A_5: $A_1 \Rightarrow \neg$*motive* and A_6: $A_2, A_5 \Rightarrow \neg$*crime* are constructed. These arguments have opposite conclusions of A_3 and A_4. □

It should be noted that, when using the support graph method, the reasons pro and con a given conclusion are not distributed over separate arguments, as is usual in argumentation, but are instead encapsulated in a single argument.

[2] In figures in this paper, circles are used in BN graphs, rectangles are used in argument graphs and rounded rectangles are used in SGs. Nodes and propositions corresponding to evidence are shaded. Capital letters are used for the nodes in BN graphs and SGs, and lowercase letters are used for propositions.

That is, all literals that are relevant for a specific proposition are taken as the premises of an argument for that proposition, which reflects the way in which Bayesian networks internally weigh all evidence.

For every argument that is returned from a BN by the support graph method, the method also returns an argument with the same 'structure' but with the opposite conclusion. Timmer and colleagues [6] employ a quantitative step to filter the set of arguments returned. As in the current paper the focus lies on the graphical structures of BNs and not on the modeled probability distribution, this quantitative step is not further discussed here.

3.2 The BR Heuristic for Constructing Bayesian Networks from Structured Arguments

Bex and Renooij [1] have proposed the BR heuristic for constructing BN graphs from structured arguments. This heuristic allows domain experts who are accustomed to argumentation to translate their reasoning expressed as arguments into a BN graph. This graph is then supplemented with CPTs to arrive at a fully specified BN for probabilistic inference over the original arguments. Focusing on argument structures in which no attack relations are present, from a given set of arguments $\mathcal{A}$ constructed from a logical language $\mathcal{L}$, knowledge base $\mathcal{K}$, and a set of defeasible rules $\mathcal{R}$, BN graphs are constructed as follows:

1. For every proposition $\phi \in \mathcal{L}$ used in $\mathcal{A}$, the BN graph includes a single node V such that $V = \mathit{true}$ corresponds to ϕ and $V = \mathit{false}$ corresponds to $\neg\phi$. For every $e \in \mathcal{K}$, the corresponding node is instantiated at the observed value.
2. For every defeasible rule $d\colon \phi_1, \ldots, \phi_n \Rightarrow \phi \in \mathcal{R}$ used in $\mathcal{A}$, a set of undirected edges between the node associated with ϕ and each of the nodes associated with $\phi_1, \ldots, \phi_n$ is created for inclusion in the BN graph.
3. The direction of the edges from the previous step is decided upon by a BN engineer in consultation with the domain expert, where a causal direction is chosen if possible, and an arbitrary direction otherwise. The resulting arcs are inserted in the BN graph.
4. The BN engineer verifies that the graph is acyclic and that all chains that should be active in the graph indeed are; if the graph does not yet exhibit these properties, appropriate arcs are removed or reversed, once more in consultation with the domain expert.

Example 2. A simple example is introduced to demonstrate the BR heuristic. The logical language, knowledge base and defeasible rules involved are $\mathcal{L} = \{p, \neg p, q, \neg q, r, \neg r\}$, $\mathcal{K} = \{p\}$ and $\mathcal{R} = \{p \Rightarrow q; q \Rightarrow r\}$. The constructed arguments are $\mathcal{A} = \{A_1\colon p;\ A_2\colon A_1 \Rightarrow q;\ A_3\colon A_2 \Rightarrow r\}$; the argument graph of $\mathcal{A}$ is depicted in Fig. 3a. Following steps 1 and 2 of the BR heuristic, the skeleton of the BN graph corresponding to this argument structure consists of nodes P, Q and R, with undirected edges between P and Q and between Q and R. Following step 3, one of the BN graphs of Fig. 3b–e is obtained, depending on how the arc directions are set. □

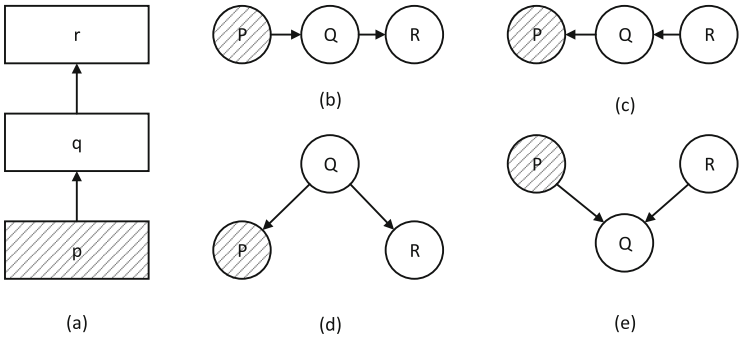


Fig. 3. An argument graph with arguments from p to r via q (a); the four corresponding BN graphs which can be constructed by following the BR heuristic (b–e).

For a given set of arguments $\mathcal{A}$, the skeleton of the BN graph is automatically constructed by following the first two steps of the BR heuristic. Step 3 then prescribes that the directions of the arcs should be set manually by a BN engineer in consultation with the domain expert, using the notion of causality as a guiding principle (see also [3]). For example, if the domain expert indicates for a defeasible rule d: $p \Rightarrow q$ that p is a typical cause of q, then the arc is set from node P to node Q. Since immoralities can result from following this guiding principle, the independence relations in the constructed BN graph should be verified manually, as prescribed by step 4 of the BR heuristic. This type of knowledge elicitation and verification is known to be a time-consuming and error-prone process in general [7]. Especially for larger or more densely connected BN graphs, it quickly becomes infeasible to verify all independence relations manually, as all possible chains for all possible combinations of instantiated variables need to be investigated. Moreover, the elicitation and verification needs to be repeated for every adjustment to the original argument graph. As this step is practically infeasible in investigative contexts, such as police investigations, in which the evidence for a case changes dynamically, the arc directions are preferably set (semi-)automatically.

4 Refining the BR Heuristic

We propose a refinement of step 3 of the BR heuristic, which specifies the directions in which arcs should be set in a BN graph under specific conditions on structured arguments. These conditions are identified from applying the support graph method. To this end, the arguments to which the BR heuristic is applied are compared to the arguments returned by the support graph method when applied to a BN graph constructed by steps 1–3 of the BR heuristic. In order to apply the support graph method, a variable of interest has to be chosen. In this paper, we assume that there is a single ultimate conclusion in the input argument graph, that is, a single argument that is not an immediate sub-argument of

another argument. The node corresponding to this ultimate conclusion is taken as the node of interest. We further assume that the input arguments for the BR heuristic are linked, in the sense that all premises relevant for a conclusion are encapsulated in a single argument; Fig. 5a shows an example of an argument graph with linked arguments only. Linked argument graphs are similar to the type of argument graphs that are returned by the support graph method.

When applying the support graph method to a BN graph constructed by steps 1–3 of the BR heuristic, a set of arguments is returned. This set may be different from the set of arguments that was used as input for the heuristic. As measures for the differences found, we distinguish between recall and precision, which for a given BN graph respectively measure the proportion of original arguments returned and the proportion of additional arguments returned. Formally, let $\mathcal{A}$ be the set of input arguments for the BR heuristic, let $\mathbf{B}$ be a BN graph constructed from $\mathcal{A}$ by steps 1–3 of the heuristic, and let $\mathcal{A}'$ be the set of arguments returned from $\mathbf{B}$ by the support graph method. We define the recall and precision of $\mathbf{B}$ as follows:

- $\mathit{Recall}(\mathbf{B}) = |\mathcal{A} \cap \mathcal{A}'|/|\mathcal{A}|$
- $\mathit{Precision}(\mathbf{B}) = |\mathcal{A} \cap \mathcal{A}'|/|\mathcal{A}'|$

where $\mathbf{B}$ has maximum recall and precision if these fractions are equal to 1.

In Sect. 4.1, we propose a refinement of the third step of the BR heuristic, which serves to increase the recall of constructed BN graphs. In Sect. 4.2, we address precision. As argued before, Timmer and colleagues [6] propose a quantitative step for filtering the set of arguments returned by the support graph method, which suggests that for improving the precision of constructed BNs, the CPTs need to be taken into account. As in this paper, the focus lies on the graphical structure of a BN, we propose a further refinement of the third step of the BR heuristic based on graphical considerations only.

4.1 Refining the BR Heuristic to Improve Recall

To illustrate how the BR heuristic can be refined such that BN graphs with higher recall are constructed, we revisit Example 2 from Sect. 3.2. By applying steps 1–3 of the heuristic to the argument graph of Fig. 3a, four possible BN graphs over the nodes P, Q and R were constructed, as shown in Figs. 3b–e. These graphs fall into two Markov equivalence classes; the first class consists of the BN graphs of Figs. 3b–d, and the second class consists of the graph of Fig. 3e. Timmer and colleagues [6] proved that for two Markov equivalent BNs and the same node of interest, the same SG is obtained. By applying the support graph method for the node of interest R, we now show that the recall of the original arguments from the BN graph in the second equivalence class is lower than that of the BN graphs in the first class. Since the logical language and knowledge base of the argument structure returned by the support graph method are derived from the BN skeleton, $\mathcal{L}'$[3] $= \{p, \neg p, q, \neg q, r, \neg r\}$ and $\mathcal{K}' = \{p\}$ are the same for all four BN graphs. For the

[3] The prime symbol is used to denote objects which result from applying the support graph method.

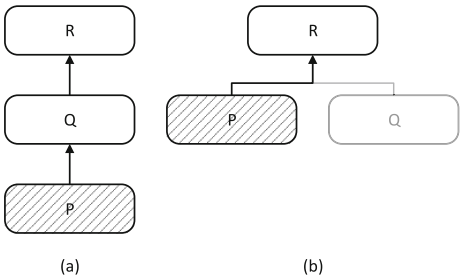


Fig. 4. The (pruned) SG obtained from the BN graphs of Figs. 3b–d (a), and the SG obtained from the BN graph of Fig. 3e (b), where Q is pruned as only P is instantiated.

graphs in the first equivalence class, the SG of Fig. 4a is obtained. The defeasible rules of the returned argument structure correspond to the arcs of this SG, that is, $\mathcal{R}' = \{p \Rightarrow q; p \Rightarrow \neg q; \neg p \Rightarrow q; \neg p \Rightarrow \neg q; q \Rightarrow r; q \Rightarrow \neg r; \neg q \Rightarrow r; \neg q \Rightarrow \neg r\}$. As $\mathcal{L} \subseteq \mathcal{L}'$, $\mathcal{K} = \mathcal{K}'$ and $\mathcal{R} \subseteq \mathcal{R}'$, all original arguments $A_1, A_2, A_3 \in \mathcal{A}$ are re-obtained from the SG. Therefore, the BN graphs of Figs. 3b–d have maximal recall.

For the BN graph in the second equivalence class, the SG of Fig. 4b is constructed. In this SG, node P is a direct parent of R and not of Q, as (P, Q, R) is an immorality. We recall that an SG is meant for constructing arguments for different sets of evidence. In the example, where just P is instantiated, node Q is pruned from the SG. The defeasible rules corresponding to this pruned SG are $\mathcal{R}' = \{p \Rightarrow r; p \Rightarrow \neg r; \neg p \Rightarrow r; \neg p \Rightarrow \neg r\}$ and the arguments which can be constructed are $A_1 \colon p$, $A_2' \colon A_1 \Rightarrow r$ and $A_2'' \colon A_1 \Rightarrow \neg r$. Timmer and colleagues [6] employ a quantitative step using the CPTs from the original BN to filter the set of constructed arguments; by this step, arguments A_2' and A_2'' are filtered out, as P and R are independent given that Q is not instantiated. The original arguments A_2 are A_3 are not returned by the support graph method. The recall of the BN graph from Fig. 3e is $\frac{1}{3}$, which is lower than that of the BN graphs in the first equivalence class. It therefore seems desirable to prohibit construction of this BN graph when using the BR heuristic.

Generalizing from the example, let $A_1, \ldots, A_n \in \mathcal{A}$, where A_i is an immediate sub-argument of A_{i+1} for all $i \in \{1, \ldots, n-1\}$, let $\textsc{Conc}(A_i) = p_i$, $p_1 \in \mathcal{K}$, and let p_n be the ultimate conclusion of the argument graph of $\mathcal{A}$. Further assume that no immorality (P_{i-1}, P_i, P_{i+1}) is formed for $i \in \{2, \ldots, n-1\}$ by steps 1–3 of the BR heuristic. As no immoralities (P_{i-1}, P_i, P_{i+1}) are present for $i \in \{2, \ldots, n-1\}$, upon constructing the SG for the node of interest P_n parents are added iteratively, that is, P_{n-1} is added as a parent of $P_n, \ldots, P_1$ is added as a parent of P_2. As P_1 corresponds to an instantiated variable, the path starting in P_1 is not pruned from the SG. The support graph method, therefore, returns the arguments $A_1, \ldots, A_n$, and the recall is maximal. On the other hand, if for a given $i \in \{2, \ldots, n-1\}$ an immorality (P_{i-1}, P_i, P_{i+1}) would be formed by steps 1–3 of the BR heuristic, then an SG would result in which P_{i+1} is an ancestor of P_n. As P_{i-1} is directly added as a parent of P_{i+1}, the argument A_i would not be returned, and the recall would not be maximal.

Based on the above observations, the following refinement of step 3 of the BR heuristic is proposed:

3′. Let $A_1, \ldots, A_n \in \mathcal{A}$, where A_i is an immediate sub-argument of A_{i+1} for any $i \in \{1, \ldots, n-1\}$ and where $\textsc{Conc}(A_i) = p_i$. Then, the directions of the arcs are set such that no immoralities (P_{i-1}, P_i, P_{i+1}) are formed for any $i \in \{2, \ldots, n-1\}$. Taking this constraint into account, the directions of the (remaining) arcs are set by a BN engineer in consultation with the domain expert, where a causal direction is chosen if possible.

4.2 A Further Refinement of the BR Heuristic

While in the previous section, simple chains in an argument structure were shown to be best translated in the BN graph by a chain without any immoralities, we now focus on argument structures that do enforce immoralities in the BN graph and propose a further refinement of the refined third step of the heuristic.

Example 3. We consider the linked argument graph of Fig. 5a. The logical language, knowledge base and defeasible rules involved are $\mathcal{L} = \{p, \neg p, q, \neg q, r, \neg r, s, \neg s, t, \neg t\}, \mathcal{K} = \{p, q\}$, and $\mathcal{R} = \{p \Rightarrow r; p, q \Rightarrow s; r, s \Rightarrow t\}$; the constructed arguments are $\mathcal{A} = \{A_1 : p; A_2 : A_1 \Rightarrow r; A_3 : q; A_4 : A_1, A_3 \Rightarrow s; A_5 : A_2, A_4 \Rightarrow t\}$. Steps 1 and 2 of the BR heuristic result in the BN skeleton of Fig. 5b. In order to

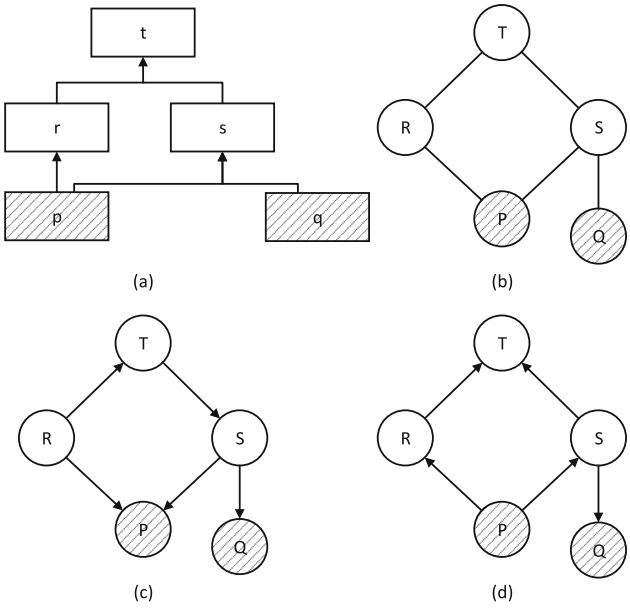


Fig. 5. An argument graph (a) and the corresponding BN skeleton that is constructed by the BR heuristic (b); a corresponding BN graph with the immorality (R, P, S) (c) and a BN graph with the immorality (R, T, S) (d).

obtain an acyclic directed graph from this skeleton, at least one immorality has to be created in the subgraph induced by the nodes P, R, S and T.

According to the refined third step of the BR heuristic, an immorality (T, R, P) should not be formed, as $A_1 : p$ is an immediate sub-argument of $A_2 : A_1 \Rightarrow r$, which in turn is an immediate sub-argument of $A_5 : A_2, A_4 \Rightarrow t$. Similarly, the immorality (T, S, P) should not be formed. Now, the equivalence class of BN graphs is considered which includes just the immorality (R, P, S); the BN graph depicted in Fig. 5c is an element of this class. With T as the node of interest, the SG of Fig. 6 is obtained from this graph. The logical language and knowledge base corresponding to this SG are $\mathcal{L}' = \{p, \neg p, q, \neg q, r, \neg r, s, \neg s, t, \neg t\}$ and $\mathcal{K}' = \{p, q\}$, matching those of the original argument graph. The set of defeasible rules $\mathcal{R}$ corresponding to the SG includes the rules $p, q \Rightarrow s$; $p, s \Rightarrow r$; $p \Rightarrow r$; $p, q, r \Rightarrow s$ and $r, s \Rightarrow t$. Among the arguments which can be constructed from the SG are $A_1 : p$, $A_2 : A_1 \Rightarrow r$, $A_3 : q$, $A_4 : A_1, A_3 \Rightarrow s$, $A_5 : A_2, A_4 \Rightarrow t$, $A'_2 : A_1, A_4 \Rightarrow r$, $A'_4 : A_1, A_2, A_3 \Rightarrow s$, and $A'_5 : A'_2, A'_4 \Rightarrow t$. While the recall of the BN graphs from Fig. 5c is maximal, the precision is not; more specifically, the returned arguments A'_2, A'_4 and A'_5 were not in the original argument set $\mathcal{A}$.

Now, the equivalence class of BN graphs with just the immorality (R, T, S) is addressed; the BN graph depicted in Fig. 5d is an element of this class. From this BN graph, again the SG of Fig. 6 is constructed for the node of interest T, and thus the same arguments as above are returned. While the precision of the BN graph of Fig. 5d is equal to that of the BN graph from Fig. 5c, we note that the nodes R and S are conditionally independent given the evidence for $\mathbf{Z} = \{P, Q\}$ in the former graph, that is, in the BN graph with just the immorality (R, T, S). The immediate sub-argument A_4 of A'_2 and the immediate sub-argument A_2 of A'_4, therefore, appear to be irrelevant, as the associated reasoning is non-existent in this BN graph. As noted before, Timmer and colleagues [6] employ a quantitative step to filter the set of arguments returned by

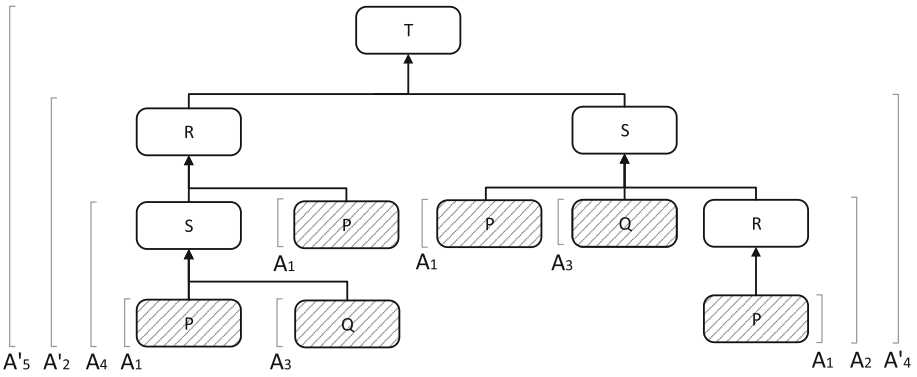


Fig. 6. The SG corresponding to the BN graph of Figs. 5c and d, with T as the node of interest; the SG is annotated with some of the possible arguments which can be extracted from it.

the support graph method; specifically, as the nodes R and S are conditionally independent given the evidence in the BN graph in Fig. 5d, A_4 and A_2 are filtered out as immediate sub-arguments of A'_2 and A'_4 respectively. Building on the conditional independence relations that can be inferred from the BN graph given the set of instantiated nodes, however, irrelevance of A_4 and A_2 as immediate sub-arguments of A'_2 and A'_4 can be decided upon by graphical considerations only, without involving the CPTs of the nodes. □

Based on the above example, we propose to set the directions of arcs in a BN skeleton such that no instantiated head-to-head nodes or head-to-head nodes with instantiated descendants are formed, as such head-to-head nodes may introduce unwarranted dependence relations. More specifically, the following refinement of step 3′ of the BR heuristic is proposed, which fully specifies the directions of the arcs in a BN graph corresponding to a set of arguments $\mathcal{A}$:

3″. The directions of the arcs in a BN graph are set in the same direction as the arcs in the argument graph, that is, if A is an immediate sub-argument of B, then an arc should be drawn from the node corresponding to $\textsc{Conc}(A)$ to the node corresponding to $\textsc{Conc}(B)$.

We note that step 3″ is a further refinement of step 3′, as none of the immoralities (P_{i-1}, P_i, P_{i+1}) mentioned in that step are formed if arcs are set in the same direction as in the argument graph. By step 3″, arcs are guaranteed to be set such that head-to-head nodes are not instantiated and do not have instantiated descendants, as the premise arguments in the argument graph, and hence the instantiated nodes in the BN graph, only have outgoing arcs. Finally, we note that step 3″ is not a strict specification of the directions of the arcs in a BN graph; directions can possibly be reversed, given that an element from the same Markov equivalence class as specified by step 3″ is obtained.

5 Conclusion and Future Research

In this paper, we have proposed a refinement of the heuristic of Bex and Renooij [1] for constructing BN graphs from structured arguments. This heuristic is aimed at aiding domain experts who are accustomed to argumentation to transform their reasoning into BNs and subsequently weigh their case evidence in a probabilistic manner. Our refinement consists of fully specifying the directions in which arcs should be set in a BN graph for a given argument structure without attack relations; more specifically, when employing the refined heuristic for a set of arguments $\mathcal{A}$, the directions of the arcs in the BN graph are set in the same direction as the arcs in the original argument graph of $\mathcal{A}$. By our refined heuristic, BN graphs with maximal recall are constructed, that is, the original arguments are returned by applying the support graph method to the constructed BN graphs. Furthermore, our refined heuristic prevents the creation of direct intercausal dependence relations between variables in the BN graph that did not exist between the corresponding propositions in the original argument

graph. In the near future, we will evaluate the heuristic in practice by establishing, for example, the extent to which the automatically derived arc directions match the perceived real-world causality or the judgments of domain experts.

In this paper, we focused on improving the recall of BN graphs constructed by the BR heuristic. In our future research, we will address the construction of BN graphs with increased precision. Furthermore, we will extend our research to a more general framework of argumentation [4], not restricting ourselves to linked argument graphs without attack relations.

References

1. Bex, F., Renooij, S.: From arguments to constraints on a Bayesian network. In: Baroni, P., Gordon, T.F., Scheffler, T., Stede, M. (eds.) Computational Models of Argument: Proceedings of COMMA 2016, pp. 95–106. IOS Press, The Netherlands (2016)
2. Fenton, N., Neil, M.: Risk Assessment and Decision Analysis with Bayesian Networks. CRC Press, Boca Raton (2012)
3. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer Verlag, Berlin (2007)
4. Prakken, H.: An abstract framework for argumentation with structured arguments. Argum. Comput. **1**(2), 93–124 (2010)
5. Shachter, R.D.: A graph-based inference method for conditional independence. In: D'Ambrosio, B.D., Smets, P., Bonissone, P.P. (eds.) Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, pp. 353–360. Morgan Kaufmann Publishers Inc., San Mateo (1991)
6. Timmer, S.T., Meyer, J.-J.C., Prakken, H., Renooij, S., Verheij, B.: A two-phase method for extracting explanatory arguments from Bayesian networks. Int. J. Approx. Reason. **80**, 475–494 (2017)
7. van der Gaag, L.C., Helsper, E.M.: Experiences with modelling issues in building probabilistic networks. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 21–26. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45810-7_4
8. van der Gaag, L.C., Meyer, J.-J.C.: The dynamics of probabilistic structural relevance. In: van der Gaag, L.C., Meyer, J.-J.C. (eds.) Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96), pp. 145–156. Utrecht University, Utrecht (1996)
9. Verheij, B., Bex, F., Timmer, S.T., Vlek, C.S., Meyer, J.-J.C., Renooij, S., Prakken, H.: Arguments, scenarios and probabilities: connections between three normative frameworks for evidential reasoning. Law Probab. Risk **15**(1), 35–70 (2015)
10. Verma, T., Pearl, J.: Equivalence and synthesis of causal models. In: Bonissone, P.P., Henrion, M., Kanal, L.N., Lemmer, J.F. (eds.) Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, pp. 255–270. Elsevier Science Inc., New York (1991)
11. Wellman, M.P., Henrion, M.: Explaining "explaining away". IEEE Trans. Pattern Anal. Mach. Intell. **15**(3), 287–292 (1993)

Reciprocation Effort Games

Gleb Polevoy[1(✉)] and Mathijs de Weerdt[2]

[1] University of Amsterdam, Amsterdam, The Netherlands
g.polevoy@uva.nl
[2] Delft University of Technology, Delft, The Netherlands
m.m.deweerdt@tudelft.nl

Abstract. Consider people *dividing their time and effort* between friends, interest clubs, and reading seminars. These are all *reciprocal interactions,* and the reciprocal processes determine the utilities of the agents from these interactions. To advise on efficient effort division, we determine the existence and efficiency of the Nash equilibria of the game of allocating effort to such projects. When no minimum effort is required to receive reciprocation, an equilibrium always exists, and if acting is either easy to everyone, or hard to everyone, then every equilibrium is socially optimal. If a minimal effort is needed to participate, we prove that not contributing at all is an equilibrium, and for two agents, also a socially optimal equilibrium can be found. Next, we extend the model, assuming that the need to react requires more than the agents can contribute to acting, rendering the reciprocation imperfect. We prove that even then, each interaction converges and the corresponding game has an equilibrium.

1 Introduction

In many real-world situations people invest effort in several interactions, such as in discretionary daily activities [16], daily communication between school pupils, sharing files over networks, or in business cooperation. In such an interaction, people tend to reciprocate, i.e., react on the past actions of others (sometimes only if a certain minimum effort is invested) [10,12]. For example, users of various social networks (Facebook, VKontakte) repeatedly interact in those projects (networks). To recommend how to divide one's limited efforts efficiently, we aim to predict stable strategies for these settings and estimate their efficiency. We study settings with and without a threshold for minimum effort.

Dividing a budget of effort is studied in *shared effort games* [4]. In these games players contribute to various projects, and given their contributions, each project attains a value, which is subsequently divided between the contributors. In order to support decisions regarding individually and publicly good stable strategy profiles in these games, the social welfare (total utility) of strategy profiles is important, and in particular of Nash equilibria (NE). For this, the

G. Polevoy—Most of this work was done at Delft University of Technology.

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 46–60, 2018.
https://doi.org/10.1007/978-3-319-76892-2_4

price of anarchy (PoA) [15], and stability (PoS) [1,23] are the most famous efficiency measures. The price of anarchy is the ratio of the least social welfare in an equilibrium to the optimal social welfare, and the price of stability is the ratio of the social welfare in a best NE to the optimal social welfare.

Bachrach et al. [4] bound the price of anarchy, but only when a player obtains at least a constant share of her marginal contribution to the project's value; this does not hold for a positive participation threshold. Polevoy et al. [20] have analyzed the Nash equilibria, and price of anarchy and stability also in the case with a threshold. When the threshold is equal to the highest contribution, such shared effort games are equivalent to all-pay auctions. In all-pay auctions, only one contributor benefits from the project. Its equilibria are analyzed by Baye et al. [5] and many others. Anshelevich and Hoefer [2] study graph nodes contributing to edges, which are minimum effort projects. In the literature, the utilities are based on the project values, which are directly defined by the contributions, such as in contributions to online communities, Wikipedia, political campaigns [25], and paper co-authorship [14]. Unlike the existing literature, our paper assumes contributions to the projects determine the interactions, which define utility.

We now review the reciprocation models. Existing models of reciprocation often consider why reciprocation has emerged. The following works consider the emergence of reciprocation. Axelrod [3] studies and motivates direct evolution of reciprocal behavior. Others consider a more elaborate evolution, like Bicchieri's work on norm emergence [6, Chap. 6] or [27]. Trivers [26] describes how altruism-related emotions like guilt and suspicion have evolved. There exist also other approaches to the nature of reciprocation, such as the *strong reciprocation* [11]. Works like [8,10,22] assume the reciprocal behavior and analyze the development of certain interactions, modeling them as appropriate games.

With a model inspired by works on arms races [7,28] and spouses' interaction [13], Polevoy et al. [21] formally analyze lengthy repeated reciprocation and show convergence. They define an action on an agent as a convex combination[1] between one's own last action, the considered other agent's and all the other agents' last actions. They call this the *floating* reciprocation attitude.

The main contributions of this paper comprise of the analysis of a unifying model of shared effort games with reciprocal projects and creating a basis for further analysis. We define two games: one without a threshold, and another one with a threshold. In the second game, those who are below the threshold in an interaction, are not allowed to participate in the respective interaction. We identify when Nash equilibria exist and find the prices of anarchy and stability. In addition to the main part, where the initial actions are fully reciprocated by the reciprocal agents [26], we model the situation when the budget of an agent to invest in the various projects may fall short of satisfying the requirements of every reciprocal interaction. This forces the agent to curb her investments in some interactions, complicating the process, but we prove it still converges, and therefore, generalizing the definitions to that case is well-defined. We also prove that the corresponding reciprocation effort game and its exclusive thresholded

[1] A combination is convex if it has nonnegative weights that sum up to 1.

version have an equilibrium. We consider only pure equilibria throughout the paper, even when we do not mention this explicitly. Since the strategies include all the ways to divide budget among the interactions, the set of pure strategies is already uncountably infinite.

The model of several reciprocal interactions is given in Sect. 2. Section 3 characterizes the equilibria and their efficiency in a game without a threshold. Then, we analyze the game with a threshold in Sect. 4. We prove the convergence of an interaction with insufficient budgets and the NE existence in Sect. 5. Section 6 concludes and outlines new research directions.

2 Model

This section models dividing effort between reciprocal interactions. Adopting the reciprocation model from [21] and the inspired by shared effort games models from [4,20], we define a *reciprocation effort game*. First, we define a reciprocal process and the agents' utilities in this process. Next, we define a reciprocation effort game, where agents divide their effort budgets between several such processes. We define a thresholded variation on this game, to model the minimal required investment, in Sect. 4.

We begin with the reciprocation model, based on models for arms race and arguments. Given agents $N = \{1, \dots, n\}$, at any time $t \in T \triangleq \{0, 1, 2, \dots\}$, every agent acts on any other agent. The action by agent $i \in N$ on another agent $j \in N$ at moment t is characterized by its weight, denoted by $\mathrm{act}_{i,j}(t) \colon T \to \mathbb{R}$. Since only the weight of an action is relevant, we usually write "action" while referring to its weight. For example, the weights of the actions of helping, nothing, or insulting are in the decreasing order.

In order to define how agents reciprocate, we need the following notation. The kindness of agent i, constant for a given reciprocal process, is denoted by $k_i \in \mathbb{R}$. Agent i's kindness models i's inherent inclination to act on any other agent: the larger the kindness, the kinder the agent acts; in particular, it determines the first action of an agent, before the others have acted. We model agent i's inclination to mimic another agent's action and the actions of all the other participants in the project by reciprocation coefficients $r_i \in [0, 1]$ and $r'_i \in [0, 1]$ respectively, both staying constant for all interactions. r_i is the fraction of $\mathrm{act}_{i,j}(t)$ that is determined by the previous action of j upon i, and r'_i is the fraction that is determined by $\frac{1}{n-1}$th of the total action on i by all the other agents at the previous time. Fractions sum up to 1, thus $r_i + r'_i \leq 1$. We denote the total received action from all the other agents at time t by $\mathrm{got}_i(t) \colon T \to \mathbb{R}$; formally, $\mathrm{got}_i(t) \triangleq \sum_{j \in N} \mathrm{act}_{j,i}(t)$.

We now define the actions. At time 0, there is nothing to react to, so the kindness determines the action: $\mathrm{act}_{i,j}(0) \triangleq k_i$.

Definition 1. *At any positive time t, agent i's action is a weighted average of her own last action (inertia), of that of the other agent j (direct reaction)*

and of the total action of all the other agents divided over all the others (social reaction):

$$\underset{i,j}{\mathrm{act}}(t) \triangleq (1 - r_i - r'_i) \cdot \underset{i,j}{\mathrm{act}}(t-1) + r_i \cdot \underset{j,i}{\mathrm{act}}(t-1) + r'_i \cdot \frac{\mathrm{got}_i(t-1)}{n-1}.$$

We have defined how agents reciprocate. An agent's *utility* from a given reciprocation project at a given time is the action one receives minus effort to act, following [19]. This is classical (see, for example, the quasilinear preferences of auction theory [17, Chap. 9.3]). Formally, define the utility of agent i at time t, $u_{i,t} \colon \mathbb{R}^{n-1} \times \mathbb{R}^{n-1} \to \mathbb{R}$, as

$$u_{i,t}\left(\left\{\underset{i,j}{\mathrm{act}}(t)\right\}_{i,j\in N}, \left\{\underset{j,i}{\mathrm{act}}(t)\right\}_{i,j\in N}\right) \triangleq \sum_{j\in N} \underset{j,i}{\mathrm{act}}(t) - \beta_i \sum_{j\in N} \underset{i,j}{\mathrm{act}}(t)$$

where the constant $\beta_i \in \mathbb{R}$ is the importance of performing actions relatively to receiving them for i's utility. The personal price of acting is higher, equal or lower than of receiving an action if β_i is bigger, equal or smaller than 1, respectively. The minus in front of i's actions subtracts the effort of acting from one's utility (unless β_i is negative, where that is added). Since the presence of negative actions would mess up this logic (since negative actions would still take effort while increasing the above expression), we assume that actions are always non-negative, which occurs if and only if all kindness values are non-negative. We can have negative influence, but we assume having added large enough a constant to all the actions, to avoid negative actions.

Every such interaction converges, as shown in [21]. To model the utility in the long run, we define the asymptotic utility, or just the *utility*, of agent i, as the limit of her utilities as the time approaches infinity. In formulas, $u_i \colon \left(\mathbb{R}^{n-1}\right)^{\infty} \times \left(\mathbb{R}^{n-1}\right)^{\infty} \to \mathbb{R}$, as $u_i\left(\bigcup_{t'=0}^{\infty}\{\mathrm{act}_{i,j}(t), \mathrm{act}_{j,i}(t)\}\right) \triangleq \lim_{t\to\infty} u_{i,t}\left(\mathrm{act}_{i,j}(t), \mathrm{act}_{j,i}(t)\right)$. This is the utility we consider here. This definition of the utility of a process is equivalent to the discounted sum of utilities when the discounting is slow enough. The proof is omitted for the lack of space.

We now define a *reciprocation effort game*. Our agents N participate in m interactions $\Omega = \{1, 2, \ldots, m\}$. Each of the m interactions is what we have defined till now, with its own kindness values and actions. The kindness, the actions, the total received action, and the utility in a concrete interaction $\omega \in \Omega$ will be denoted, when the concrete interaction is important, by $(k_i)_\omega$ and $(\mathrm{act}_{i,j}(t))_\omega$, $(\mathrm{got}_i(t))_\omega$, and $(u_{i,t})_\omega$ or $(u_i)_\omega$, respectively. Each player's strategies are the possible contributions to the interactions at time zero (the further contributions are determined by the reciprocation and not by the player). A contribution goes to the whole interaction, not to a particular action on another agent, but it determines the kindness values of the interactions as follows.

Player i's kindness at reciprocal interaction ω is determined by her contribution to that interaction at time zero, called just "the contribution", divided by the number of other agents who participate in the interaction at ω, accounting for acting on them. This means that i's kindness at interaction j is $\frac{x_j^i(0)}{n-1}$.

Therefore, the sum of all the actions of agent i at time $t = 0$ is equal to her contributions to all the reciprocation projects, which are bounded by her budget b_i. The contribution of player $i \in N$ to interaction project $\omega \in \Omega$ at a general time $t \in T$ is defined as the sum of her actions in that interaction at that time, i.e. $x^i_\omega(t) \triangleq \sum_{j \in N \setminus \{i\}} (\text{act}_{i,j}(t))_\omega$.

An agent contributes something in the beginning of a reciprocation, and from that time on the reciprocation "automatically" uncurls according to Definition 1. We assume that not only the sum of the contributions at $t = 0$, but also the sum of the contributions at any time $t > 0$ is within the acting agent's budget. Each player i has a normal budget $b_i > 0$ (or just a budget) to contribute from at $t = 0$ and an extended budget $B_i \geq b_i$ that can be used when the actions are required by the reciprocation process at $t > 0$, perhaps resulting in a higher summarized contribution than the voluntarily chosen at $t = 0$. We differentiate between these two budgets, since the need to reciprocate can urge people to act more actively [11], and we assume that B_is are high enough to allow reciprocation.

Formally, the strategy space of player i consists of her contributions (at time zero), determining her kindness values at the interactions, $\left\{ x^i = (x^i_\omega)_{\omega \in \Omega} \in \mathbb{R}_+^{|\Omega|} \,|\, \sum_{\omega \in \Omega} x^i_\omega \leq b_i \right\}$. As mentioned, a "contribution" always means the contribution at $t = 0$. Since the strategy profile $x = (x^i)_{i \in N}$ determines all the interactions, the above defined utilities in a reciprocal interaction, namely $(u_{i,t})_\omega$ and $(u_i)_\omega \triangleq \lim_{t \to \infty} (u_{i,t})_\omega$, are also functions of x. The utility $u_i(x)$ of a player $i \in N$ in the game is defined to be the sum of the utilities it obtains from the various projects, $u_i(x) \triangleq \sum_{\omega \in \Omega} (u_i(x))_\omega$, completing the definition of a *reciprocation effort game*.

An agent does not have to use up all her budget, so that the inequality $\sum_{\omega \in \Omega} x^i_\omega \leq b_i$ may be strict. The strategies of all the players except i are denoted x^{-i}. We denote the vector of all the contributions by $x = (x^i_\omega)^{i \in N}_{\omega \in \Omega}$.

We now give a concrete example of the model.

Example 1. People choose between going to an interest club, meeting friends, or going to a scientific reading seminar, as illustrated in Fig. 1. A player first decides on how much she wants to invest in each interaction, determining her kindness in each one of them. Subsequently, she reciprocates. Each of these projects is an interaction; for instance, in an interest club, a positive action

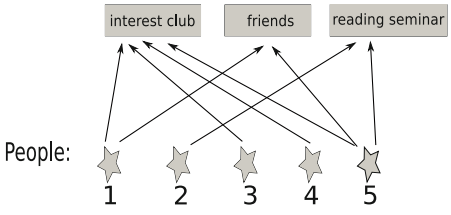


Fig. 1. People divide their own effort between interactions.

can be supporting another person, while showing contempt would be negative. Interacting, a person continues her previous course of action, represented by $(1 - r_i - r'_i) \cdot \mathrm{act}_{i,j}(t-1)$ in Definition 1, reacts on the other person's previous action, represented by $r_i \cdot \mathrm{act}_{j,i}(t-1)$, and reacts on the social climate, for which $r'_i \cdot \frac{\mathrm{got}_i(t-1)}{n-1}$ stands.

For the sake of efficiency analysis, we remind that the social welfare is defined as $\mathrm{SW} \triangleq \sum_{i \in N} u_i(x)$, and the prices of anarchy [15] and stability [1,23] are defined as $\frac{\min\{\mathrm{SW}(x) | x \text{ is an NE}\}}{\max\{\mathrm{SW}(x) | x \text{ is a strategy}\}}$ and $\frac{\max\{\mathrm{SW}(x) | x \text{ is an NE}\}}{\max\{\mathrm{SW}(x) | x \text{ is a strategy}\}}$, respectively. We define $0/0$ to be 1, because 0 from 0 means no loss occurs in the social welfare in the equilibria.

Polevoy et al. [21] prove the following theorem.

Theorem 1. *In an interaction, where for any agent i, $r'_i > 0$ and at least one agent i has $r_i + r'_i < 1$, for all pairs of agents $i \neq j$, the limit $\lim_{t \to \infty} \mathrm{act}_{i,j}(t)$ exists. The convergence is geometrically fast (exponential). All these limits are equal to each other and it is a convex combination of the kindness values, namely*

$$L = \frac{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \cdot k_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \right)}. \tag{1}$$

3 Reciprocation Effort Game Without a Threshold

We first completely analyze existence of NE, and then we find all the prices of anarchy and stability. This theorem characterizes the existence of equilibria.

Theorem 2. *Assume that for any agent i, $r'_i > 0$, and in addition, either $n > 2$ or $r_1 + r'_1 + r_2 + r'_2 < 2$. The set of all the NE is exactly all the strategy profiles where every agent with $\beta_i < 1$ somehow divides all her budget among the projects $\{1, \ldots, m\}$, and every agent with $\beta_i > 1$ contributes nothing. These strategies are also dominant. In particular, there always exists an NE.*

Proof. Consider an arbitrary player l, and let her strategy (her contributions[2]) be $x^l = (x^l_1, \ldots, x^l_m)$. By Formula(s) (1), the limit of the actions at (project) interaction j is

$$\left(\frac{\left(\frac{1}{r_l + r'_l} \cdot (x^l_j) \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \right)} + C_j \right),$$

where C_j is independent of l's strategy. This is both given and received by an agent w.r.t. the $n-1$ other agents, so we need to multiply the limit by $(n-1)(1-\beta_l)$. Summarizing, agent j's utility from this strategy is

$$(n-1)(1-\beta_l) \left(\frac{\left(\frac{1}{r_l + r'_l} \cdot (x^l_1 + \ldots + x^l_m) \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \right)} + C \right),$$

[2] Contributions by default refer to the contributions at time zero.

for C that is independent of l's strategy. Therefore, if $\beta_l < 1$, then l's strategy is a best response to others' strategies if and only if l arbitrarily divides all her budget among the projects $\{1, \ldots, m\}$. On the other hand, if $\beta_l > 1$, then a strategy is a best response if and only if all the contributions are zero. This is true for every agent l, proving that this is an NE. Since each agent is independent of the others, these strategies are also dominant. ■

The possible variations in an NE profile are what the agents with $\beta = 1$ do. This is important for analyzing the efficiency of the NE.[3] To analyze efficiency, we define: $N^{<} \triangleq \{i \in N : \beta_i < 1\}$, $N^{\leq} \triangleq \{i \in N : \beta_i \leq 1\}$, $N^{=} \triangleq \{i \in N : \beta_i = 1\}$. We now analyze the efficiency of the most and the least efficient equilibria, comparing their social welfare to the maximum possible social welfare.

Proposition 1. *Under the assumptions of Theorem 2, if* $(n > \sum_{i \in N} \beta_i)$*, we have* $\text{PoA} = \frac{\sum_{i \in N^{<}} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}$*, and the* PoS *is given by the same expression, where we use* $N^{\leq}$ *instead of* $N^{<}$*. Consequently, if* $(n = \sum_{i \in N} \beta_i)$*, we have* $\text{PoA} = \text{PoS} = 1$*. If* $(n < \sum_{i \in N} \beta_i)$*, then:*

If $N^{<} \neq \emptyset$*, then we have* $\text{PoA} = \text{PoS} = -\infty$.
If $N^{<} = \emptyset$**, but** $N^{\leq} \neq \emptyset$*, then* $\text{PoA} = -\infty$*, but* $\text{PoS} = 1$.
If $N^{\leq} = \emptyset$*, then* $\text{PoA} = \text{PoS} = 1$.

The proof compares the possible social welfare in equilibria with the optimum social welfare.

Proof. The possible social welfare values that an NE can achieve are exactly

$$(n-1)(n - \sum_{i \in N} \beta_i) \frac{\sum_{i \in N^{<}} \left(\frac{1}{r_i + r'_i} \cdot b_i \right) + \sum_{i \in N^{=}} \left(\frac{1}{r_i + r'_i} \cdot x^i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \right)},$$

where $0 \leq x^i \leq b_i$. The optimum social welfare is

$$(n-1)(n - \sum_{i \in N} \beta_i) \frac{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \right)}$$

if $(n > \sum_{i \in N} \beta_i)$, and 0 otherwise.

Thus, if $(n > \sum_{i \in N} \beta_i)$, we have

$$\text{PoA} = \frac{\sum_{i \in N^{<}} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)} \text{ and } \text{PoS} = \frac{\sum_{i \in N^{\leq}} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r'_i} \cdot b_i \right)}.$$

[3] $\beta_i > 1$ implies negative utilities that sometimes result in negative PoA and PoS.

If $(n = \sum_{i \in N} \beta_i)$, we have PoA = PoS = 1, since the social welfare is always zero, and we define here $0/0 = 1$.

If $(n < \sum_{i \in N} \beta_i)$, then we may get negative social welfare, since zero is optimal, while some NE yield a negative social welfare. Concretely, we have the following subcases:

If $N^{<} \neq \emptyset$, then we have PoA = PoS = $-\infty$, because any NE has the social welfare of at most $(n-1)(n - \sum_{i \in N} \beta_i) \frac{\sum_{i \in N^{++}} \left(\frac{1}{r_i + r_i'} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r_i'} \right)}$.

If $N^{<} = \emptyset$, **but** $N^{\leq} \neq \emptyset$, then PoA = $-\infty$ but PoS = 1. The reason is that an NE can have the social welfare from zero and down to $(n-1)(n - \sum_{i \in N} \beta_i) \frac{\sum_{i \in N^{\leq}} \left(\frac{1}{r_i + r_i'} \cdot b_i \right)}{\sum_{i \in N} \left(\frac{1}{r_i + r_i'} \right)}$.

If $N^{\leq} = \emptyset$, then PoA = PoS = 1, since an NE has the social welfare of zero.

■

In particular, we have shown that if all the agents find acting easy (i.e., all $\beta_i < 1$), or if all agents really do not like acting (i.e., all $\beta_i > 1$), then PoA = PoS = 1, so that any NE is optimum for the society. Intuitively, this is because here, all the agents have similar preferences: either everyone wants to act and receive action, or no one does. We have also shown, that if the average agent finds not contributing more important than receiving (i.e., $\sum_{i \in N} \beta_i > n$), but still $\beta_i < 1$ for some agent i, then PoA = PoS = $-\infty$, so any NE is catastrophic to the society. Intuitively, this stems from the differences in the agents' preferences. Finally, we see that if $\sum_{i \in N} \beta_i > n$, some agents have $\beta_i = 1$, but none have $\beta_1 < 1$, then PoA = $-\infty$ but PoS = 1, requiring regulation.

Theorem 2 implies that if all the projects have $\beta \leq 1$, then any dividing of all the budget in cooperating is always an NE. This is unintuitive, since usually, some groups are more efficient to interact with than some other groups. The reason for this is that the model assumes that all agents always interact at every project $\omega \in \{1, \ldots, m\}$, and only their kindness depends on the strategy. Basically, everyone attends all the interactions, and some people are passive.

4 Exclusive Thresholded Reciprocation Effort Game

We now define a variation on a shared effort game with reciprocation, where only the agents who contribute at least the threshold may interact. First, following [20], we define a *θ-sharing mechanism*. This models, for example, a minimum invested effort to be considered a coauthor, or a minimum effort to master a technology before working with it. Define, for every $\theta \in [0, 1]$, the players who get a share from project ω to be $N_\omega^\theta \triangleq \left\{ i \in N | x_\omega^i \geq \theta \cdot \max_{j \in N} x_\omega^j \right\}$, which are those who bid at least θ fraction of the maximum contribution to ω.

We now define an *exclusive thresholded reciprocation effort game*, as a reciprocation effort game, where exclusively the agents in N_ω^θ interact. Others do

not obtain utility and do not even interact. If an agent ends up participating alone at a project, he obtains zero utility from that project, since no interaction occurs. Exclusive thresholded reciprocation effort games model situations when joining an interaction requires contributing enough, like the initial effort it takes to learn the required technology to contribute to Wikipedia, the effort to become a member of a file sharing community or to start a firm.

In this section, we assume w.l.o.g. that $b_n \geq \ldots \geq b_1$. The existence of an equilibrium is easy, since no-one contributing constitutes an NE. Then, we show that also less trivial equilibria exist. Finally, the harder question of equilibrium efficiency is answered for two agents. We first notice a trivial equilibrium.

Observation 1. *The profile where all agents contribute nothing is an NE.*

Proof. In this profile, any agent who deviates by contributing a positive amount to a project will be the only one to interact there, so her utility will still be zero. ■

We call an NE where at any project, at most one agent interacts (reaches the threshold) and positively contributes there, a Zero NE. There may be multiple Zero NE. We have just shown that a Zero NE always exists. A natural question is whether there exist non-Zero NE as well. They do.

Theorem 3. *Assume that all agents have $\beta_i \leq 1$. Assume that for any agent i, $r'_i > 0$ and in addition, for any pair of agents i, j we have $r_i + r'_i + r_j + r'_j < 2$. There exists a non-Zero NE.*

Proof. Consider the profile where all agents $1, \ldots, n-1$ contribute their whole respective budgets to project 1, and agent n contributes $\min\left\{b_n, \frac{b_{n-1}}{\theta}\right\}$ to project 1, and nothing to other projects.

This is an NE, for the following reasons. Any agent would be alone at any project other than 1 if it contributed to such a project, and therefore, it will not contribute there. At project 1, the only agent who perhaps can increase her contribution is n, but she will stay alone, if she does, so no deviation is profitable. ■

The next question is the efficiency of the equilibria. Since we always have the Zero NE, and by contributing to the same project the same positive amounts we achieve a positive social welfare, we always have PoA $= 0$. Regarding the price of stability, we immediately know that it is positive, since there always exists a non-Zero NE. We now show that the price of stability for two agents is 1, meaning that there exists a socially optimal NE.

Proposition 2. *For $n = 2$ and under the assumptions of Theorem 3,* PoS $= 1$.

Proof. When we have only two players, we can assume w.l.o.g. that in a profile with maximum social welfare, a project that receives a positive contribution, receives it from both agents. Therefore, social welfare is maximized by maximizing the total contribution to the projects where interaction occurs.

Then, the following profile maximizes the social welfare. Agent 1 spreads her budget equally between all the projects. If $b_1 \geq \theta b_2$, then agent 2 divides her budget equally between all the projects, and otherwise, she contributes $\frac{1}{\theta}\frac{b_1}{m}$ to every project. Since this profile constitutes an NE, we conclude that PoS $= 1$. ■

5 Insufficient Budgets

Till now, we have been assuming that there is enough extended budget to allow the agents make the contributions required by the sum of the reciprocal actions at any time. In this section, we consider dividing effort between reciprocal projects, where the extended budgets B_i may not suffice to reciprocate at some positive time t, and therefore the actions have to be curbed, such that the total action at any time is bounded by the B_i. In Example 1, this can happen if people are unable to keep up with the others because their free time is strictly limited. In order to justify studying the asymptotic behavior here, we prove that for any curbing, the actions in all interactions converge, as time approaches infinity. Then, we study the equilibria of the corresponding game.

The convergence of normal reciprocation is proven in [21], and we now prove the convergence of curbed reciprocation. Consider the undirected interaction graph $G = (N, E)$ of an interaction project, such that agent i can act on j and vice versa if and only if $(i, j) \in E$. Our model assumes that this graph is a clique, meaning that everyone interacts, but this is not necessary for the following theorem. At a given time, let the reciprocation from Definition 1 require actions denoted by the column vector $\boldsymbol{q} \in \mathbb{R}_+^{|E|}$, in the sense that its (i, j)th coordinate contains $\mathrm{act}_{i,j}$ (for $(i, j) \in E$). Then, the curbing is denoted by $D_{\boldsymbol{q}} \cdot \boldsymbol{q}$, where $D_{\boldsymbol{q}}$ is the diagonal curbing matrix. We omit the subscript $\boldsymbol{q}$ when the vector on which we act is clear. We denote the curbing matrix at time t by $D(t)$.

Theorem 4. *Consider dividing effort between reciprocal interactions, where every interaction has some connected interaction graph, and for all agents i, $r'_i > 0$. At every interaction, if there exists a cycle of an odd length in the interaction graph, or at least one agent i has $r_i + r'_i < 1$, then, for all pairs of agents $i \neq j$, the limit $L_{i,j} \triangleq \lim_{t\to\infty} \mathrm{act}_{i,j}(t)$ exists.*

In our model, we assume a completely connected graph, so if at least 3 agents interact, we have an odd cycle, namely a triangle. Therefore, then we only need to assume that for all agents i, $r'_i > 0$.

The proof expresses reciprocation as matrix multiplication. Without curbing, the convergence is proven using the Perron-Frobenius theorem. Keeping convergence when curbing can occur uses the following definition and lemma.

Definition 2. *We remind that a square non-negative matrix A is called* primitive*, if there exists a positive l, such that $A^l > 0$ (see [24, Definition* 1.1*]).*

The following lemma, used to prove the theorem, has a value of its own as well. Given a convergent sequence of primitive matrices, the lemma shows that arbitrarily squeezing the matrices keeps the convergence.

Lemma 2. *Given a vector $\boldsymbol{p}(0) \in \mathbb{R}^d$, a primitive matrix $A \in \mathbb{R}^{d^2}$, such that $\lim_{t\to\infty} A^t$ exists, and a sequence of diagonal matrices $\{D(t)\}_{t=0}^{\infty}$, $D(t) = \mathrm{diag}(\lambda_1(t), \ldots, \lambda_d(t))$, where each $\lambda_i(t) \in (0, 1]$, define the sequence $\{\boldsymbol{p}(t)\}_{t=0}^{\infty}$ by $\boldsymbol{p}(t) \triangleq D(t)AD(t-1)A \ldots D(1)A\boldsymbol{p}(0)$. Then, $\lim_{t\to\infty} \boldsymbol{p}(t)$ exists.*

Proof. Assume to the contrary, that $\{\boldsymbol{p}(t)\}$ diverges. Define the sequence $\{\boldsymbol{p}'(t)\}_{t=0}^{\infty}$ by $\boldsymbol{p}'(t) \triangleq A^t\boldsymbol{p}(0)$. Since $\{\boldsymbol{p}(t)\}$ diverges and $\{\boldsymbol{p}'(t)\}$ converges, they differ at some point, intuitively speaking. We now formalize this argument. Since $\{\boldsymbol{p}(t)\}$ diverges and the space is complete, it is not a Cauchy sequence, and so there exists a positive ϵ, such that for each $N > 0$ there exist $n, m > N$, such that $|\boldsymbol{p}(n) - \boldsymbol{p}(m)| > \epsilon$ ($\|$ is the Euclidean norm). Since $\{\boldsymbol{p}'(t)\}$ converges, it is a Cauchy sequence, so there exists $N > 0$, such that for all $n, m > N$ we have $|\boldsymbol{p}'(n) - \boldsymbol{p}'(m)| < \epsilon/2$. If $|\boldsymbol{p}(n) - \boldsymbol{p}(m)| > \epsilon$ and $|\boldsymbol{p}'(n) - \boldsymbol{p}'(m)| < \epsilon/2$, we cannot both have $|\boldsymbol{p}(n) - \boldsymbol{p}'(n)| < \epsilon/4$ and $|\boldsymbol{p}(m) - \boldsymbol{p}'(m)| < \epsilon/4$. Therefore, for some integer l, $|\boldsymbol{p}(l) - \boldsymbol{p}'(l)| > \delta$, for some $\delta > 0$, depending solely on ϵ. Since the product defining $\boldsymbol{p}(l)$ is like that of $\boldsymbol{p}'(l)$, but with more $D(t)$ matrices, and $D(t) = \operatorname{diag}(\lambda_1(t), \ldots, \lambda_d(t))$, where each $\lambda_i(t) \in (0, 1]$, we have $\mathbf{0} \le \boldsymbol{p}(l) \le \boldsymbol{p}'(l)$. Remembering this, and that matrix A is primitive, thereby propagating a change of an entry to every entry, we can choose l such that every coordinate of $\boldsymbol{p}(l)$ will be at most α fraction of the corresponding coordinate of $\boldsymbol{p}'(l)$, for some $\alpha < 1$. The α can be made to depend solely on ϵ, because of the boundedness of all the relevant vectors. So, we have $\boldsymbol{p}(l) \le \alpha A^l\boldsymbol{p}(0)$.

By reiterating the same argument with $\boldsymbol{p}'_1(t) \triangleq A^t\boldsymbol{p}(l)$ and $\boldsymbol{p}_1(t) \triangleq \boldsymbol{p}(t + l)$, we find $l_1 > 0$, such that $\boldsymbol{p}_1(l_1) \le \alpha A^{l_1}\boldsymbol{p}(l)$. Thus, $\boldsymbol{p}(l_1 + l) = \boldsymbol{p}_1(l_1) \le \alpha A^{l_1}\boldsymbol{p}(l) \le \alpha A^{l_1}\alpha A^l\boldsymbol{p}(0) = \alpha^2 A^{l_1+l}\boldsymbol{p}(0)$.

Continuing in this manner, and using the boundedness of the converging $\{A^t\boldsymbol{p}(0)\}$, we prove that $\{p(t)\}$ converges to zero. A contradiction. [4] ■

We can now prove Theorem 4.

Proof. We extend the proof of the Theorem 1 from [21], which proves the convergence without the curbing. We show that the curbing still keeps the convergence. We recapitulate the used properties from there, to stay self-contained.

We express the dynamics of interaction in a matrix, and prove the theorem by applying the Perron–Frobenius theorem [24, Theorems 1.1 and 1.2], using the above lemma to handle the curbing of actions. Denoting the neighbors of i as $\mathrm{N}(i)$, we define the dynamics matrix $A \in \mathbb{R}_+^{|E|\times|E|}$ as

$$A((i,j),(k,l)) \triangleq \begin{cases} (1 - r_i - r'_i) & k = i, l = j; \\ r_i + r'_i \frac{1}{|\mathrm{N}(i)|} & k = j, l = i; \\ r'_i \frac{1}{|\mathrm{N}(i)|} & k \neq j, l = i; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Assume that for each time $t \in T$, the column vector $\boldsymbol{p}(t) \in \mathbb{R}_+^{|E|}$ describes the actions at time t. Then, $\boldsymbol{p}(t+1) = D(t)A\boldsymbol{p}(t)$, where $D(t)$ is the diagonal matrix, describing the curbing. We call $\boldsymbol{p}(t)$ an action vector. Initially, $\boldsymbol{p}(0)_{(i,j)} = k_i$.

We will use the Perron-Frobenius theorem for primitive matrices. We now prepare to use it, and first we show that A is primitive. In the proof of Theorem 1,

[4] The actual limit does not have to be zero; zero is just the result from the contradictory assumption.

it is shown that A is irreducible and aperiodic, and therefore primitive by [24, Theorem 1.4]. Since the sum of every row is 1, the spectral radius is 1.

According to the Perron-Frobenius theorem for primitive matrices [24, Theorem 1.1], the absolute values of all the eigenvalues except one eigenvalue of 1 are strictly less than 1. The eigenvalue 1 has unique right and left eigenvectors, up to a constant factor. Both these eigenvectors are strictly positive. Therefore, [24, Theorem 1.2] implies that $\lim_{t\to\infty} A^t = \mathbf{1}\boldsymbol{v}'$, where $\boldsymbol{v}'$ is the left eigenvector of the value 1, normalized such that $\boldsymbol{v}'\mathbf{1} = 1$.

Now, Lemma 2 implies that $L_{i,j}$ exists. ■

We have proven the reciprocation effort game where curbing can occur is well defined, because all the reciprocation processes converge. We now prove the existence of equilibria in such a game. In the exclusive thresholded model, Observation 1 holds in the curbed case as well, so contributing nothing is an NE. From now on, assume that no threshold exists. Since the curbing renders finding a formula for the actions in the limit unlikely, we take an abstract approach.

Theorem 5. *Consider dividing effort between reciprocal interactions, where for all agents i, $r'_i > 0$. Assume that $n \geq 3$ or at least one agent i has $r_i + r'_i < 1$. Assume that the curbing function $D\colon \mathbb{R}^n \to \mathbb{R}^n$ is a weak contraction w.r.t. norm L_∞, i.e. $||D_{\boldsymbol{x}}\boldsymbol{x} - D_{\boldsymbol{y}}\boldsymbol{y}||_\infty \leq ||\boldsymbol{x} - \boldsymbol{y}||_\infty$.*

Then, there exist small enough β_is such that an NE exists.

Proof. By Theorem 4, the reciprocation processes converge and so the game is well defined. We prove the existence using Proposition 20.3 from [18]. The strategy set of every player consists of all the possible divisions of the budgets between the projects, which is a nonempty compact convex set.

The continuity of the utility functions follows from the action limits depending continuously on the total contributions of agents to projects. To this end, we can inductively show that at ant time t, the change in the action is $||\boldsymbol{p}'(t) - \boldsymbol{p}(t)||_\infty \leq ||\boldsymbol{\Delta}x||_\infty$, where $\boldsymbol{p}'(t)$ represents the actions at time t if $\boldsymbol{p}'(0) = \boldsymbol{p}(0) + \boldsymbol{\Delta}x$. This boundedness keeps holding in the limit of time approaching infinity as well, implying continuity. For the quasi-concavity of an agent's strategy space, notice that for small enough β_i, agent i would like to increase its contribution exactly till it can increase the limit of the actions of at least one another agent. Finally, Proposition 20.3 implies the theorem. ■

We also prove that when agents react identically, the game boils down to a single reciprocal interaction.

Proposition 3. *Assume that the curbing is determined by the sum of the actions of an agent and that all the reciprocation coefficients are equal among the agents, i.e. $r_i = r_j$ and $r'_i = r'_j, \forall i \neq j$. Then, the total contribution of agent i at any time $t \in T$, i.e. $x^i(t) \triangleq \sum_{\omega\in\Omega} x^i_\omega(t)$, and the total received action, i.e. $\sum_{\omega\in\Omega} (\mathrm{got}_i(t))_\omega$, are fully determined by the total contributions and the total received actions at time zero of the agents (i and others), regardless how the actions were divided between the projects.*

Proof. We prove this by induction on time. At the *basis*, $t = 0$ and the statement is trivial. At the *induction step*, assume that $\sum_{\omega\in\Omega} x^i_\omega(t-1)$ and $\sum_{\omega\in\Omega} (\mathrm{got}_i(t-1))_\omega$ are fully determined by the total contributions and the total received actions at time zero and prove this determinancy for $\sum_{\omega\in\Omega} x^i_\omega(t)$ and $\sum_{\omega\in\Omega} (\mathrm{got}_i(t))_\omega$. Indeed, $x^i_\omega(t)$ is equal to

$$\begin{aligned}\sum_{j\neq i} \underset{i,j}{\mathrm{act}} &= (1-r_i-r'_i)\sum_{j\neq i} (\underset{i,j}{\mathrm{act}}(t-1))_\omega + r_i \sum_{j\neq i} (\underset{j,i}{\mathrm{act}}(t-1))_\omega + r'_i(\underset{i}{\mathrm{got}}(t-1))_\omega \\ &= (1-r_i-r'_i)x^i_\omega(t-1) + r_i(\underset{i}{\mathrm{got}}(t-1))_\omega + r'_i(\underset{i}{\mathrm{got}}(t-1))_\omega.\end{aligned}$$

Sum it up over all the projects to obtain

$$\sum_{\omega\in\Omega} x^i_\omega(t) = (1-r_i-r'_i)\sum_{\omega\in\Omega} x^i_\omega(t-1) + r_i \sum_{\omega\in\Omega} (\underset{i}{\mathrm{got}}(t-1))_\omega + r'_i \sum_{\omega\in\Omega} (\underset{i}{\mathrm{got}}(t-1))_\omega.$$

Since everything on the right hand side is, by the induction hypothesis, determined by the total contribution and the total received action at time zero, the actions on time t before curbing are determined by them as well. Furthermore, curbing is determined by the total action of the agents, and thus, the curbed actions are also determined by the total contribution and the total received action at time zero.

Regarding the total received action, the derivation of the step is analogous, but it requires moving $1-r_j-r'_j, r_j$ and r'_j out of the parentheses, where we use the equality of these parameters across the agents. ■

6 Conclusions and Further Research

In order to predict investing effort in several reciprocal interactions, we define a game that models dividing efforts between several reciprocal projects. We include an analysis of a model both with and without a contribution threshold.

When no contribution threshold exists, there always exists an equilibrium, and if acting is easy to everyone (for all i, $\beta_i < 1$) or hard to everyone (for all i, $\beta_i > 1$), then every NE is socially optimal. We also show that any dividing of all the budget when acting is easy to everyone is a Nash equilibrium. The result may seem surprising. Intuitively, this happens because everyone participates in each interaction, and the concrete division of the budget does not matter to the social welfare. However, life does not often provide such situations. We also characterize when both efficient and inefficient equilibria exist, calling for regulation.

If a minimum contribution is necessary to participate in interaction, we show that the situation where no-one contributes is an equilibrium. This models the case where people are very passive, and this continues since no-one can start an interaction project on his own. In addition to this trivial equilibrium, we find an equilibrium where all the agents contribute to the same project, like Facebook, instead of participating in the other social networks. This describes the case

when people interact with each other on the same topic. Such a situation is clearly not the only option, since people often have many friendships [9]. For two agents, there exists an equilibrium which is socially optimal.

The choices of strategies by the agents who are indifferent can significantly influence the social welfare. For instance, this happens in the case without threshold to agents for whom acting and receiving action are equally important. Making such agents do what benefits the society can increase the social welfare.

We also model the case when the extended budgets are not big enough and curbing is required. We show that any way of dividing effort between reciprocal interactions results in converging interactions, regardless of how actions are curbed to fit the budgets. We also prove that the resulting reciprocation effort game possesses an equilibrium, with and without threshold.

For future research, we are curious about the efficiency of the equilibria in the game with curbing. Consecutive decisions can be modeled by the agents first contributing to the interactions and then deciding on their reciprocation parameters. Additionally, looking at interactions in large groups where not everyone can act on everyone else would be a natural generalization of our work. Another point is that we assumed that two agents who interact in multiple projects, interact in these projects independently. Modeling the dependency between these interactions is realistic. Analyzing a mixed set of projects, only some of which are interaction projects, would model reality better. Also modeling and analyzing voting to approve who else may participate in an interaction seems promising.

This work models and analyzes a ubiquitous class of interactions and lays the basis for further research, aimed to provide more advice to the agents and to the manager who wants to maximize the social welfare.

Acknowledgments. We thank Prof. Orr M. Shalit from the Technion, Israel for the useful discussions. This work has been supported by SHINE, the flagship project of DIRECT (Delft Institute for Research on ICT at the TU Delft).

References

1. Anshelevich, E., DasGupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 295–304, October 2004
2. Anshelevich, E., Hoefer, M.: Contribution games in social networks. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6346, pp. 158–169. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15775-2_14
3. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
4. Bachrach, Y., Syrgkanis, V., Vojnović, M.: Efficiency and the Redistribution of Welfare. Technical report, May, Microsoft Reserach (2012)
5. Baye, M.R., Kovenock, D., de Vries, C.G.: The all-pay auction with complete information. Econ. Theor. **8**(2), 291–305 (1996)
6. Bicchieri, C.: The Grammar of Society: The Nature and Dynamics of Social Norms. Cambridge University Press, Cambridge (2006)
7. Dixon, W.J.: Reciprocity in United States-Soviet relations: multiple symmetry or issue linkage? Am. J. Polit. Sci. **30**(2), 421–445 (1986)

8. Dufwenberg, M., Kirchsteiger, G.: A theory of sequential reciprocity. Games Econ. Behav. **47**(2), 268–298 (2004)
9. Ennett, S.T., Bauman, K.E.: Adolescent social networks: friendship cliques, social isolates, and drug use risk, pp. 83–92. Tanglewood Research Inc., Greensboro, North Carolina (2000)
10. Falk, A., Fischbacher, U.: A theory of reciprocity. Games Econ. Behav. **54**(2), 293–315 (2006)
11. Fehr, E., Fischbacher, U., Gächter, S.: Strong reciprocity, human cooperation, and the enforcement of social norms. Hum. Nat. **13**(1), 1–25 (2002)
12. Fehr, E., Gächter, S.: Fairness and retaliation: the economics of reciprocity. J. Econ. Perspect. **14**(3), 159–181 (2000)
13. Gottman, J., Swanson, C., Murray, J.: The mathematics of marital conflict: dynamic mathematical nonlinear modeling of newlywed marital interaction. J. Fam. Psychol. **13**, 3–19 (1999)
14. Kleinberg, J., Oren, S.: Mechanisms for (mis)allocating scientific credit. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 529–538. ACM, New York (2011)
15. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, pp. 404–413, 4–6 March 1999
16. Nie, N.H., Hillygus, D.S.: Where does internet time come from? A reconnaissance. IT Soc. **1**, 1–20 (2002)
17. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
18. Osborne, M.J., Rubinstein, A.: A Course in Game Theory, vol. 1. MIT Press, Cambridge (1994)
19. Polevoy, G., de Weerdt, M., Jonker, C.: The game of reciprocation habits. In: Proceedings of the 2016 European Conference on Artificial Intelligence (ECAI 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 417–425 (2016)
20. Polevoy, G., Trajanovski, S., de Weerdt, M.M.: Nash equilibria in shared effort games. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2014), pp. 861–868. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2014)
21. Polevoy, G., de Weerdt, M., Jonker, C.: The convergence of reciprocation. In: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), pp. 1431–1432. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2016)
22. Rabin, M.: Incorporating fairness into game theory and economics. Am. Econ. Rev. **83**(5), 1281–1302 (1993)
23. Schulz, A.S., Moses, N.S.: On the performance of user equilibria in traffic networks. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003), pp. 86–87. Society for Industrial and Applied Mathematics, Philadelphia (2003)
24. Seneta, E.: Non-negative Matrices and Markov Chains. Springer Series in Statistics. Springer, New York (2006). https://doi.org/10.1007/0-387-32792-4
25. Siegel, R.: All-pay contests. Econometrica **77**(1), 71–92 (2009)
26. Trivers, R.L.: The evolution of reciprocal altruism. Q. Rev. Biol. **46**, 35–57 (1971)
27. Van Segbroeck, S., Pacheco, J.M., Lenaerts, T., Santos, F.C.: Emergence of fairness in repeated group interactions. Phys. Rev. Lett. **108**, 158104 (2012)
28. Ward, M.D.: Modeling the USA-USSR arms race. Trans. Soc. Model. Simul. Int. **43**, 196–203 (1984)

Get Your Virtual Hands Off Me! – Developing Threatening IVAs Using Haptic Feedback

Linford Goedschalk, Tibor Bosse(✉), and Marco Otte

Vrije Universiteit Amsterdam, De Boelelaan 1081, 1081 HV, Amsterdam, The Netherlands
t.bosse@vu.nl

Abstract. Intelligent Virtual Agents (IVAs) become widely used for numerous applications, varying from healthcare decision support to communication training. In several of such applications, it is useful if IVAs have the ability to take a negative stance towards the user, for instance for anti-bullying or conflict management training. However, the believability of such 'virtual bad guys' is often limited, since they are non-consequential, i.e., are unable to apply serious sanctions to users. To improve this situation, this research explores the potential of endowing IVAs with the ability to provide haptic feedback. This was realized by conducting an experiment in which users interact with a virtual agent that is able to physically 'touch' the user via a haptic gaming vest. The effect on the loudness of the speech and the subjective experience of the participants was measured. Results of the experiment suggest there might be an effect on the subjective experience of the participants and the loudness of their speech. Statistical analysis, however, shows no significant effect but due to the relatively small sample size it is advisable to further look into these aspects.

1 Introduction

Intelligent Virtual Agents (IVAs) are intelligent digital interactive characters that can communicate with humans and other agents using natural human modalities like facial expressions, speech, gestures and movement [19]. Recently, IVAs have become widely used for numerous applications, varying from healthcare decision support [8] to communication training [20]. In such applications, IVAs play various roles in which they interact with users, for instance as an instructor, therapist or teammate [15].

In the vast majority of these cases, IVAs are friendly and supportive towards the user. Instead, there was less attention for IVAs with a 'negative' or 'aggressive' attitude towards users (i.e., 'virtual bad guys'). This could be considered a missed opportunity, since the concept of virtual bad guys opens up a range of useful applications. Examples include virtual training of aggression de-escalation skills [6], anti-bullying education [23], and Virtual Reality exposure therapy [16].

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 61–75, 2018.
https://doi.org/10.1007/978-3-319-76892-2_5

However, a common difficulty in the design of IVAs is to make them *believable*, i.e., to provide the illusion of being alive [2]. This poses a particular challenge for 'virtual bad guys', since effective applications involving aggressive agents require that users feel indeed seriously threatened or stressed by the IVA. However, IVAs are typically non-consequential, i.e., they are unable to apply serious sanctions to (or even physically harm) their human interlocutors. As a result, users still perceive IVAs as rather artificial beings, which possibly also influences the way they interact with them.

Triggered by this insight, the question addressed in this paper is how to develop aggressive virtual agents that are taken seriously. This question is tackled by designing and experimentally examining the effects of a threatening IVA that is able to physically 'touch' users by means of haptic feedback, which is realized by means of a haptic gaming vest. More specifically, we investigate whether endowing an IVA with the ability to provide such a physical threat has an impact on the verbal behaviour of users as well as their subjective experience.

The remainder of this article is structured as follows. In Sect. 2, the recent literature on aggressive virtual agents and on haptic feedback is reviewed. Next, in Sect. 3 the design of the performed experiment is presented, and the results are provided in Sect. 4. These results are evaluated in detail in Sect. 5. A conclusion is provided in Sect. 6, and Sect. 7 completes the paper with a discussion.

2 Related Work

The relevant literature for this project covers two main areas, namely aggressive virtual agents and virtual touch. The state-of-the art in these two areas is discussed in the following sub-sections.

2.1 Aggressive Virtual Agents

Research on emotions within IVAs has received much attention in recent years. An important stream of research addresses the development of generic computational models of emotion [12]. Probably the most influential approach is EMA [13], a computational model that formalises the main assumptions behind appraisal theory [11]. Although such models could be used to have agents generate emotional states like 'angry', they do not focus on agents that take a threatening attitude towards humans.

Instead, other research has focused more explicitly on the impact of emotional agents on humans in interpersonal settings. For example, the Sensitive Artificial Listener paradigm enables studying the effect of agents with different personalities on human interlocutors, which provided evidence that IVAs with an angry attitude indeed trigger different (subjective and behavioural) responses than agents with other personalities [17]. Similarly, a study in the domain of negotiation led to the conclusion that IVAs expressing anger (in terms of utterances and facial expressions) lead human negotiation partners to make larger concessions [7]. Another recent study pointed out that a virtual agent that made an

'outburst of aggression' (in terms of shouting to and insulting the user) was able to trigger increased physiological responses [5].

Nevertheless, as also concluded in [5], these responses are still insufficiently strong to be really useful for effective applications where heavy emotional stimuli play a role, such as aggression de-escalation training systems for law enforcement personnel [9] or public transport employees [6]. The assumption underlying the current paper is that this is due to the inability of existing IVAs to apply serious (e.g., physical) sanctions to human interlocutors. This is in line with research in the domain of shooting behaviour training for police officers, which indicates that 'simulated threat' is a necessary criterion to realize an adequate transfer of training from the simulated to the real world [14].

Hence, the current paper aims to bring interactions between humans and aggressive virtual agents to a next level of realism, which is done by introducing two technological innovations, namely immersive Virtual Reality and haptic feedback.

2.2 Virtual Touch

The domain of virtual reality is seeing its technology applied in fields like entertainment, education and even medicine. One of these applications is virtual reality-based training, which can be used for various purposes. In order for the effects of such training applications to be applied in the physical world it is important that the scenarios used resemble this world as closely as possible. This way users are offered an experience as if they were in the physical world itself.

Since a number of years, virtual reality applications are combined with haptic feedback, enabling users to 'touch' objects in the simulated environment. Most of these applications focus on touching static objects rather than conversational agents, for instance for surgical training (e.g., to improve performance in cadaver temporal bone dissection) [22].

Recently, virtual touch is also applied in a more social setting, leading to the area of 'virtual interpersonal touch' (e.g., [1]). For instance, research by Cheok and colleagues explores the use of haptic technology to reproduce multisensory sensations related to intimate activities like kissing [21] and hugging [18]. Although the primary use of such technology was to enable intimate touch sensations between humans remotely, it is also claimed to have potential in the area of human-agent interaction. Similarly, other researchers have studied the use of social touch with the aim to make virtual agents more 'warm' or empathic (e.g., [4,10]). Nevertheless, all of these developments are in the context of 'positive interpersonal touch'. As far as could be determined, research into virtual agents that may touch human conversation partners with the purpose of intimidating or threatening them is still in its infancy.

3 Method

To investigate the effect of touch by a threatening virtual agent on human participants, the following experiment was set up. Participants were asked to interact with a virtual agent in a virtual reality environment through free speech. At some point during this interaction, the virtual agent would start threatening the user, which was followed by a 'push' that was simulated through haptic feedback. A haptic gaming vest was chosen, because this enabled the users to be provided with a serious physical stimulus, while avoiding physically harming them (which would be the case by using for instance electric surges), which obviously is ethically irresponsible. This section describes the experimental set-up in detail.

3.1 Participants

A convenience sample of 47 people was recruited, most of which were academic students. The age of the participants varied between 18 and 25 years. Participants were randomly assigned to the experimental group that received haptic feedback or the control group that did not receive any feedback. The experimental group consisted of 21 participants (12 male, 9 female), and the control group consisted of 23 participants (13 male, 10 female). Three runs of the experiment resulted in corrupted or incomplete data and so these have been removed.

3.2 Experimental Design

Participants were placed in either a condition with haptic feedback during the interaction (condition A) or a condition without haptic feedback (condition B). In both conditions, participants were wearing the haptic feedback equipment (as to eliminate any effects of the equipment itself), but they were not told in advance what was the purpose of the equipment. In the control condition, the haptic vest was turned off, but the participants did not know this.

The experiment used a between-participants design (where each participant is only allocated to one condition) instead of a within-participants design (where each participant would experience both conditions sequentially), because in the latter case, the participants would already expect the virtual push after having experienced the scenario once.

3.3 Tasks

The participants were asked to engage in a virtual reality scenario (displayed on a Head Mounted Display) taking place in the context of a nightclub in which they can freely move around. The participants were tasked with finding the bathroom in the virtual environment, after which they were to return to the bar area they initially started in and have a drink with their friend. However, on their way to the bathroom the participants would encounter a virtual agent named Mason.

Mason poses the threat in the virtual environment by acting very aggressively towards the participants. As they walk through the corridor towards Mason, he walks into them and so spills his drink. This sets up a situation in which the agent behaves aggressively towards the users and at some point even physically 'attacks' the user. This attack has the form of a push which is transferred to the user through the haptic feedback vest.

Throughout the nightclub several virtual agents can be seen and heard interacting with each other. The users can only interact with two of the agents in the scenario through free speech responses during a conversation. The first agent they encounter is used to make the participants more comfortable with the free speech interaction paradigm. The second agent, Mason, is used to analyse the responses of the participant. Both agents have been created in such a way that they always give the same responses, no matter what the user says. This was done to minimize the differences across individual trials.

The responses of virtual agent Mason have been inspired by the Sensitive Artificial Listener paradigm [17], which enables users to interact with virtual agents using free speech. The dialogue can be set-up in such a way that the agent always seems to respond to what the user says even if this is not the case and the agent just follows a script.

During the conversation with Mason the participants were free to respond in whatever way they saw fit. Participants were only limited in that they should speak loud and clear, always respond to the agent and use at least one full sentence to respond. The conversation consists of ten user responses which results in a select set of data to analyse. To this end, participants were asked to respond at all times. The participants were alerted to the fact that the microphone might not pick up their voice if they did not speak loud and clear and this would result in difficulties for the analysis of the data.

Both conversations in the scenario are turn-based, meaning that both the agent and the participant take turns while speaking. Only the agents can initiate conversations as to avoid the participants trying to start a conversation with every agent they encounter.

3.4 Variables

Two types of dependent variables were used in this study, namely subjective and objective variables. As subjective variable, the participants' experience was measured through a questionnaire they had to fill in at the end of the experiment. This questionnaire contained the following questions, which had to be answered using a 5-point Likert scale (from 'not at all' to 'very much'):

Q1 Did you have any experience with the use of head mounted display devices prior to this experiment?
Q2 Did you have any experience with the use of haptic feedback hardware prior to this experiment?
Q3 Did you find the virtual scenario to be realistic?
Q4 Did you find Mason to be aggressive?
Q5 Did you find Mason to be threatening?

In addition, the following yes-no questions were used:

Q6 Were you startled when Mason pushed you?
Q7 Did you look at Mason when he pushed you?
Q8 Did you first walk through the club before talking to Mason?
Q9 Did you react differently to Mason after he pushed you?

Some of these questions (Q1, Q2, Q7, Q8) were used as control questions, to avoid that any differences found could be attributed to other factors. For the other questions, the aim was to investigate *whether people have a more intense experience in the condition with haptic feedback than in the condition without* (Research Question 1).

The objective variable that was studied was the verbal behaviour of the participants during the interaction with Mason. More specifically, we used the loudness of their speech as an indicator for the participants' engagement in the scenario, as people who are excited typically speak louder [3]. The relevant data for this were obtained through the use of a web cam that recorded the experiment session. The audio from the recordings was extracted and the amplitudes from the audio files were sampled. This way it could be analysed whether the participants spoke louder or softer after being pushed in the virtual scenario. Hence, the aim was to investigate *if people use louder speech in the condition with haptic feedback than in the condition without* (Research Question 2).

3.5 Material and Facilities

The experiment has been conducted in a quiet room in which only the participant and experimenter were present. This room contained a desk with the computer that hosted the virtual environment, a four-legged chair for the participants to sit on during the experiment and a desk with the equipment used during the experiment. The chair on which the participants took place was selected not to be an office chair, as these chairs can turn. When the participant is using the Head Mounted Display to look around in the virtual environment sitting on an office chair would mean they would be able to look behind themselves in the environment while their virtual body would still be facing the other way.

The Virtual Environment was presented to the user using a Head Mounted Display, in this case the Oculus Rift Developer Kit (version 2)[1]. Using an advanced high-quality Virtual Environment and a Head Mounted Display requires a high-end gaming computer with a high-end graphics card to ensure smooth performance for an optimally effective Virtual Environment. The computer used an Intel i7-4630 CPU with 16 GB DDR4 memory, a 500 GB SSD and a Nvidia GTX-780 graphics card with 1 GB of memory. To facilitate the haptic feedback a so-called gaming vest was used (the KOR-FX[2]). These vests incorporate vibration motors that mimic physical impact to the torso. The KOR-FX vest uses two large vibration motors, one on left side of the chest and one on the

[1] https://www3.oculus.com/en-us/dk2/.
[2] http://korfx.com/.

right side. The vest is wirelessly connected to a control box. This control box accepts low-voltage input (0–5 V) and is meant to accept standard sound output of the sound card of a computer. To gain complete control over the haptic feedback, an Arduino One board[3] has been used with an analogue line (0–5 V) as output to the KOR-FX controller box. The Arduino accepted commands from the Virtual Environment, via the USB connection to the computer, to activate the vibration motors in the gaming vest. This way the Virtual Environment had complete control over the haptic feedback to the participants. The Arduino also used a microphone that recorded the volume level of the sound in the environment, i.e. the voice of the participant. The microphone polled from a script inside the Virtual Environment to monitor the speech of the participant. A standard USB game controller was used to control the virtual agent of the user. See Fig. 1 for an overview of the system's architecture.

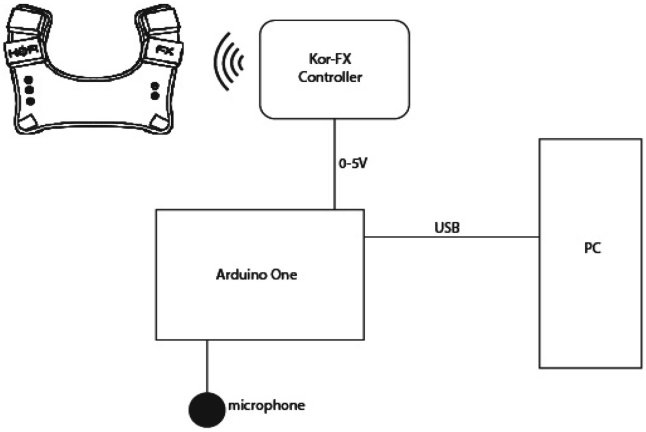


Fig. 1. System architecture.

3.6 Virtual Environment

The Virtual Environment has been developed in Unity Pro (version 5)[4]. A ready-made model from the Unity Asset Store has been purchased for the club environment that has been used in the experiment. This model has been further adapted in order to suit the needs of this research. Atmosphere was added by including special lighting and additional props on the virtual stage. All the humanoid agents in the Virtual Environment have been generated using the iClone Pipeline software (version 6)[5]. The Character Creator[6] has been used to generate realistic and unique human agents. iClone itself has been used to create the body animations and lip-sync movements.

[3] https://www.arduino.cc/en/Main/ArduinoBoardUno/.
[4] https://unity3d.com/.
[5] http://www.reallusion.com/iclone/default.html.
[6] http://www.reallusion.com/iclone/character-creator/default.html.

3DXchange[7] has been used to convert the agents including their animations into FBX format that could be imported into Unity Pro. Inside Unity Pro the non-interactive characters were scripted using C#, looping animations and speech to create a livelier atmosphere in the club. The interactive agents, the character in the role of the friend of the participant and Mason, have been separately scripted for more advanced actions. These two agents had a larger set of animations and speech, plus the ability to react to speech of the participants. The agent would monitor if the participant was speaking. If the participants did speak the agent would wait until the participant stopped, allowing for small pauses in speech (of 1 s), or until a maximum amount of time (of 10 s) had elapsed. This produced a more realistic reaction of the agent.

Additional scripts made sure that once the participant entered the hallway to the toilets, an encounter with Mason was unavoidable. Both speed and direction of movement of the avatar of the participant were taken over by the script so that the participant and Mason would end up directly in front of each other. A screenshot of the application is shown in Fig. 2.

Fig. 2. Screenshot of the application.

3.7 Procedure

After entering the room the participants were asked to sign an informed consent form, allowing for the gathered data to be saved and used for the duration of the research project. Participants also read the health and safety warnings for the Oculus Rift and KOR-FX gaming vest to be able to indicate whether they could safely work with this equipment.

Next, the participants read the experiment instructions and put on the KOR-FX gaming vest. They would take their seat behind the computer and the experimenter would inform them of the instructions once more, highlighting the importance of speaking loud and clear, always responding and using at least one whole

[7] http://www.reallusion.com/iclone/3DXchange.html.

sentence to respond with. If the participants had no further questions they put on the Oculus Rift and the experimenter started a tutorial scenario. In this scenario participants could walk around in order to get accustomed to the controls. The scenario is a grey plain with several blocks placed on it for orientation purposes. The 'ceiling' of the scenario is a sky with a sun. When the participant indicated to understand the controls the experimenter started the recording and the virtual scenario in which all interactions took place.

The participants interacted with the first agent in the environment and then moved on to find the bathroom as instructed. After completing their conversation with virtual agent Mason the screen faded to black and the experimenter stopped the recording. All equipment used was removed and the participant was asked to fill in the questionnaire on the computer. During this the experimenter did not answer any questions the participants had, nor respond to any of their remarks regarding the experiment as not to influence their answers to the questionnaire. For their participation in the experiment, participants were rewarded with a sweet roll after completion of all tasks.

4 Results

This section describes the results of the experiment in detail. First, the subjective measures will be presented, followed by the objective measures.

4.1 Subjective Measures

Figure 3 shows the means of the answers given by the participants to the Likert-scale questions. For example, in condition A (the condition in which haptic feedback was received), the mean of the answers to the question regarding experience with head mounted display devices was 2.33, whereas for condition B it was 2.00.

To analyse whether there was a significant difference between the two conditions regarding the mean answers that were provided to the Likert-scale questions, unpaired t-tests have been performed, under the assumption that the scales reflect continuous data. The results of these tests are displayed in Table 1.

Table 1. T-test results on Likert questions (significance level = 0.05).

Question	P-value	Significant difference
Q1 (HMD experience)	0.37	No
Q2 (haptics experience)	0.62	No
Q3 (scenario realistic)	0.12	No
Q4 (agent aggressive)	0.76	No
Q5 (agent threatening)	0.73	No

Figure 4 displays the results for the yes-no questions for condition A and B, respectively. As an illustration, The figure shows that in condition A, 13

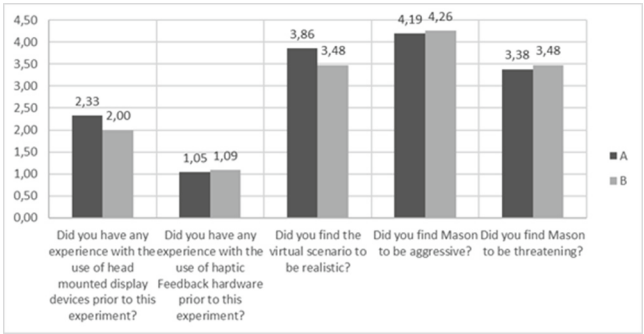


Fig. 3. Answers to the Likert-scale questions.

participants gave a positive answer to Question 6 ('Were you startled when Mason pushed you?'), whereas 8 participants gave a negative answer. Instead, in condition B, 8 participants gave a positive answer to this question, and 15 participants gave a negative answer.

To analyse whether there was a significant difference between the two conditions regarding the answers that were provided, a series of Chi-square tests have been performed. The results of these Chi-square tests are displayed in Table 2.

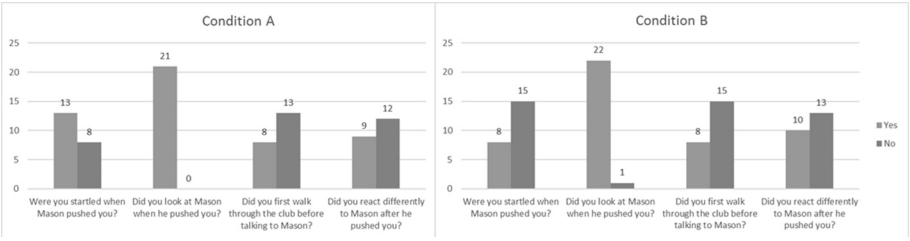


Fig. 4. Answers to yes-no questions for condition A (with haptic feedback) and B (without haptic feedback).

Table 2. Chi-square test results on yes-no questions (significance level = 0.05).

Question	Chi-square value	Variable independence
Q6 (startled by push)	0.07	Yes
Q7 (looked at agent)	0.79	Yes
Q8 (walked around)	0.99	Yes
Q9 (different after push)	0.99	Yes

4.2 Objective Measures

Objective data were obtained by analysis of the audio recording of the experiment sessions. From the audio files, obtained during the experiment sessions using a web cam, the amplitudes of the speech sample concerned with the conversation with Mason have been extracted using Audacity[8], an audio editing tool. These data were then processed using a script written in Python 2.7 in order to obtain the mean amplitude of the utterances prior to the haptic feedback event (i.e., the virtual push) and of the utterances after the event. This was done to determine whether participants spoke louder or softer after being pushed. A summary of the results is presented in Table 3.

Table 3. Comparison of speech volume before and after the virtual push.

Condition	Louder	Softer
A (haptic feedback)	15	6
B (no haptic feedback)	10	13

Statistical analysis of these data was performed again using Chi-square tests. The Chi-square value of this test was 0.06.

5 Evaluation

In this section the results that have been presented in the previous section will be evaluated in the context of the original research questions. First, the subjective results will be evaluated (Research Question 1) and thereafter the objective results (Research Question 2). Finally, a brief follow-up experiment is described, along with its results.

5.1 Research Question 1

Table 1 shows there is no significant difference between the ratings for prior experience between the experimental and control group (Q1 and Q2). The fact that there is no significant difference between the two groups in the level of experience with any of the devices used, indicates that any effect that is found can not be contributed to this.

There was also no significant difference found between how aggressive (Q4) or threatening (Q5) the participants perceived virtual agent Mason to be. Neither was there any difference between how realistic the experience was for the participants (Q3). This would indicate that the haptic feedback that was provided to participants in condition A did not affect any of these factors. Participants in the control group have indicated to have had almost the same experience as those participants that did receive haptic feedback.

[8] http://www.audacityteam.org/.

The Chi-square tests applied on the yes-no questions, displayed in Table 2, show that there is variable independence between the experimental group to which the participants were allocated (A or B) and their answers to these questions. This means that the group in which the participants were placed did not affect their answer to these questions. Therefore, on the one hand, any effect of the haptic feedback cannot be attributed to some of the participants looking at Mason and others looking away (Q7) or some participants walking around in the environment first and others heading straight for their goal (Q8). On the other hand, this also means that no effect of haptic feedback on the reaction of the participants (Q9) or them being startled by the virtual push (Q6) is found.

However, the statistical test performed to determine variable independence between participants being startled and receiving haptic feedback returned a P-value of 0.07. In addition, several people in the haptic feedback condition mentioned that they found the feedback experience at least 'surprising'. As this was an experiment with a relatively low number of participants (44) and the significance value used for this test was 0.05, it is advisable to perform a second experiment with a larger sample size in order to determine whether there is actually no correlation between participants receiving haptic feedback and being startled by the virtual agent.

5.2 Research Question 2

The data obtained after processing of the audio files seem to suggest that participants that received haptic feedback on average spoke louder after receiving this feedback compared to participants in the control group. However, the Chi-square test shows variable independence, indicating that the pattern could be obtained through chance. Just as with the subjective data regarding the startling of participants it is important to remark that the Chi-square value is 0.06. Therefore, it would be advisable to perform a second experiment with a larger sample size in order to determine whether there actually is no pattern between loudness of speech and the application of haptic feedback.

5.3 Follow-Up Experiment

In order to investigate the effect of haptic feedback on the experience of being startled (Q6) with a (slightly) larger sample, ten additional participants performed the experiment at a later date. Due to problems with the equipment, this follow-up experiment could not be conducted for the objective measures. Since a second analysis of the data would be performed (including the data from the first 44 participants) the significance level was adjusted to 0.025, under the assumption that adding more participants would otherwise always lead to some kind of significant effect.

Analysing the subjective data regarding the startling effect for all 54 participants yielded the results presented in Table 4. After increasing the number of participants, there still seems to be no significant effect of startling. On the contrary: the Chi-square value has increased from 0.07 to 0.1.

Table 4. Chi-square test results on question Q6 (54 participants, significance level = 0.025).

Condition	*Yes*	*No*
A (haptic feedback)	15	12
B (no haptic feedback)	9	18
Question	*Chi-square value*	*Variable independence*
Q6 (startled by push)	0.1	Yes

6 Conclusion

In this research the effects of negative haptic feedback in the form of a 'push' by a virtual agent in a threatening scenario are explored. To this end an experiment was set-up featuring 44 participants, distributed over two conditions. During this experiment participants interacted with two virtual agents through free speech in a virtual environment. For this an Oculus Rift and the KOR-FX gaming vest have been used. Participants in the experimental group received haptic feedback, through vibrations created by the KOR-FX vest, at a certain point during their conversation with one of the virtual agents. At this moment in the conversation the participants were being attacked by the virtual agent in the form of a push that was synchronised with the haptic feedback that was received. The participants in the control group also used both the Oculus Rift and the KOR-FX vest, but did not receive haptic feedback during their interactions with the virtual agent.

Subjective data were obtained from a questionnaire that was filled in by participants after the experiment had been completed. Objective data were obtained through a recording of the experiment session using a web cam. The audio recording of the experiment was analysed in order to determine the loudness of speech of the participants prior to- and after the haptic feedback event.

Statistical tests indicate that haptic feedback did not have any effect on the experience of the participants in this scenario for the measured variables. The Chi-square test that was performed on the loudness of speech resulted in a value of 0.06, which was close to the significance level of 0.05. In the questionnaire participants were asked whether they were startled when they were pushed in the scenario. The statistical test performed on these answers resulted in a Chi-square value of 0.07 for a significance level of 0.05. As this study featured 44 participants, ten additional participants performed the experiment in order to gain better insight in the near-significant effects. A second statistical analysis of the subjective data for the startling effect resulted in a value of 0.1. As a consequence, no significant effects of the haptic feedback could be demonstrated on the various subjective aspects (Research Question 1) and objective aspects (Research Question 2) measured.

7 Discussion

Despite the fact that no statistically significant effects of haptic feedback were found, this research provides several useful pointers for follow-up research. First of all, the fact that no effect on subjective experience could be demonstrated might be related to the particular set-up of the experiment, which used a relatively low number of participants, and a between-participants design. Since participants played the scenario only once, they had no frame of reference to which they could compare their experience, which made their Likert-scale response difficult to interpret. It might be the case that if participants would experience both conditions (with and without haptic feedback), they would still feel a big difference between them. This is a common problem in user experience research, and it is worthwhile to explore this in more detail.

Secondly, although this research has explored the effect of negative haptic feedback on several aspects of the experience of the user, it has not been exhaustive in that regard. Future research might look into the effect on loudness of speech using more participants in order to ascertain whether haptic feedback might have an effect or not. Other considerations for future research might include alternative ways in which negative haptic feedback influences user experience, the role of the intensity of the feedback (possibly up to the point where the feedback actually hurts), and providing haptic feedback multiple times.

All in all, it is concluded that the lack of significant effects found in the present study should rather be explained by the specific design of this experiment than by the paradigm as a whole, and that follow-up research is required to investigate the full potential of threatening virtual agents based on haptic feedback.

References

1. Bailenson, J.N., Yee, N.: Virtual interpersonal touch: haptic interaction and copresence in collaborative virtual environments. Multimedia Tools Appl. **37**(1), 5–14 (2008)
2. Bates, J.: The role of emotions in believable agents. Commun. ACM **37**(7), 122–125 (1994)
3. Bell, R.T.: Sociolinguistics. BT Batsford, New York (1976)
4. Bickmore, T.W., Fernando, R., Ring, L., Schulman, D.: Empathic touch by relational agents. IEEE Trans. Affect. Comput. **1**(1), 60–71 (2010)
5. Blankendaal, R., Bosse, T., Gerritsen, C., de Jong, T., de Man, J.: Are aggressive agents as scary as aggressive humans? In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4–8, 2015, pp. 553–561 (2015)
6. Bosse, T., Gerritsen, C., de Man, J.: An intelligent system for aggression deescalation training. In: ECAI 2016–22nd European Conference on Artificial Intelligence, 29 August–2 September 2016, The Hague, The Netherlands – Including Prestigious Applications of AI (PAIS 2016), pp. 1805–1811 (2016)
7. de Melo, C.M., Carnevale, P.J., Gratch, J.: The effect of expression of anger and happiness in computer agents on negotiations with humans. In: 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, 2–6 May 2011, vol. 1–3, pp. 937–944 (2011)

8. DeVault, D.: SimSensei kiosk: a virtual human interviewer for healthcare decision support. In: International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2014, Paris, France, May 5–9, 2014, pp. 1061–1068 (2014)
9. Frank, G., Guinn, C., Hubal, R., Pope, P., Stanford, M., Lamm-Weisel, D.: Justtalk: an application of responsive virtual human technology. In: Proceedings of the Interservice/Industry Training, Simulation and Education Conference (2011)
10. Huisman, G., Kolkmeier, J., Heylen, D.: With us or against us: simulated social touch by virtual agents in a cooperative or competitive setting. In: Bickmore, T., Marsella, S., Sidner, C. (eds.) IVA 2014. LNCS (LNAI), vol. 8637, pp. 204–213. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09767-1_25
11. Lazarus, R.S.: Cognition and motivation in emotion. Am. Psychol. **46**, 352–367 (1991)
12. Marsella, S., Gratch, J., Petta, P.: Computational models of emotion. Cross-fertilization between Emotion Psychology, Affective Neuroscience, and Affective Computing, A blueprint for an affectively competent agent (2010)
13. Marsella, S.C., Gratch, J.: EMA: a process model of appraisal dynamics. Cogn. Syst. Res. **10**(1), 70–90 (2009)
14. Nieuwenhuys, A., Oudejans, R.R.D.: Training with anxiety: short- and long-term effects on police officers' shooting behavior under pressure. Cogn. Process. **12**(3), 277–288 (2011)
15. Rickel, J.: Intelligent virtual agents for education and training: opportunities and challenges. In: de Antonio, A., Aylett, R., Ballin, D. (eds.) IVA 2001. LNCS (LNAI), vol. 2190, pp. 15–22. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44812-8_2
16. Rizzo, A., Reger, G., Gahm, G., Difede, J., Rothbaum, B.O.: Virtual reality exposure therapy for combat-related PTSD. In: Post-Traumatic Stress Disorder, pp. 375–399 (2009)
17. Schroeder, M., Bevacqua, E., Cowie, R., Eyben, F., Gunes, H., Heylen, D., ter Maat, M., McKeown, G., Pammi, S., Pantic, M., Pelachaud, C., Schuller, B., de Sevin, E., Valstar, M., Woellmer, M.: Building autonomous sensitive articial listeners. IEEE Trans. Affect. Comput. **3**(2), 165–183 (2012)
18. Teh, J.K.S., Cheok, A.D., Peiris, R.L., Choi, Y., Thuong, V., Lai, S.: Huggy pajama: A mobile parent and child hugging communication system. In: Proceedings of the 7th International Conference on Interaction Design and Children, pp. 250–257 (2008)
19. Traum, D., Swartout, W., Khooshabeh, P., Kopp, S., Scherer, S., Leuski, A. (eds.): IVA 2016. LNCS (LNAI), vol. 10011. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47665-0
20. Ben Youssef, A., Chollet, M., Jones, H., Sabouret, N., Pelachaud, C., Ochs, M.: Towards a socially adaptive virtual agent. In: Brinkman, W.-P., Broekens, J., Heylen, D. (eds.) IVA 2015. LNCS (LNAI), vol. 9238, pp. 3–16. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21996-7_1
21. Zhang, E.Y., Cheok, A.D.: A networked device for reproducing multisensory kissing. In: Proceedings of the 2016 workshop on Multimodal Virtual and Augmented Reality (2016)
22. Zhao, Y.C., Kennedy, G., Yukawa, K., Pyman, B., O'Leary, S.: Can virtual reality simulator be used as a training aid to improve cadaver temporal bone dissection? results of a randomized blinded control trial. Laryngosc. **121**(4), 831–837 (2011)
23. Zoll, C., Enz, S., Schaub, H., Aylett, R., Paiva, A.: Fighting bullying with the help of autonomous agents in a virtual school environment. In: Proceedings of the 7th International Conference on Cognitive Modelling (ICCM) (2006)

Tracking Perceptual and Memory Decisions by Decoding Brain Activity

Marieke van Vugt[1(✉)], Armin Brandt[2], and Andreas Schulze-Bonhage[2]

[1] Institute of Artificial Intelligence & Cognitive Engineering, University of Groningen, Groningen, The Netherlands
m.k.van.vugt@rug.nl

[2] Epilepsy Center, University Medical Center Freiburg, Freiburg im Breisgau, Germany

Abstract. Decision making is thought to involve a process of evidence accumulation, modelled as a drifting diffusion process. This modeling framework suggests that all single-stage decisions involve a similar evidence accumulation process. In this paper we use decoding by machine learning classifiers on intracranially recorded EEG (iEEG) to examine whether different kinds of decisions (perceptual vs. memory) exhibit dynamics consistent with such drift diffusion models. We observed that decisions are indeed decodable from brain activity for both perceptual and memory decisions, and that the time courses for these types of decisions appear to be quite similar. Moreover, the high spatial resolution of iEEG revealed that perceptual and memory decisions rely on slightly different brain areas. While the accuracy of decision decoding can still be improved, these initial studies demonstrate the power of decoding analyses for testing computational models of cognition.

1 Introduction

Decision making is a basic cognitive process that comes to play in many different tasks. Most of the research on decision making focuses on simple tasks such as detecting the direction of randomly moving dots. The theories developed on the basis of those experiments presume that all decisions between two alternatives (at least those consisting of a single stage process) behave with similar dynamics. Specifically, according to drift diffusion models (DDMs; [1]), decisions follow a drifting diffusion process, where the random walk is driven by the decision information. The DDM starts the process of evidence accumulation at the moment the stimulus comes on the screen, and then slowly drifts towards one of the decision thresholds which each correspond to a particular decision option. As soon as the decision threshold is reached, the response corresponding to the relevant decision option is given. The model has been found to produce excellent fits to performance in a variety of tasks, as well as the detailed shape of the associated response time distributions. The parameters of this model each can be interpreted as specific cognitive processes such as attention allocation in the

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 76–85, 2018.
https://doi.org/10.1007/978-3-319-76892-2_6

drift rate parameter and speed-accuracy trade-off in the location of the decision threshold [2]. In addition to the drift rate and the decision threshold, the third main parameter of the model is the non-decision time, which reflects non-decision-related processes such as preparing a motor response and fixed delays in the perceptual system. By varying the values of these parameters, subtle differences in shapes of the response time distributions can be reproduced.

While DDMs were developed exclusively based on behavioral data, more recently it has also been suggested that the brain may implement such diffusion processes. For example, in seminal work, Shadlen and colleagues observed monotonously increasing firing rates of neurons in the lateral intraparietal area while monkeys were deciding about the direction of randomly moving dots [3,4]. This neural signature was modulated by the strength of the decision evidence (the proportion of dots moving coherently) and the traces seemed to all move up to the same final firing rate around the time of the response. Subsequent studies on monkeys in similar tasks instead placed evidence accumulation in the frontal eye fields [5–9]. Some of the differences between studies could be traced back to the response modality (e.g., accumulation-like activity is more likely in frontal eye field when monkeys use saccades to indicate their response than when they use reaching).

In humans, accumulation processes have been studied as well, although in that case the challenge is the trade-off between poor temporal resolution of functional magnetic resonance imaging (fMRI) and the poor spatial resolution of electroencephalograph (EEG). FMRI studies have suggested evidence accumulation may take place in the dorsolateral prefrontal cortex [10], inferior frontal gyrus [11–13] but as demonstrated in a meta-analysis, in fact almost the whole brain [14]). Using EEG, we found neural correlates of evidence accumulation in parietal 4–9 Hz theta oscillations when people were making decisions about randomly moving dots [15]. MEG (magnetoencephalography) studies have implicated different brain regions in the accumulation process, such as 14–24 Hz beta oscillations over motor cortex; [16]. In addition to these brain oscillations, it has been suggested that two event-related potentials–the centroparieto potential (CPP; [17]), and the lateralized readiness potential [18]–reflect evidence accumulation. While the CPP may arise from parietal cortex, the lateralized readiness potential may come from premotor areas of the brain. However, none of this localization is very specific since it is derived from scalp-recorded EEG, which has poor localization.

An alternative approach to localizing the decision process in the brain has been to use classifiers, which are increasingly popular in neuroscience. The first studies using these methods focused on finding specific moments in time at which decisions can be best classified, rather than tracking the complete decision process over time. For example, Ratcliff and colleagues [19] found that a logistic regression-based classifier in the period around 400 ms post-stimulus exhibited behavior consistent with evidence accumulation during a face-car discrimination task–the output of this classifier covaried with between-trial differences in the drift rate. Philliastides and colleagues [20] followed up on this in a similar

face-house discrimination task and showed that a Fisher discriminant analysis could also track the decision process over time and this process appeared to be predominantly localized to parietal regions.

While these results are promising, they do not give very detailed localization of the decision process and restrict themselves to the cortex due to the inherent constraints of EEG data. Moreover, as is clear from the above discussion, different studies have claimed that evidence accumulation occurs in many different brain areas, dependent on the study. One potential reason for contradictory results may be that decisions on the basis of different kinds of evidence may be implemented by different brain regions. For this reason, it is worthwhile to examine whether evidence accumulation looks similar for different kinds of decisions, such as perceptual decisions and decisions about remembered information.

To enhance spatial localization we decode decision information from intracranially-recorded brain oscillations (rather than the scalp EEG used in most previous studies). Intracranial EEG is data with a high degree of spatial and temporal precision that can be obtained from epileptic patients who are implanted with electrodes for clinical purposes [21]. To determine what brain areas are involved in decision making, and where the decision information is available over time, we ran a regularized logistic regression classifier in short (50-ms) time bins, and assessed how classification accuracy develops for memory and perceptual decisions. We focused on classifying 4–9 Hz theta oscillations, since we previously demonstrated that those are most informative for decision making [15]. We then looked at the classifier weights to determine what Brodmann areas carry most of the decision information, and whether those differed between perceptual and memory decisions.

2 Methods

2.1 Participants

Participants were recruited from the patients undergoing long-term invasive monitoring for pharmacologically intractable epilepsy at Freiburg University hospital (Germany). Sixteen individuals were recruited and participated in our behavioral experiments.

2.2 Task

To be able to compare perceptual and memory decisions, we created a task with perceptual and memory conditions that used the same set of stimuli. We created synthetic face stimuli by means of the Basel Face model [22], which allowed us to manipulate the stimulus similarity (and hence task difficulty) very precisely. In the perceptual condition (Fig. 1(a)), participants saw two faces facing outward and had to determine whether this face belonged to the same person (i.e., was simply a rotated version of the same face). In the memory condition (Fig. 1(b)), participants were first shown two faces during a 2000–2075 ms (jittered) study period, followed by a 1000–1150 ms (jittered) blank delay period, after which

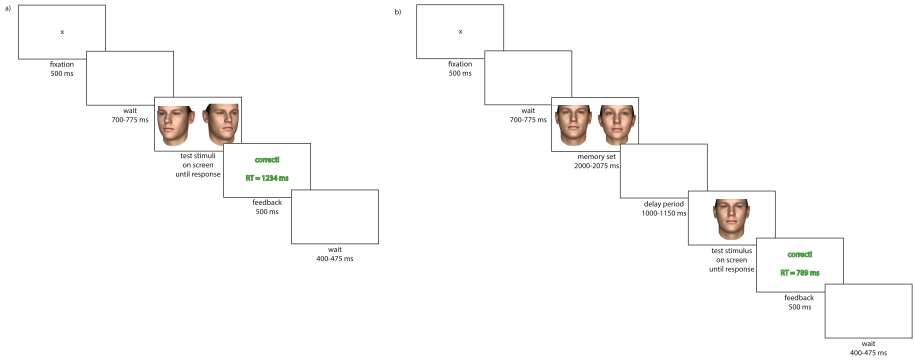


Fig. 1. Example trials of the perceptual (a) and memory (b) condition.

a probe face was shown. Participants were then asked to indicate by a button press whether this probe face was identical to one of the two faces presented in the study. The jitter in the task was used to ensure that no spurious oscillatory phase-locking could occur.

2.3 Recordings

Data were recorded with a 2000 Hz sampling rate on the clinical EEG recording system (Compumedics). We then segmented the data into trials of 4000 ms duration, starting 200 ms prior to the onset of the probe stimulus. We checked for the occurrence of epileptiform activity and one of the participants' data had to be discarded due to epileptic spikes in a majority of the trials. Trials whose response time exceeded the trial segment duration (3800 ms) were discarded, and the classifier analysis was done only on correct trials. Similarly, trials with a kurtosis larger than 15 (indicative of epileptic spikes) were removed. The total dataset involves 1178 electrodes.

2.4 Data Analysis

Data were analyzed by means of in-house matlab code that was based on toolboxes developed by Jelmer Borst and Per Sederberg. First, we ensured that the two classes to be separated (match/non-match decisions) had an equal number of trials by randomly removing trials from the larger class. We then performed a wavelet transform to obtain EEG time courses in the 4–9 Hz theta band, which previously has been shown to be important for decision making [15]. Next, we z-transformed all trials to ensure the data had an average of zero and a standard deviation of one. We then vincentized the data–that is, we turned each trial into an equal number of bins between the stimulus and the response (with the exception of the first 300 ms, which contains roughly the same peaks irrespective of the response time, so this period was not stretched or compressed and simply divided into 6 bins of 50 ms duration). The number of bins was chosen such that on average, bin duration would be approximately 50 ms.

For each time bin, we trained a regularized logistic regression model to distinguish between the match and non-match responses. Essentially, this model tries to find a matrix $\mathbf{W}$ that maps between the $n \times p$ matrix of examples $\mathbf{X}$ (p is the number of features, n is the number of trials) and the vector of labels (match or non-match) Y: $\mathbf{W} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^T Y$. In this equation, I_p is the $p \times p$ identity matrix and λ is the regularization matrix. The regularization allows the algorithm to deal with many correlated predictors. For each classifier the regularization parameter lambda was determined by means of a search between 0 and 10,000 [23]. The lambda that minimized the root-mean-square prediction error across all labels was chosen. We then assessed the classifier's performance using 10-fold cross-validation.

3 Results

3.1 Accuracy Across Subjects

We first examined how well decisions could be decoded from intracranial EEG data per participant. Figure 2 illustrates the maximum accuracy across the

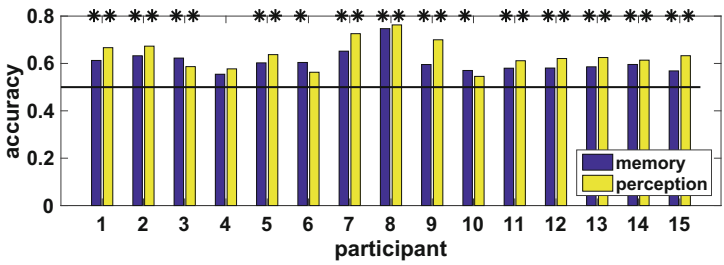


Fig. 2. Maximum decision decoding accuracy for each participant, separately for perceptual and memory trials. Stars indicate classification accuracies that are larger than chance.

Table 1. DDM parameters (mean and standard error of the mean). Perceptual and memory trials were split between low and high-similarity conditions (difficult and easy, respectively). Decision threshold was kept fixed between all conditions. Non-decision time was fixed between the similarity conditions. This model came out as best from a BIC comparison of various model configurations.

Condition	Drift	Decision threshold	Non-decision time (s)
Perception low similarity	−0.41 (0.13)	0.27 (0.025)	0.92 (0.15)
Perception high similarity	0.24 (0.066)	0.27 (0.025)	0.92 (0.15)
Memory low similarity	−0.11 (0.014)	0.27 (0.025)	0.50 (0.047)
Memory high similarity	0.11 (0.016)	0.27 (0.025)	0.50 (0.047)

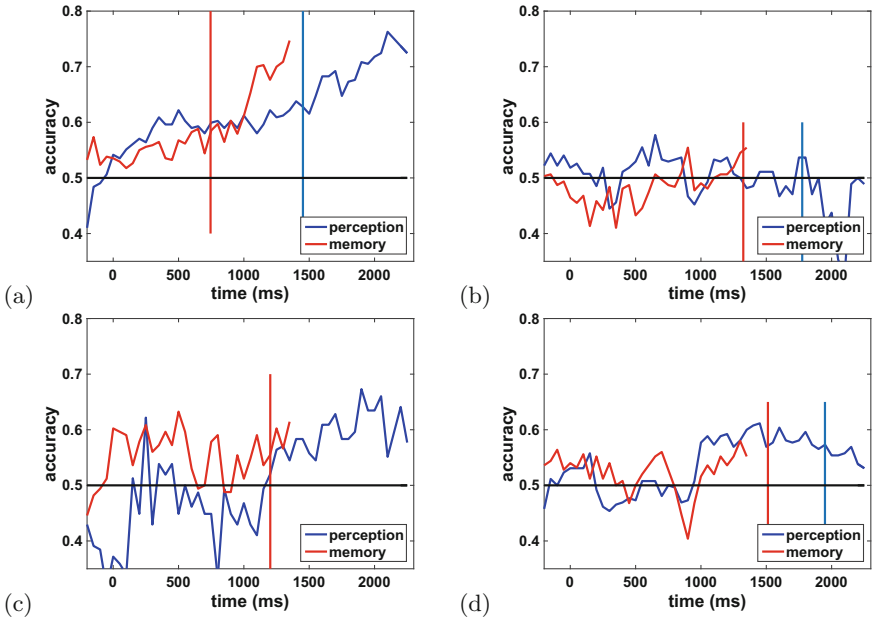


Fig. 3. Time courses of classification accuracy in the 4–9 Hz theta band for a participant with good classification (a) and for a participant with poor classification (b). For comparison, two other participants are also shown ((c) and (d)). Vertical lines indicate the average time of the response for the relevant condition (blue for perception, red for memory). (Color figure online)

decision interval (between the moment the probe stimulus came on the screen and the response). As can be seen, there are large differences between individuals in classification accuracy, which ranges from close to chance (55%) to 76%. Most accuracies significantly exceed chance according to a binomial proportion test (see stars in Fig. 2; $p < 0.016$, reflecting a 5% False Discovery Rate).

Having established that decision classification is possible to some extent, we can now examine our main question: what is the time course of these classifications, and does it differ between perceptual and memory decisions? Figure 3 demonstrates that while for some participants there is no classification possible, and the signal hovers around chance level, for others there is meaningful classification, and consistent with the DDM, classification accuracy increases slowly over time (the slope of the classification accuracy is larger than zero for both memory decisions ($t(14) = 2.55$, $p = 0.012$) and for perceptual decisions ($t(14) = 3.04$, $p = 0.0044$)). Unexpectedly, classification accuracy appears to continue to increase even after the response has been made. One possible interpretation of such a pattern is provided by recent modeling studies that suggest that after the decision has been made, the accumulation process continues with the objective of estimating decision confidence [24].

Comparing the two conditions, classifier accuracy appears to get higher for perception than for memory trials ($M_{memory} = 0.61$; $M_{perception} = 0.64$; $t(14) = 2.82$, $p = 0.014$). In short, while perceptual and memory decisions involve quite different tasks and response times, the decodable decision information forms quite a similar trajectory for both, consistent with the DDM predictions. The only difference is that the classifiers start to increase at different points in time, presumably because the response time in the perception condition is significantly longer than in the memory condition. This difference in response time is presumably caused by the fact that in the perception condition, participants need to first mentally rotate the images before they can make their decision. DDM fits are consistent with this idea (see parameters in Table 1): there is a significant effect of task on the non-decision time parameter ($F(1, 60) = 14.1$, $p = 0.0004$, similar to previous studies of mental rotation decisions [27]). Another interesting finding is that classification accuracy appears to continue increasing even after the decision has been made. This is consistent with some findings from experiments with monkeys suggesting that after the decision has been made, participants continue to accumulate information to make estimates of their confidence in the decision.

3.2 How Do Different Brain Areas Contribute to Classification?

Next, we asked what brain areas are involved in classification, and how these regions differ between perceptual and memory decisions. Specifically, for every Brodmann area, we reported the proportion of electrodes in that area that were significantly involved in evidence accumulation (i.e., having z-scores larger than 2).

Table 2. Proportion of significant electrodes by Brodmann area for classification of target-lure/match-nonmatch decisions on the basis of 4–9 Hz theta activity. The Brodmann areas are ordered by proportion of significant electrodes. Only Brodmann areas with more than 15 electrodes are included. Brodmann areas for which the proportion of significant electrodes is zero: 9, 22, 28,40, 47 and hippocampus.

Brodmann area	Memory	Perception	$N_{electrodes}$	$N_{participants}$
Brodmann area 7	0.17	0.06	18	1
Brodmann area 13	0.03	0.09	34	10
Brodmann area 20	0.01	0.07	179	13
Brodmann area 36	0.02	0.07	46	9
Brodmann area 37	0.01	0.04	98	12
Brodmann area 41/42	0.03	0.03	33	8
Amygdala	0.00	0.03	31	9
Brodmann area 21	0.01	0.03	246	13
Brodmann area 19	0.02	0.00	57	9
Brodmann area 38	0.02	0.00	101	10

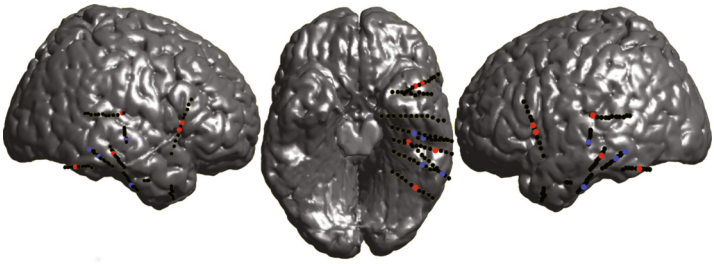


Fig. 4. Electrodes that contribute significantly ($z > 2$) to decision classification. Black dots indicate electrodes that are not showing significant decision-related activity. Red: perceptual decisions. Blue: memory decisions. The electrode locations for two participants were missing, so those are not included in this figure and in Table 1. In addition, Brodmann areas having no electrodes in our dataset are also excluded from the table. (Color figure online)

Figure 4 shows the electrodes that were, across participants most involved in classification, separately for perception and memory decisions. It is clear that most of these electrodes are in lateral parietal and temporal areas. Part of the reason for that is of course that the locations that are relevant for clinical purposes tend to be temporal areas, which are often the sources of epileptic seizures. Table 2 demonstrates that the Brodmann areas that have the largest proportion of electrodes carrying decision information are parietal areas (Brodmann area 7) and perceptual-motor areas (Brodmann areas 1-2-3-5).

4 Discussion

We examined the time course of the availability of decision information during a perceptual and a memory decision task. As the DDM predicted, overall the decoded decision evidence shows similar dynamics for perceptual and memory decisions. The difference between the two lies in the time at which they move upwards, which is later for the perception condition than for the memory condition. This is not surprising given that for the perception task the response time is significantly longer than in the memory task. The second question we asked was whether accumulation in the perception and memory conditions relies on different brain regions. Because we have no full brain coverage we can only make tentative claims, but the data so far suggest that there are differences between those two types of decisions (see also Fig. 4). While memory decisions rely predominantly on Brodmann area 7 (parietal cortex) and sensorimotor cortex, perceptual decisions rely more on Brodmann area 13 (anterior insula) and 30 (visual area).

While we did obtain classification accuracies above chance, classification is far from stellar. Potentially other classifiers such as lasso or artificial neural nets could do better than these. On the other hand, such classifiers run a higher risk of overfitting the data. Another potential approach could be to make use of the

known similarity structure of the face stimuli. Previous studies have shown that such decoded similarity structures can help to track memory representations in the brain [25]. In addition, we focused here on the theta band, because that frequency was suggested by prior studies. Nevertheless, it could easily be the case that for these data, better decision information could be decoded from other frequencies. Finally, there are large differences between individuals. We examined whether individual differences in task performance (accuracy and response time) could account for these differences in classification accuracy. We found that the data were too unreliable to make any connection between task performance and classifier accuracies (all Bayes Factors between 0.4 and 2.2).

Another weaker part of this study is that it did not make a direct connection to DDM parameters. If the dynamics of the decoded decision process were to covary with model parameter estimates [15], this would bolster our confidence that we are in fact observing neural correlates of a drift diffusion process. One approach that we can use in the future to examine this is to use the classifier readout to separate the trials into low-evidence and high-evidence, and then to fit the DDM separately to these classes of trials. Previous studies using such an approach have demonstrated that such within-participant model verification can be used to identify neural correlates of drift diffusion processes [19,26].

In short, we have demonstrated how decoding decision information from brain data can help us to better understand the dynamics of decision making. This builds on a larger body of work that uses classifiers to track covert cognitive processes over time. As classifiers become more powerful and can deal with more noisy data, these are very useful tools to help us uncover how the brain "does" cognition. Moreover, we will gain a better and better time-resolved picture of cognitive processes, which can subsequently inform computational models of these processes.

References

1. Ratcliff, R.: A theory of memory retrieval. Psychol. Rev. **85**, 59–108 (1978)
2. Ratcliff, R., Smith, P.L., Brown, S.D., McKoon, G.: Diffusion decision model: current issues and history. Trends Cogn. Sci. **20**(4), 260–281 (2016)
3. Shadlen, M.N., Newsome, W.T.: Motion perception: seeing and deciding. Proc. Natl. Acad. Sci. U.S.A. **93**, 628–633 (1996)
4. Roitman, J.D., Shadlen, M.N.: Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. J. Neurosci. **22**(21), 9475–9489 (2002)
5. Ferrera, V.P., Yanike, M., Cassanello, C.: Frontal eye field neurons signal changes in decision criteria. Nat. Neurosci. **12**, 1458–1462 (2009)
6. Hanes, D.P., Schall, J.D.: Neural control of voluntary movement initiation. Science **274**, 427–430 (1996)
7. Purcell, B.A., Heitz, R.P., Cohen, J.Y., Schall, J.D., Logan, G.D., Palmeri, T.P.: Neurally constrained modeling of perceptual decision making. Psychol. Rev. **117**(4), 1113–1143 (2010)
8. Ding, L., Gold, J.I.: Neural correlates of perceptual decision making before, during, and after decision commitment in monkey frontal eye field. Cereb. Cortex **22**, 1052–1067 (2012)

9. Ding, L., Gold, J.I.: Caudate encodes multiple computations for perceptual decisions. J. Neurosci. **30**, 15747–15759 (2010)
10. Heekeren, H.R., Marrett, S., Ruff, D.A., Bandettini, P.A., Ungerleider, L.G.: Involvement of human left dorsolateral prefrontal cortex in perceptual decision making is independent of response modality. Proc. Nat. Acad. Sci. U.S.A. **103**(26), 10023–10028 (2006)
11. Ho, T., Brown, S., Serences, J.T.: Domain general mechanisms of perceptual decision making in human cortex. J. Neurosci. **29**(27), 8675–8687 (2009)
12. Krueger, P.M., van Vugt, M.K., Simen, P., Nystrom, L., Holmes, P., Cohen, J.D.: Evidence accumulation detected in BOLD signal using slow perceptual decision making. J. Neurosci. Meth. **281**, 21–32 (2017)
13. Ploran, E.J., Nelson, S.M., Velanova, K., Donaldson, D.I., Petersen, S.E., Wheeler, M.E.: Evidence accumulation and the moment of recognition: dissociating perceptual recognition processes using fMRI. J. Neurosci. **27**(44), 11912–11924 (2007)
14. Mulder, M.J., van Maanen, L., Forstmann, B.U.: Perceptual decision neurosciences - a model-based review. Neuroscience **277**, 872–884 (2014)
15. van Vugt, M.K., Simen, P., Nystrom, L., Holmes, P., Cohen, J.D.: EEG oscillations reveal neural correlates of evidence accumulation. Front. Hum. Neurosci. **6**, 106 (2012)
16. Donner, T., Siegel, M., Fries, P., Engel, A.K.: Buildup of choice-predictive activity in human motor cortex during perceptual decision making. Curr. Biol. **19**, 1581–1585 (2009)
17. O'Connell, R.G., Dockree, P.M., Kelly, S.P.: A supramodal accumulation-to-bound signal that determines perceptual decisions in humans. Nat. Neurosci. **15**(12), 1729–1735 (2012)
18. van Vugt, M.K., Simen, P., Nystrom, L., Holmes, P., Cohen, J.D.: Lateralized readiness potentials reveal properties of a neural mechanism for implementing a decision threshold. PLoS ONE **9**(3), e90943 (2014)
19. Ratcliff, R., Philiastides, M., Sajda, P.: Quality of evidence for perceptual decision making is indexed by trial-to-trial variability of the EEG. Proc. Nat. Acad. Sci. U.S.A. **106**(16), 6539 (2009)
20. Philiastides, M.G., Heekeren, H.R., Sajda, P.: Human scalp potentials reflect a mixture of decision- related signals during perceptual choices. J. Neurosci. **34**(50), 16877–16889 (2014)
21. Jacobs, J., Kahana, M.J.: Direct brain recordings fuel advances in cognitive electrophysiology. Trends Cogn. Sci. **14**(4), 162–171 (2010)
22. IEEE: A 3D Face Model for Pose and Illumination Invariant Face Recognition, Genova, Italy. IEEE (2009)
23. Borst, J., Schneider, D.W., Walsh, M.M., Anderson, J.R.: Stages of processing in associative recognition: evidence from behavior, EEG, and classification. J. Cogn. Neurosci. **25**(12), 2151–2166 (2013)
24. Pleskac, T., Busemeyer, J.R.: Two-stage dynamic signal detection: a theory of choice, decision time, and confidence. Psychol. Rev. **117**(3), 864–901 (2010)
25. Zhang, H., Fell, J., Staresina, B., Weber, B., Elger, C.E., Axmacher, N.: Gamma power reductions accompany stimulus-specific representations of dynamic events. Curr. Biol. **25**, 635–640 (2015)
26. Ratcliff, R., Sederberg, P.B., Smith, T., Childers, R.: A single trial analysis of EEG in recognition memory: tracking the neural correlates of memory strength. Neuropsychologia **93**, 128–141 (2016)
27. Provost, A., Heathcote, A.: Titrating decision processes in the mental rotation task. Psychol. Rev. **122**(4), 735–754 (2015)

The Origin of Mimicry
Deception or Merely Coincidence?

Bram Wiggers[1(✉)] and Harmen de Weerd[1,2]

[1] Institute of Artificial Intelligence, University of Groningen, Groningen, The Netherlands
bramwiggers94@gmail.com
[2] Professorship User Centered Design, Hanze University of Applied Sciences, Groningen, The Netherlands

Abstract. One of the most remarkable phenomena in nature is mimicry, in which one species (the mimic) evolves to imitate the phenotype of another species (the model). Several reasons for the origin of mimicry have been proposed, but no definitive conclusion has been found yet. In this paper, we test several of these hypotheses through an agent based co-evolutionary model. In particular, we consider two possible alternatives: (1) Deception, in which mimics evolve to imitate the phenotype of models that predators avoid to eat, and (2) Coincidence, in which models evolve a warning color to avoid predation, which coincidentally benefits the mimics. Our agent-based simulation shows that both these hypotheses are plausible origins for mimicry, but also that once a mimicry situation has been established through coincidence, mimics will take advantage of the possibility for deception as well.

1 Introduction

One of the most remarkable phenomena in nature is mimicry, in which one species (*mimic* animals) imitates the phenotype of another species (*model* animals). Typically, the effect is called mimicry when the model species are dangerous to predators. In this case, the mimic species benefits from mimicry when predators mistake the mimic animals for model animals. Depending on characteristics of the mimic species, the model species may benefit (Müllerian mimicry, [11]) or suffer (Batesian mimicry, [1]) from the presence of mimic animals. In this paper, we investigate two possible hypotheses for the origin of mimicry through an agent-based co-evolutionary model.

The pioneer in mimicry research was Bates [1]. Bates found that there are poisonous animals with very bright colors, and camouflaged animals which were not poisonous. Even though the brightly colored animals are more easily detected by predators, they were also identified as dangerous by these predators. This so-called *aposematism* effect became more remarkable when Bates found animals with similar colors and shapes as the toxic animals that were not toxic. This type of mimicry is called Batesian mimicry, in which non-toxic animals imitate

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 86–100, 2018.
https://doi.org/10.1007/978-3-319-76892-2_7

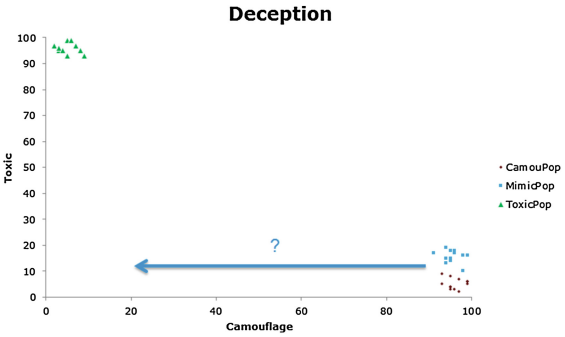


Fig. 1. In deception, the mimic and camouflaged animals start with the same characteristics. The hypothesis states that the mimicry group will move toward the place with low toxicity and camouflage: the mimicry place, since this place has the lowest evolutionary cost. (Color figure online)

the phenotype of toxic animals. This effect has been found in butterflies [1,11], snakes [12], and various other animals [9].

In Batesian mimicry, mimic animals are not toxic. As a result, whenever a mimic animal is eaten or tasted by a predator and found to be harmless, this gives positive feedback to the predator to eat similar animals. This results in model animals being eaten more. Hence, the more mimic animals exist in the habitat of the model, the lower the survival chance of the model. This results in a negative or parasitical effect on the model. Müllerian mimicry [11], on the other hand, involves two species of animals that are both toxic to a certain degree, and therefore both contribute to this anti-predation mechanism.

In this paper, we investigate two possible hypotheses for the origin of Batesian mimicry. A common assumption in the literature is that the mimic animals deliberately deceive their predators by imitating model animals [7,12]. That is, mimic animals evolve to have the same phenotype as the model animals because this lowers predation. However, mimicry may also come about through coincidence. That is, model animals may evolve a phenotype that allows predators to distinguish them, and which happens to be the phenotype of the mimic animals.

These two hypotheses will be tested through an agent-based co-evolution model. Agent-based modeling has proven its usefulness as a research tool to investigate how behavioral patterns may emerge from the interactions between individuals (cf. [4,5]). Among others, agent-based models have been used to explain fighting in crowds [8], the evolution of cooperation and punishment [2,13], and the evolution of language [3]. In this paper, we use agent-based modeling to test two hypotheses on the origins of mimicry. We will elaborate on the hypotheses in the next two sections.

1.1 Deception Hypothesis

The deception hypothesis reflects the typical assumption about mimicry. According to the deception hypothesis, mimic animals evolve to have a phenotype that

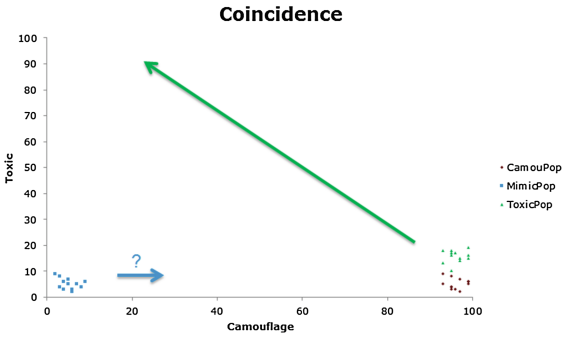


Fig. 2. In coincidence, the toxic model animals and the camouflaged animals start with the same characteristics. The mimicry group is already in place. The toxic population now gets more toxic, and moves toward the lesser camouflaged site. (Color figure online)

is as similar as possible to the phenotype of model animals. This benefits mimic animals because they are mistaken for animals that are dangerous to predators. Therefore, if model animals are less camouflaged, the mimic animals are expected to evolve a lower level of camouflage as well.

This idea is represented graphically in Fig. 1. The figure shows three types of prey: model animals (ToxicPop), mimic animals (MimicPop), and a control population of camouflaged animals (CamouPop). Only the model animals are dangerous to the predators, as shown by a high level of toxicity, while both the control and mimic animals start out with high levels of camouflage. The deception hypothesis predicts that, over time, the mimic animals take advantage of the eating behavior of predators and evolve lower levels of camouflage (blue arrow in Fig. 1), since being camouflaged has a negative influence on their chance to reproduce. That is, the deception hypothesis describes a process of *speciation*, where one population of prey splits into a population of mimic and a population of control animals.

1.2 Coincidence Hypothesis

In contrast with the deception hypothesis, the coincidence hypothesis describes a small role for the mimic animals. The coincidence hypothesis describes the situation in which the mimic animals do not change their phenotype, but that the model animals evolve a distinct phenotype, which happens to be the same as the phenotype of the mimic animals. Note that in this situation, there is an important role for the behavior of the predators. Predators learn to create a discriminatory line between the model animals and the control population, which drives the selective pressure for model animals to evolve a phenotype that is distinct from that of the control animals. The coincidence hypothesis states that the mimic animals may happen to be on the model animal side of this line, and therefore experience a coincidental benefit.

The coincidence hypothesis is described graphically in Fig. 2. The figure shows three types of prey: model animals (ToxicPop), mimic animals (MimicPop), and a control population of camouflaged animals (CamouPop). All these populations start out with low levels of toxicity, and are therefore harmless for the predators. In addition, the mimic animals start out with low camouflage, while the control and model animals have high levels of camouflage. The coincidence hypothesis predicts that when the model animals evolve higher levels of toxicity, they will also decrease their level of camouflage (long green arrow in Fig. 2). In addition, the coincidence hypothesis predicts that the mimic population would not increase its camouflage to more closely resemble the model animals (short blue arrow in Fig. 2). Note that the coincidence hypothesis also describes a process of *speciation*, but in this case, one population of prey splits into a population of model and a population of control animals.

The idea behind the coincidence hypothesis is that there are relatively few mimic animals who already have a distinctive color because of pre-adaptation [7]. The model animals experience selective pressure towards the phenotype of the mimic animals because of the relative low population sizes of the mimic animals compared to the population size of the control animals.

1.3 Structure of the Paper

The remainder of this paper is set up as follows. In Sect. 2, we will discuss the simulation model, first in general terms and then in more depth. We present our simulation results in Sect. 3. Section 4 will discuss the results and provides directions for future research.

2 Model

While mimicry is defined in terms of the evolutionary behavior of prey animals, mimicry also depends on the behavior of predator animals. As a result, there are three different ways to study mimicry [10]:

- The evolutionary dynamics way, which studies the evolution of the prey but ignores the behavior of the predators [6];
- The receiver psychology way, which focuses on the behavior of the predators, but tends to ignore the evolution of the prey [7]; and
- The traditional natural historical way, which analyzes the behavior of both predator and prey. In these kinds of research, the co-evolution between predators and prey is studied [1].

In this paper, we follow the traditional natural historical way by explicitly modeling both the evolution of prey animals and the behavior of predator animals. To study this co-evolution, we construct an agent-based model that models individual prey and predator animals. In Sect. 2.1, we first give a general explanation of our model. A more technical discussion of the model can be found in Sect. 2.2.

2.1 Model Description

Our model of mimicry investigates the co-evolution of predator and prey animals. Prey animals are further subdivided into three separate populations, which we will call the *toxic*, *camouflaged*, and the *mimic* populations.

Predator agents perform two actions: eating prey and reproducing. A predator consists of a neural network that determines whether a predator will eat a prey that it encounters. This network is evolved, which means that predators do not learn over their lifetime, but instead inherit their decision function from their parent. At each time-step of the model, the predator encounters a number of prey. For each encountered prey, the predator decides whether or not to eat the prey, depending on the prey's phenotype. Eating non-toxic prey increases evolutionary fitness, while eating toxic prey decreases fitness. Reproduction occurs by selecting the agent with the highest fitness from a random sub-set of predators as the parent. The child inherits all characteristics of this parent, subject to a small probability of mutation, which will be elaborated on in Sect. 2.2.

Prey are defined by three characteristics: camouflage, toxicity, and pattern. A prey's camouflage determines the probability of being detected, so that a higher camouflage lowers the probability of being encountered by a predator. A prey's pattern, on the other hand, does not influence the probability of being encountered. A prey's phenotype consists of its camouflage and pattern. That is, both camouflage and pattern are observable characteristics, while toxicity is a characteristic that cannot be observed by predators.

Prey reproduce by selecting two parents with the highest fitness from a random subset of the population. The fitness of a prey is determined by the number of times it is eaten by a predator. In addition, both toxicity and camouflage decrease a prey's fitness.

To investigate mimicry, prey animals are subdivided into three separate populations of constant size that reproduce independently. The first population, called the *toxic* prey, has a small genetic drift toward higher toxicity. This population is meant to simulate model animals. Similarly, the *camouflaged* prey experience a small genetic drift towards higher camouflage, and are meant as a control population. The third, *mimic* population does not experience any genetic drift.

2.2 Model Details

In this section we look at the model in more detail. In particular, we take a closer look at the eating behavior and knowledge of the predator, the mechanism of reproduction, and the setup of different parameters.

Eating Behavior of Predators. During every time step of the model, each predator encounters a fixed number $Y_{encountered}$ of randomly selected prey animals (see also Algorithm 1). For each of these encounters, the camouflage of the prey animal determines the probability with which the prey is found, so that prey with a high camouflage are more likely to hide from the predators. If the predator finds the prey, it can choose whether or not to eat the prey.

Algorithm 1. Eating behavior of predators.

```
let C_y camouflage of prey y, scaled [−50, 50]
let P_y pattern of prey y, scaled [−50, 50]
let NN(X, Y) neural network function of predators
for each encountered prey y do
    if random(100) > C_y then                          ▷ Prey y is found
        if NN(P_y, C_y) > random(1) then
            Eat prey y
        end if
    end if
end for
```

The predator uses a simple feed-forward neural network to propagate the phenotype of the found prey, which results in a decision on whether or not to eat a prey. This neural network consists of two input nodes, which represent the prey's camouflage and pattern; three hidden nodes; and one output node that controls the predator's eating decision. The output node is implemented as a probability between 0 and 1, so that there is a low probability that predators will try to eat prey that they believe to be dangerous.

Each node is connected to all nodes in the next layer. This results in 6 synapses from the input nodes to the hidden nodes and 3 from the hidden nodes to the output node. The total number of synapses (weights) is thus 9 per predator. After each synapse round, an activation function is applied to scale the values between 0 and 1. The activation function used is the sigmoid function.

Reproduction. The mechanism with which animals reproduce is different for predators than it is for prey. However, for the selection of the parents, both types use tournament selection.

Predators reproduce asexually, so that every child has a single parent. A child inherits the neural network from its parent, subject to a low probability of mutation. When a weight mutates, a value between -0.25 and 0.25 is added to it. The value is then cut to the domain between -2 and 2. Because of the survival of the fittest principle, the best predators evolve and anticipate on the changes within the prey. This way of learning can be seen as a random search.

Prey, on the other hand, reproduces sexually. The two parents with the highest fitness are chosen with the tournament selection, and the child is a combination of these parents. Each child has camouflage, toxicity, and pattern that is the mean of the corresponding characteristic of its two parents. The values of the prey characteristics have a value between 0 and 100. Each characteristic has a low probability of mutation, in which case a random number between -10 and 10 is added to it. If the new value exceeds the borders of 0 or 100, it is cut off at that value. In the case of genetic drift, the `Genetic Drift` parameter is added to the mutation value, giving more chance for an increasing mutation.

For both predators and prey, reproduction occurs in generations. After each generation, all animals in the old generation die and are replaced by an identical

Table 1. Parameter settings used in the simulation runs.

Parameter	Predators	Toxic	Camouflaged	Mimic
Population size	10	300	300	30
Genetic drift	-	3	3	0
Prey encounters ($Y_{encountered}$)	3	-		
Mutation rate	20	2		
Tournament size	3	10		
Lifespan	3	5		
Chance-being-found	101	-		
Camouflage disadvantage	-	3.0		
Toxicity disadvantage	-	0.2		

number of new individuals. This means that there are no animals older than other animals, and that all animals die at the same moment after a predefined number of time-steps. This number of time-steps differs between predators and prey (see the lifespan parameter in Table 1) to reflect differences in learning. Note that prey animals do not die due to being eaten by a predator, but only die when their generation dies. Instead, the fitness of a prey decreases when it is eaten by a predator, reducing the chance for reproduction.

Parameters Settings and Fitness. Within our simulation, the number of predators is fixed, as well as the number of prey within each subpopulation. In every run of the model, there are fewer predators than prey, corresponding to the real world.

The predators have a lower life span than preys, to reflect that they learn faster than the rate at which prey evolves. The prey becomes older, which makes the difference in fitness between preys which are eaten and that are not eaten bigger.

Each time-step, a predator encounters the number of prey divided by the number of predators. This is multiplied by $Y_{encountered}$ to make the selective pressure higher. From the point of view of the prey, it has $Y_{encountered}$ encounters with predators.

In our model, each individual prey and predator represents a group of animals. For this reason, prey does not die when it is eaten. Instead, the fitness of a prey animal y is determined by the number of times it is 'eaten' (E_y). In our model, we assume both toxicity and camouflage to be detrimental to fitness. The toxicity (T_y) and camouflage (C_y) of prey y are multiplied by the toxic disadvantage (TD) and camouflage disadvantage (CD) parameters respectively. For example, a prey with a toxicity of 80 and toxicity disadvantage 0.2 will experience a 16 point penalty to its fitness. The fitness of a prey is updated according to

$$F_y = -(T_y \cdot TD) - (C_y \cdot CD) - (500 \cdot E_y). \tag{1}$$

Note that the most detrimental effect to the fitness of a prey is to be eaten. In addition, a prey's fitness is a non-positive number, with 0 being the highest possible value.

The fitness of a predator is determined by what prey it eats, according to the following formula

$$F_r = \sum_{y \in Eat_r} (T_y - 60). \tag{2}$$

The formula shows that the fitness of the predator $r(F_r)$ depends on the sum of the toxicity of all the prey it has eaten (Eat_r). Toxicity values are reduced by 60, so that predators increase their fitness whenever they eat a prey with toxicity lower than 60, and decrease their fitness otherwise.

3 Results

We used the model outlined in Sect. 2 to perform simulation runs, the results of which are discussed in this section. The results are divided in four different sections. In Sect. 3.1, we discuss the deception hypothesis. In Sect. 3.2, we investigate the coincidence hypothesis. For both these hypotheses, we show results from 100 runs of 14,000 time steps each. After this, in Sect. 3.3, we will discuss an individual run. Lastly, in Sect. 3.4 the difference between model animals and mimic animals in the different hypotheses will be discussed.

3.1 Deception Hypothesis

For the deception hypothesis (also see Fig. 1), the mimic and camouflaged populations start out with high camouflage. The toxic population starts at its final position with high toxicity and little camouflage (that is, brightly colored and toxic animals). The deception hypothesis predicts that the mimic population would evolve to decrease its camouflage while maintaining low toxicity.

Figure 3 shows the average camouflage of the three prey populations across 100 runs. The figure shows that the mimic population indeed decreases its level of camouflage over time. The camouflaged population also initially reduces its level of camouflage, but later returns to high camouflage levels. This can be explained by the genetic drift of the camouflaged population. However, the larger population size of the camouflaged prey also slows down the evolutionary process. In addition, predators find and eat more of the less camouflaged individuals than the more camouflaged individuals, which gives additional selective pressure to increase camouflage.

In contrast, Fig. 3 shows that the camouflage of the toxic population quickly drops and remains fairly stable. Since the toxic individuals rely on the predators' choice not to eat them, there is an evolutionary pressure to be as distinct as possible from other prey. In this case, the other prey have high camouflage, so the selective pressure encourages the toxic population to lower camouflage.

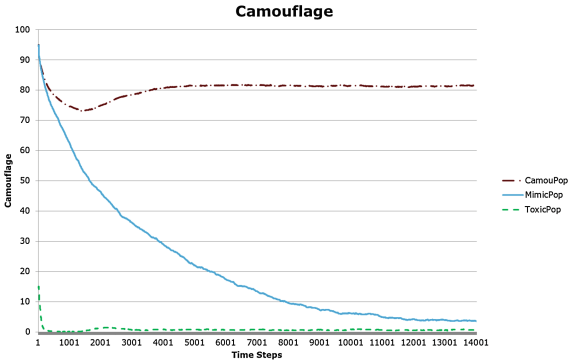


Fig. 3. The average camouflage value for each of the three prey populations across 100 runs for the deception hypothesis. The mimic and camouflaged populations start out with the same high camouflage, while the toxic population starts out with low camouflage.

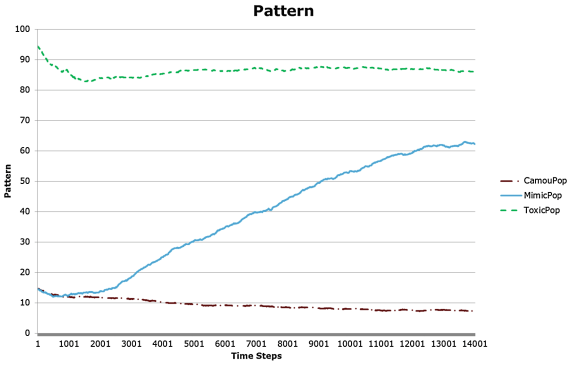


Fig. 4. The average pattern value for each of the three prey populations across 100 runs for the deception hypothesis. The mimic and camouflaged populations start out with the same pattern, which is different from the pattern of the toxic population.

Figure 4 shows the average pattern of the three prey populations across 100 runs. In this figure, we can see that the pattern of the mimics moves toward that of the toxic population. Both Figs. 3 and 4 support the deception hypothesis, since the mimic moves from a position of high camouflage towards a position of no camouflage, thereby mimicking the toxic population. In addition, the mimics also evolve to have the same pattern as the toxic group. That is, these results suggest that the mimics evolve to trick the predators into not eating them.

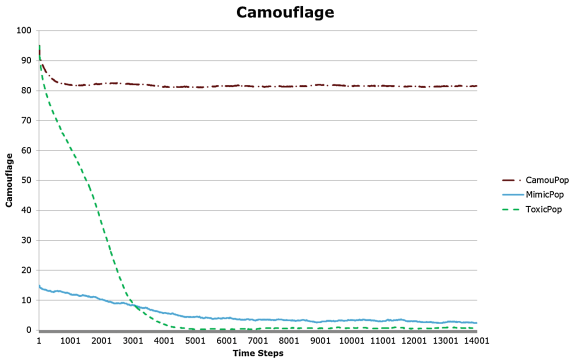


Fig. 5. The average camouflage value for each of the three prey populations across 100 runs for the coincidence hypothesis. The toxic and camouflaged population start out with the same high camouflage, while the mimic populations starts out with low camouflage.

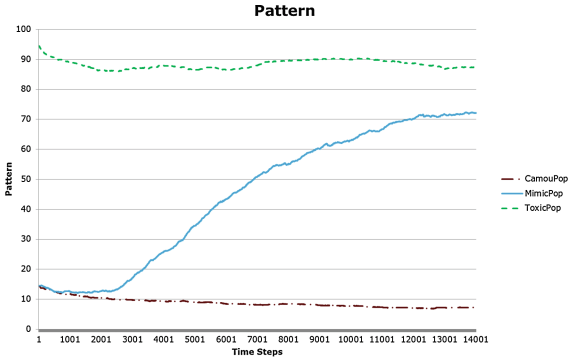


Fig. 6. The average pattern value for each of the three prey populations across 100 runs for the coincidence hypothesis. The mimic and camouflaged populations start out with the same pattern, which is different from the pattern of the toxic population.

3.2 Coincidence

In the coincidence hypothesis, the toxic and camouflaged group start at the same position, with low toxicity and high camouflage. The mimic population, on the other hand, starts out with low toxicity and low camouflage. The coincidence hypothesis predicts that, in order to distinguish itself from the camouflaged population, the toxic population evolves to a position with high toxicity and low camouflage, which coincidentally gives the same camouflage as the mimic population (also see Fig. 2).

Figure 5 shows the average camouflage of the three prey populations across 100 runs. Note that while the camouflaged population maintains high camouflage, the toxic population evolves to lower its camouflage. In addition,

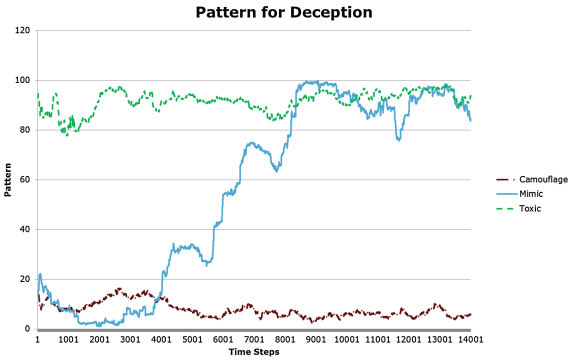


Fig. 7. The pattern of an individual run in the deception hypothesis. The pattern changes with two distinctive bumps, before being similar to the toxic population. After this we see the toxic population moving away from the pattern of the mimics, and the mimics chasing this pattern.

Fig. 5 shows that, on average, the mimic population does not increase its camouflage to increase resemblance with the toxic population.

Figure 6 shows the average pattern of the three prey populations across 100 runs. This figure shows that, while the mimic population does not increase its camouflage to resemble the toxic population, the mimic population does evolve to have a pattern that is similar to the toxic population. That is, while Fig. 5 supports the coincidence hypothesis, Fig. 6 is more suggestive of the deception hypothesis.

3.3 Individual Run

While the average results presented in the previous sections give a good impression of the way the prey's phenotype (i.e., camouflage and pattern) evolves over time, closer inspection shows that the average does not fit any individual run particularly well. For this reason, we take a closer look at a representative individual run in this section.

Figure 7 shows the evolution of pattern for all three prey populations of a representative individual run of 14,000 time-steps. Note that while the average results (Figs. 4 and 6) suggest that the pattern of the mimic population gradually evolves over 14,000 time steps, Fig. 7 suggests a more rapid evolution. Indeed, individual runs typically show a rapid evolution of the pattern of the mimic population. The average results show a more gradual development because the moment at which this rapid evolution starts is different for each run.

Figure 7 shows that the pattern of the mimic population not only evolves in the direction of the pattern of the toxic population, but also continues to

converge on the same value. In addition, due to the differences in population size of the mimic and toxic populations, the pattern of the mimic populations exhibits more volatility than that of the toxic population. This corresponds well with the idea of Holmgren and Enquist [7], who say:

> "For mimicry to be established, the movement of the mimic should always be faster than the movement of the model."

After approximately 9000 steps in the simulation, we notice the pattern of the model animals moves away from that of the mimic animals. When the mimics have a higher pattern, the models get a lower pattern and vice versa. This is consistent with the idea of [7]. In the results we can see that the pattern of the mimics changes faster, but the models try to distinguish themselves from the mimics. The reason for this is that model animals that are more similar to mimic animals are more likely to be eaten by predators, since mimics are harmless for predators. Model animals that look less like the mimics therefore have an evolutionary advantage, a development we can observe in the individual run of the model.

Figure 7 shows that the mimic population changes its pattern in several bumps. These bumps can be explained by the model animals getting less toxic. As a result, the predators start eating more model animals. When we observe the model animals, we can see that they start losing their toxicity when the predators do not eat them, since toxicity is detrimental to individual fitness. However, when the toxicity becomes too low, predators start eating more model animals. In Fig. 7, this effect is shown when the pattern of the mimics moves away from the pattern of the model animals.

3.4 Euclidean Distance Model and Mimic

Figure 8 shows the distance in phenotype between model and mimic over time. This graph tells us that the distance gets smaller, thus mimicry is created. The euclidean distance is measured as the distance between the mean of the pattern and camouflage between the mimics and the models by the following formula:

$$EUD = \sqrt{(C_{mim} - C_{mod})^2 + (P_{mim} - P_{mod})^2} \tag{3}$$

For the deception hypothesis, the mimic moves towards the model to trick the predators. Over time, we can see that the mimic population becomes increasingly more similar to the model population. This means that a lot of the simulations evolve into mimicry. For the coincidence hypothesis, the toxic group evolves towards the same camouflage as the mimics. After this, the mimics follow the models over the phenotype plane. As a result, the average euclidean distance gets smaller. Since both distances are decreasing, the model supports both hypotheses for mimicry.

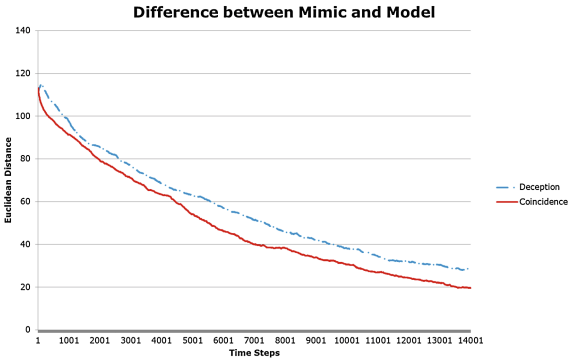


Fig. 8. The Euclidean Distance over time for both the hypotheses. In both hypotheses, the difference between model and mimic decreases over time, arguing for mimicry. The coincidence hypothesis has a little less difference. Keep in mind this is an average over 100 runs, and therefore is not a representation of one run, but a probability of mimic and model being the same.

4 Discussion

In this paper, we constructed an agent-based co-evolutionary model to investigate two possible origins of mimicry. The deception hypothesis predicts that due to selective pressure, mimic animals change their phenotype to resemble the model animals. In contrast, the coincidence hypothesis states that due to selective pressure, model animals change their phenotype to be different from some control population that has high levels of camouflage, and coincidentally get a phenotype that is similar to the mimic animals.

For both these possible origins of mimicry, we can say that they are plausible. The deception has very clear evidence in the deception set-up, where we can see that the mimic animals always change their phenotype to more resemble the phenotype of the model animals. This suggests that mimic animals indeed evolve to deceive predators. For the relatively small population of mimics, it is possible to explore new peaks in the adaptive plane, and successfully deceive predators. For the larger camouflage population, we can see that this population is too big to explore new adaptive peaks.

Our model results also show evidence for the coincidence hypothesis. Given the appropriate starting conditions, the model animals may change their phenotype to more resemble the mimic animals rather than vice versa. However, this can also be explained by the assumed negative fitness contribution of camouflage. This alone may result in both model and mimic animals to experience a selective pressure to reduce camouflage. Indeed, when we consider the pattern alone, the mimics again attempt to deceive the predators by evolving a pattern that resembles that of the model animals.

Note that we considered two different setups to determine the plausibility of the two hypotheses for the origin of mimicry. Both of these setups resulted in

the same ecosystem, with no possibility to determine what was the initial setup. That is, while our results show support for both hypotheses, they do not allow us to draw conclusions about which hypothesis matches best with biological data.

According to Holmgren and Enquist [7], the model animals always attempt to distinguish themselves from mimic animals. By creating distance from the mimics, the predators experience less confusion between model and mimic animals. However, since the mimics follow the phenotype of the models and evolve faster, this is an endless cat and mouse game. In our simulation model, this can be observed in the pattern, where the pattern of the mimic animals closely follows the pattern of the model animals.

Our simulation model can be used to do more research on theories of mimicry. Since the parameters can be easily adjusted, more experiments can be done. Firstly, more experiments can be done with different starting positions of the camouflage, toxicity and pattern values, starting with a control run. In this case, one population without genetic drift would be created to see how preys evolve without other animals. Another example is the toxicity and mimic group starting in the toxic position, and the camouflage on the camouflage position. This would make for another coincidence set-up, which assumes the speciation of the mimicry being a sub-population from the toxic population. Alternatively, the pattern can be altered. The pattern of the camouflage and mimic start the same in this paper, but it can be altered to a situation where the toxic and camouflage population start the same, and see whether the toxic population moves away. Besides this we can see whether the mimics move toward the models, which supports the Coincidence hypothesis.

The dynamics of the model can be altered as well. One possibility is adding more dimensions of recognition. This would mean that instead of 1 pattern, the model would have 50 patterns, which all can be mutated and inherited individually. In these recognition dimensions the scale between 0 and 100 can be removed, so the models and mimics can move through the adaptive space with more freedom. This way, neophobia and the idea of Holmgren and Enquist [7] can be researched in more detail. When the dimensions are implemented we hypothesize that the models will keep evading the phenotype of the mimics and the mimics chasing this phenotype. If the domains are removed, we expect very high and low values in the dimensions, arguing for the very bright colors of the animals.

In our model, we assume that prey consists of three populations of constant size that cannot interbreed. In future work, it would be interesting to see how removing this assumption influences our results. This would create hybrid populations of prey, which may have interesting properties.

To research Müllerian mimicry, more populations can be added which have intermediate values of genetic drift toward toxic. With two toxic populations, a research can be conducted whether the animals imitate each others phenotype or keep their own phenotype. The number dependent theory [11] can be tested in the same way. We hypothesize that when more toxic populations are implemented, there will be one center where all the animals converge to, to make one clear

aposematism. The representation of knowledge can be differentiated. At the moment, the predators have a line in their choice to eat camouflage or not. If a more curved line, or other methods are implemented, the idea of *Novelty and Recognizability* [10] can be researched in more depth. If this is combined with a variation of punishment for toxicity, we expect that neophobia emerge from the simulation.

Lastly, a spatial model can be created, in which agents have a x- and y-coordinate. This way mimicry rings can be researched, which are discussed in great depth by Holmgren and Enquist [10], and found by Bates [1]. When the spatial model is implemented, all the aforementioned can be combined in one simulation, since every place can evolve something else. Especially the borders of different mimicry systems will be interesting to research. Using a bigger adaptation space, better knowledge of the predators and a spatial dimension in the model, we aim to have a better understanding of the origin of mimicry in the future.

References

1. Bates, H.W.: XXXII. Contributions to an insect fauna of the Amazon valley. Lepid. Heliconidae Trans. Linn. Soc. Lond. **23**(3), 495–566 (1862)
2. Boyd, R., Gintis, H., Bowles, S., Richerson, P.J.: The evolution of altruistic punishment. Proc. Natl. Acad. Sci. **100**(6), 3531–3535 (2003)
3. Cangelosi, A., Parisi, D.: Simulating the Evolution of Language. Springer, London (2012)
4. Epstein, J.M.: Agent-based computational models and generative social science. Complexity **4**(5), 41–60 (1999)
5. Epstein, J.M.: Generative Social Science: Studies in Agent-Based Computational Modeling. Princeton University Press, Princeton (2006)
6. Gavrilets, S., Hastings, A.: Coevolutionary chase in two-species systems with applications to mimicry. J. Theor. Biol. **191**(4), 415–427 (1998)
7. Holmgren, N.M., Enquist, M.: Dynamics of mimicry evolution. Biol. J. Linn. Soc. **66**(2), 145–158 (1999)
8. Jager, W., Popping, R., van de Sande, H.: Clustering and fighting in two-party crowds: simulating the approach-avoidance conflict. J. Artif. Soc. Soc. Simul. **4**(3), 1–18 (2001)
9. Maan, M.E., Cummings, M.E.: Poison frog colors are honest signals of toxicity, particularly for bird predators. Am. Nat. **179**(1), E1–E14 (2011)
10. Mallet, J., Joron, M.: Evolution of diversity in warning color and mimicry: polymorphisms, shifting balance, and speciation. Annu. Rev. Ecol. Syst. **30**(1), 201–233 (1999)
11. Müller, F.: Ituna and Thyridia: a remarkable case of mimicry in butterflies. Trans. Entomol. Soc. Lond. **1879**, 20–29 (1879)
12. Rabosky, A.R.D., Cox, C.L., Rabosky, D.L., Title, P.O., Holmes, I.A., Feldman, A., McGuire, J.A.: Coral snakes predict the evolution of mimicry across new world snakes. Nat. Commun. **7**, 11484 (2016)
13. de Weerd, H., Verbrugge, R.: Evolution of altruistic punishment in heterogeneous populations. J. Theor. Biol. **290**, 88–103 (2011)

Feature Selection in High-Dimensional Dataset Using MapReduce

Claudio Reggiani(✉), Yann-Aël Le Borgne, and Gianluca Bontempi

Machine Learning Group, Faculty of Science, Université Libre de Bruxelles, Boulevard du Triomphe, CP 212, 1050 Brussels, Belgium
{creggian,yleborgn,gbonte}@ulb.ac.be

Abstract. This paper describes a distributed MapReduce implementation of the minimum Redundancy Maximum Relevance algorithm, a popular feature selection method in bioinformatics and network inference problems. The proposed approach handles both *tall/narrow* and *wide/short* datasets. We further provide an open source implementation based on Hadoop/Spark, and illustrate its scalability on datasets involving millions of observations or features.

1 Introduction

The exponential growth of data generation, measurements and collection in scientific and engineering disciplines leads to the availability of huge and high-dimensional datasets, in domains as varied as text mining, social network, astronomy or bioinformatics to name a few. The only viable path to the analysis of such datasets is to rely on data-intensive distributed computing frameworks [1].

MapReduce has in the last decade established itself as a reference programming model for distributed computing. The model is articulated around two main classes of functions, *mappers* and *reducers*, which greatly decrease the complexity of a distributed program while allowing to express a wide range of computing tasks. MapReduce was popularised by Google research in 2008 [2], and may be executed on parallel computing platforms ranging from specialised hardware units such as parallel field programmable gate arrays (FPGAs) and graphics processing units, to large clusters of commodity machine using for example the Hadoop or Spark frameworks [2–4].

In particular, the expressiveness of the MapReduce programming model has led to the design of advanced distributed data processing libraries for machine learning and data mining, such as Hadoop Mahout and Spark MLlib. Many of the standard supervised and unsupervised learning techniques (linear and logistic regression, naive Bayes, SVM, random forest, PCA) are now available from these libraries [5–7].

Little attention has however yet been given to feature selection algorithms (FSA), which form an essential component of machine learning and data mining workflows. Besides reducing a dataset size, FSA also generally allow to improve

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 101–115, 2018.
https://doi.org/10.1007/978-3-319-76892-2_8

the performance of classification and regression models by selecting the most relevant features and reducing the noise in a dataset [8].

Three main classes of FSA can be distinguished: *filter*, *wrapper* and *embedded* methods [8,9]. Filter methods are model-agnostic, and rank features according to some metric of information conservation such as mutual information or variance. Wrapper methods use the model as a black-box to select the most relevant features. Finally, in embedded methods, feature evaluation is performed alongside the model training. In this paper, feature metrics are named hereafter *feature score* functions.

Early research on distributing FSA mostly concerned wrapper methods, in which parallel processing was used to simultaneously assess different subsets of variables [10–14]. These approaches effectively speed up the search for relevant subsets of variables, but require the dataset to be sent to each computing unit, and therefore do not scale as the dataset size increases.

MapReduce based approaches, which address this scalability issue by splitting datasets in chunks, have more recently been proposed [15–21]. In [15], an embedded approach is proposed for logistic regression. Scalability in the dataset size is obtained by relying on an approximation of the logistic regression model performance on subsets of the training set. In [16], a wrapper method based on an evolutionary algorithm is used to drive the feature search. The first approaches based on filter methods were proposed in [17,18], using variance preservation and mutual information as feature selection metrics, respectively. Two other implementations of filter-based methods have lately been proposed, addressing the column subset selection problem (CSSP) [19], and the distribution of data by features in [20]. Recently, a filter-based feature selection framework based on information theory [22] has been implemented using Apache Spark [21].

In this paper we tackle the implementation of *minimal Redundancy Maximal Relevance* (*mRMR*) [23], a forward feature selection algorithm belonging to filter methods. mRMR was shown to be particularly effective in the context of network inference problems, where relevant features have to be selected out of thousands of other noisy features [1,24].

The main contributions of the paper are the following: (*i*) design of minimum Redundancy Maximum Relevance algorithm using MapReduce paradigm; (*ii*) open-source implementation for Apache Spark available on a public repository; (*iii*) analysis of the scalability properties of the algorithm. In an extended version [25], we also detail how to customize the feature score function during the feature selection process.

The paper is structured as follows. Section 2 provides an overview of the MapReduce paradigm, and Sect. 3 describes the two main layouts along which data can be stored. Section 4 presents our distributed implementation of mRMR. Section 5 finally provides a thorough experimental evaluation, where we illustrate the scalability of the proposed implementation by varying the number of rows and columns of the datasets, the number of selected features in the feature selection step and the number of nodes in the cluster.

2 MapReduce Paradigm

MapReduce [2] is a programming paradigm designed to analyse large volumes of data in a parallel fashion. Its goal is to process data in a scalable way, and to seamlessly adapt to the available computational resources.

A MapReduce job transforms lists of input data elements into lists of output data elements. This process happens twice in a program, once for the *Map* step and once for the *Reduce* step. Those two steps are executed sequentially, and the Reduce step begins once the Map step is completed.

In the Map step, the data elements are provided as a list of key-value objects. Each element of that list is loaded, one at a time, into a function called *mapper*. The mapper transforms the input, and outputs any number of intermediate key-value objects. The original data is not modified, and the mapper output is a list of new objects.

In the Reduce step, intermediate objects that share the same key are grouped together by a *shuffling* process, and form the input to a function called *reducer*. The reducer is invoked as many times as there are keys, and its value is an iterator over the related grouped intermediate values.

Mappers and reducers run on some or all of the nodes in the cluster in an isolated environment, i.e. each function is not aware of the other ones and their task is equivalent in every node. Each mapper loads the set of files local to that machine and processes it. This design choice allows the framework to scale without any constraints about the number of nodes in the cluster. An overview of the MapReduce paradigm is reported in Fig. 1a.

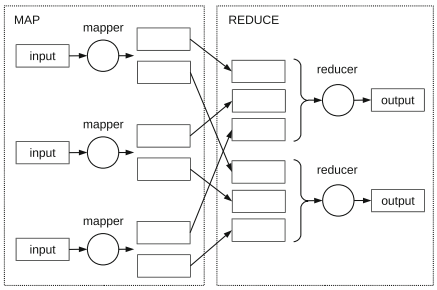
Algorithms written in MapReduce scale with the cluster size, and Execution Time (ET) can be decreased by increasing the number of nodes. The design of the algorithm and the data layout are important factors impacting ET [26].

In ET terms, jobs perform better in MapReduce when transformations are executed locally during the Map step, and when the amount of information transferred during the shuffling step is minimised [27]. In particular, MapReduce is very well-suited for associative and commutative operators, such as *sum* and *multiplication*. These can indeed be partially processed using an intermediate *Combine* step, which can be applied between the Map and Reduce stages.

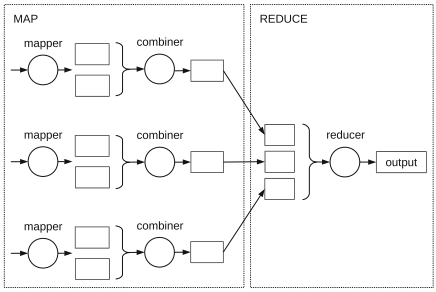
The combiner is an optional functionality in MapReduce [2]. It locally aggregates mapper output objects before they are sent over the network. It operates by taking as input the intermediate key-value objects produced by the mappers, and output key-value pairs for the Reduce step. This optional process allows to reduce data transfer over the network, therefore reducing the global ET of the job. An illustration of the use and advantages of the combiner is given in Fig. 1b.

3 Data Layout

In learning problems, training data from a phenomenon is usually encoded in tables, using rows as observations, and columns as features. Let M be the number



(a) MapReduce overview of the data flow. The dataset stored in the distributed storage system is split into chunks across nodes (rectangular *input* boxes). Each chunk is fed as input to the mapper functions, which may output intermediate objects. These objects are shuffled and grouped by keys across the network. Finally, the reducers generate the groups of intermediate objects and output the results. All objects (input, intermediate, output) are identified by a key-value pair.



(b) MapReduce overview of the data flow with the additional combiner function. Intermediate objects produced by mappers are locally aggregated by a commutative and associative function implemented in the combiner logic. In this example three, instead of six, intermediate objects are sent over the network to the reducer function. Combiners provide an efficient way to reduce the amount of shuffled data, and to reduce the overall execution time of the job.

Fig. 1. MapReduce data flow overview with and without combiner.

of observations, and N be the number of features. Training data can be represented as a collection of vectors, $\mathbf{X}$,

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M)$$

where

$$\mathbf{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,N}) \quad \forall j \in (1, \ldots, M)\,.$$

We will refer to this type of structure as the *conventional* encoding, see Table 1.

It is however worth distinguishing two types of tables: *tall and narrow* (T/N) tables, where $M \gg N$, and *short and wide* (S/W) tables, where $M \ll N$.

The distinction is important since MapReduce divides input data in chunks of rows, that are subsequently processed by the mappers. MapReduce is therefore well suited to ingest T/N table, but much less S/W tables, since data cannot be efficiently split along columns. S/W tables are for example encountered in domains such as text mining or bioinformatics [28,29], where the number of features can be on the order of tens or hundreds of thousands, while observations may only be on the order of hundreds or thousands.

In such cases, it can be beneficial to transform S/W into T/N tables, by storing observations as columns and features as rows. We refer to this type of structure as *alternative* encoding, see Table 2.

Table 1. Conventional encoding: Observations ($x_{i,.}$) are stored along rows, and features ($x_{.,j}$) are stored along columns.

$x_{1,1}$	$x_{1,2}$	...	...	$x_{1,N}$
$x_{2,1}$	$x_{2,2}$	...	...	$x_{2,N}$
...	...	...	...	...
...	...	...	...	...
$x_{M,1}$	$x_{M,2}$	...	...	$x_{M,N}$

Table 2. Alternative encoding: Observation ($x_{i,.}$) are stored along columns, and features ($x_{.,j}$) are stored along rows.

$x_{1,1}$	$x_{2,1}$	...	...	$x_{M,1}$
$x_{1,2}$	$x_{2,2}$	...	...	$x_{M,2}$
...	...	...	...	...
...	...	...	...	...
$x_{1,N}$	$x_{2,N}$	...	...	$x_{M,N}$

4 Iterative Feature Selection Framework

This section first recalls the standard mRMR algorithm [23]. We then detail our MapReduce implementation, for both conventional and alternative encodings. An implementation of a custom feature score function using the Pearson correlation coefficient is reported in the extended article [25].

4.1 minimal Redundancy Maximal Relevance

Let us define the dataset as the table $\mathbf{X}$ with M rows, N columns and discrete values. We define $\mathbf{x}_k$ as the k-th column vector of the dataset and $\mathbf{c}$ as the class vector. Furthermore, let us define L as the number of features to select and i_c^l and i_s^l as the sets at step l ($1 \leqslant l \leqslant L$) of candidate and selected features indices, respectively. At $l = 1$, we have $i_c^1 = \{1, ..., N\}$ and $i_s^1 = \varnothing$. The pseudo-code of the algorithm is reported in Listing 1.1.

Listing 1.1. minimum Redundancy Maximum Relevance Pseudo-code. $I(\cdot)$ is the function that, given two vectors, returns their mutual information. $\mathbf{x}_k$ is the k-th column vector of the dataset and $\mathbf{c}$ is the class vector. L is the number of features to select, i_c^l and i_s^l as the sets at step l ($1 \leqslant l \leqslant L$) of candidate and selected features indices.

$i_c^1 = \{1, ..., N\}$
$i_s^1 = \varnothing$
for $l = 1 \rightarrow L$
  for $k \in i_c^l$
    $I_{\mathbf{x}_k,\mathbf{c}} \leftarrow I(\mathbf{x}_k, \mathbf{c})$
    for $j \in i_s^l$
      $I_{\mathbf{x}_k,\mathbf{x}_j} \leftarrow I(\mathbf{x}_k, \mathbf{x}_j)$
    $g_k \leftarrow I_{\mathbf{x}_k,\mathbf{c}} - \frac{1}{|i_s^l|} \sum_{j \in i_s^l} I_{\mathbf{x}_k,\mathbf{x}_j}$
  $k^* \leftarrow$ `argmax(`g_k`)`
  $i_c^{l+1} \leftarrow i_c^l \backslash k^*$
  $i_s^{l+1} \leftarrow i_s^l \cup k^*$
`output` i_s^L

mRMR is an iterative greedy algorithm: at each step the candidate feature is selected based on a combination of its mutual information with the class and the selected features:

$$\underset{k \in i_c^l}{\operatorname{argmax}} g_k(\cdot)$$
$$g_k(\cdot) = \begin{cases} I(\mathbf{x}_k; \mathbf{c}) & l = 1 \\ I(\mathbf{x}_k; \mathbf{c}) - \frac{1}{|i_s^l|} \sum_{j \in i_s^l} I(\mathbf{x}_k; \mathbf{x}_j) & l > 1 \end{cases} \quad (1)$$

where $I(\cdot)$ returns the mutual information of two vectors. The *feature score* $g(\cdot)$ is assessed in Lines 5–8 in Listing 1.1.

We redesigned the algorithm using MapReduce paradigm on Apache Spark, distributing the feature evaluation into the cluster. Besides the core features of MapReduce previously described, our design takes advantage of the *broadcast* operator provided in Apache Spark. Broadcasted variables are commonly used in machine learning algorithms to efficiently send additional data to every mapper and reducer as read-only variables [30].

4.2 mRMR in MapReduce with Conventional Encoding

Let us define the dataset as a Resilient Distributed Dataset (RDD) [31] of M tuples $(\mathbf{x}, c)$, where $\mathbf{x}$ is the input (observation) vector and c is the target class value.

Considering the dataset with only discrete values, we represent with d_c the set of categorical values of the class, and with d_v the (union) set of unique categorical values of all features. If the dataset has binary values, then $d_c = d_v = \{0, 1\}$. In case of having features with different sets of categorical values, then d_v is the union of unique categorical values of all features.

The input vector is partitioned in candidate and selected features, labeled respectively as $\mathbf{x}_c$ and $\mathbf{x}_s$ ($\mathbf{x}_c \cup \mathbf{x}_s = \mathbf{x}$, $|\mathbf{x}| = N$). Variables L, i_c^l and i_s^l are defined as in the previous section and i_{class} is the class column index. Listings 1.2, 1.3 and 1.4 report the MapReduce job, the mapper and reducer functions, respectively, while an illustrative overview of the data flow is reported in Fig. 2.

Listing 1.2. mRMR MapReduce job with conventional data encoding. L is the number of features to select, i_c^l and i_s^l are the sets at step l ($1 \leqslant l \leqslant L$) of candidate and selected features indices. i_{class} is the class column index. d_c is the set of categorical values of the class, and d_v is the (union) set of unique categorical values of all features.

```
1  i_c^1 = {1, ..., N}
2  i_s^1 = ∅
3  for l = 1 → L
4    broadcast i_class, i_c^l, i_s^l, d_v, d_c
5    scores <- mapreduce(RDD, mapper, reducer)
6    k* ← collectArgmax(scores)
7    i_c^{l+1} ← i_c^l \ k*
8    i_s^{l+1} ← i_s^l ∪ k*
9  output i_s^L
```

Listing 1.3. mRMR MapReduce mapper function with conventional data encoding. i_c^l and i_s^l as the sets at step l $(1 \leqslant l \leqslant L)$ of candidate and selected features indices. i_{class} is the class column index. d_c is the set of categorical values of the class, and d_v is the (union) set of unique categorical values of all features. e is a single observation fed as input to the mapper, k and j represent column indices and *contTable* is the function that creates a contingency table.

```
# broadcasted vars: i_class, i_c^l, i_s^l, d_v, d_c
mapper(·, e)
  for k ∈ i_c^l
    output (k, contTable(e_k, e_{i_class}, d_v, d_c))
    for j ∈ i_s^l
      output (k, contTable(e_k, e_j, d_v, d_c))
```

Listing 1.4. mRMR MapReduce reducer function with conventional data encoding. k is a column index and t is a collection of contingency tables. The *score* function process all the contingency tables associated with the column with index k and return the feature score.

```
reducer(k, t)
  output(k, score(t))
```

For every $(e_k, e_{i_{class}})$ pair, the mapper task outputs a contingency table, *contTable*, with rows defined as the categorical values in d_c and columns defined as the categorical values in d_v. The element corresponding to row $e_{i_{class}}$ and column e_k is set to 1, while all the others are set to 0. Considering the dataset in Table 3, having one binary class column and four categorical features (with three possible values: -2, 0, 2), an example of emitted contingency table is reported in Table 4. In this example the class vector can only have two possible values: *0* and *1*; any feature can only have three possible values: -2, *0* and *2*. The input pair $(e_k, e_{i_{class}})$ is $(2, 0)$, therefore the element corresponding to row *0* and column *2* is set to 1, all the others are set to 0.

In case of (e_k, e_j) pair, the contingency table has both rows and columns defined by categorical values in d_v.

Table 3. Example of dataset encoded with conventional layout.

#entry	Class	Features			
	c	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$
1	0	2	0	0	-2
2	0	0	-2	2	0
3	0	0	2	0	-2
4	1	-2	0	0	0
...	...	...	...	...	...

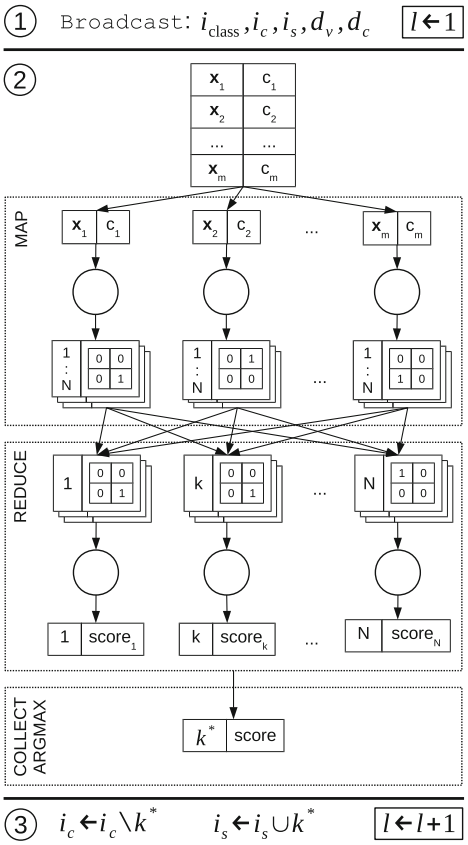


Fig. 2. Illustrative representation of the first iteration of a MapReduce job with discrete values using the conventional encoding. There are as many iterations as the number of features to select. At each iteration, each mapper outputs $N - l + 1$ contingency tables for every combination of candidate features and class vector, and, from the second iteration, $(N - l + 1) * \left|i_s^l\right|$ contingency tables for every combination of candidate and selected features.

At the cost of managing discrete values only, the commutative and associative properties of the contingency table allows the use of the combiner function, thus minimizing the amount of data exchanged across the cluster during shuffling. While the single mapper outputs one or more contingency tables for each candidate feature, those tables emitted by mappers executed on a given node can be locally reduced via the Combine step. Assuming that the first four entries in Table 3 are processed by four mappers in the same machine, Table 5 is the result of the combiner after the aggregation of four contingency tables of the $\mathbf{x}_1$ feature produced by the mappers. In this example, the combiner performs an element-wise sum of the contingency tables given as input.

Table 4. Contingency table emitted by the mapper function as a result of processing the pair $(\mathbf{x}_1, c)$ of the first entry in Table 3.

		d_v		
		−2	**0**	**2**
d_c	**0**	0	0	1
	1	0	0	0

Table 5. Aggregated contingency table emitted by the combiner function as a result of processing the pair $(\mathbf{x}_1, c)$ of the first four entries in Table 3.

		d_v		
		−2	**0**	**2**
d_c	**0**	0	2	1
	1	1	0	0

4.3 mRMR in MapReduce with Alternative Encoding

Data stored in alternative encoding has one column per observation and one row per feature. In this case, let us define the dataset as a RDD of N tuples $(k, \mathbf{x})$, where $\mathbf{x}$ is the feature vector and k is the row index ($k \in \{1, ..., N\}$). Feature and class values could be discrete and continuous as well. With respect to the design of mRMR in MapReduce with conventional encoding, a set of vectors are broadcasted across the cluster: $\mathbf{v}_{class}$ is the class vector, v_s is the collection of selected feature vectors and i_s is the collection of selected feature indices. Variable L is defined as in the previous section and *getEntry* function is a MapReduce task that retrieves the feature vector from the RDD, given a feature index. Listings 1.5 and 1.6 report the MapReduce job and the mapper function, respectively.

Listing 1.5. mRMR MapReduce job with alternative data encoding. *RDD* represents the distributed dataset and L is the number of features to select. $\mathbf{v}_{class}$ is the class vector, v_s is the collection of selected feature vectors and i_s is the collection of selected feature indices. The *getEntry* function retrieves the feature vector from the RDD, given a feature index.

```
1  i_s^1 = ∅
2  v_s^1 = ∅
3  for l = 1 → L
4    broadcast v_class, i_s^l, v_s^l
5    scores <- mapreduce(RDD, mapper)
6    k* ← collectArgmax(scores)
7    v* <- getEntry(RDD, k*)
8    i_s^{l+1} ← i_s^l ∪ k*
9    v_s^{l+1} ← v_s^l ∪ v*
10 output i_s^L
```

Listing 1.6. mRMR MapReduce mapper function with alternative data encoding. $\mathbf{v}_{class}$ is the class vector, v_s is the collection of selected feature vectors. The tuple $(k, \mathbf{x})$ is composed by the feature vector, $\mathbf{x}$, and the feature index, k. The *score* function processes the vectors and returns the feature score.

```
# broadcasted variables: v_class, v_s^l
mapper(·, (k, x))
  score <- score(x, v_class, v_s^l)
  output (k, score)
```

While in conventional encoding we used the contingency table as intermediate data structure, the design of mRMR in MapReduce with alternative encoding broadcasts at each iteration all required data for calculation to mappers. This design provides two main advantages: it deals with both discrete and continuous features as well, and the MapReduce job is composed by the Map step only. At the small cost of broadcasting some variables, all operations are executed locally. An illustrative overview of the data flow is reported in Fig. 3.

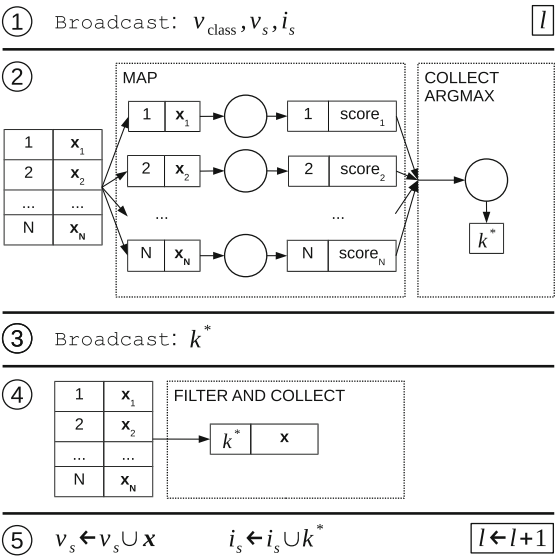


Fig. 3. Illustrative representation of a single iteration of a MapReduce job with alternative encoding. Steps 1–5 represent one iteration of the loop; there are as many iterations as the number of features to select.

5 Results

The source code of mRMR implementation in MapReduce with both encodings is available as a Scala library, along with examples, on a public repository (https://github.com/creggian/spark-ifs).

We studied the scalability of the implementation of mRMR in MapReduce in both encodings in a cluster with the following specifications: Hadoop cluster of 10 nodes, where each node has Dual Xeon e5 2.4 GHz processor, 24 cores, 128 GB RAM and 8 TB hard disk; all nodes are connected with a 1 Gb ethernet connection. Using Apache Spark v1.5.0, we submit jobs with 4 GB of RAM for both the driver and the executors.

For the evaluation of mRMR implementations we used binary artificial datasets. We followed the principles of CorrAL dataset [32], in which four features determine the class value with the following formula: $c = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$, one is irrelevant and the last one is partially correlated with the class. In all our datasets, the class value (c) depends on the value of 8 features (Formula 2); the remainings are irrelevant.

$$c = ((x_1 \wedge x_2) \vee (x_3 \wedge x_4)) \wedge ((x_5 \wedge x_6) \vee (x_7 \wedge x_8)) \tag{2}$$

We assessed the scalability on the number of rows, the number of columns, the number of selected features, and the number of nodes. We used two kinds of dependent variables: the relative execution time per executor and the computational gain. The former is the ratio between ET divided by ET of `1x`, the latter is the ratio between ET of 1-node and ET. We ran the tests three times to assess the variability of the results; in all figures the maximum, minimum and mean of these three values are connected through a solid vertical line.

5.1 Scalability Across the Number of Rows

We tested the scalability on the number of rows by means of four datasets, each with 1000 columns and an increasing number of rows: 1M, 4M, 7M and 10M (M = millions). We configured the cluster and the algorithm to select 10 features in a distributed environment of 10 nodes (Fig. 4a).

5.2 Scalability Across the Number of Columns

We assessed the scalability on the number of columns using four datasets, each with 1M rows and an increasing number of columns: 100, 400, 700 and 1000. We configured the cluster and the algorithm to select 10 features in a distributed environment of 10 nodes (Fig. 4b).

5.3 Scalability Across the Number of Selected Features

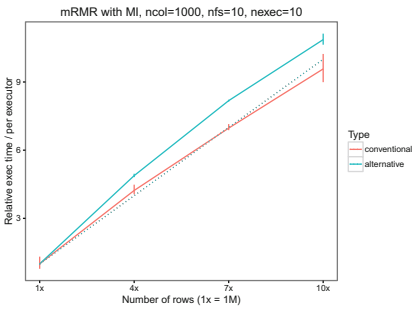
We investigated the scalability on the number of selected features using a dataset with 1M rows and 50k (k = thousands) columns. We parametrised the cluster to distribute the computation over 10 nodes, and the algorithm to select an increasing number of features: 1, 2, 4, 6, 10 (Fig. 4c).

5.4 Scalability Across the Number of Nodes

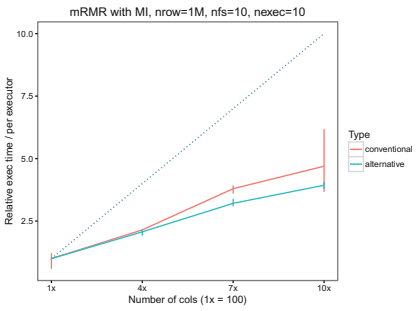
We tested the scalability across the number of nodes using a dataset with 1M rows and 100 columns. We configure the algorithm to select 10 features, and the cluster to distribute the work over 1, 2, 5 and 10 nodes (Fig. 4d).

By comparing the linear scalability (dotted line) with the actual performances, results show that the scalability of mRMR in MapReduce is linear with respect to the number of rows, as expected by MapReduce design; superlinear with respect to the number of columns; sublinear with respect to the number of selected features and nodes, as expected by our iterative algorithm design and the increasing amount of data exchanged in the network with the increasing of nodes, respectively.

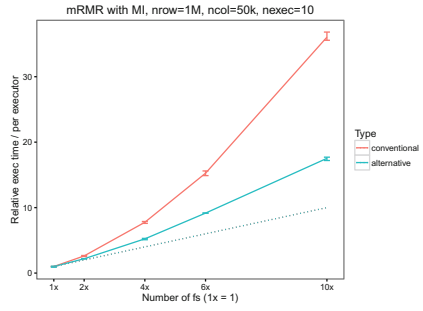
In studying mRMR with conventional and alternative layouts, we chose to use as independent variable the number of rows (columns) instead of the number of observation (features) for the following reason: while in the conventional layout we are able to scale across a very large number of rows, in the alternative layout



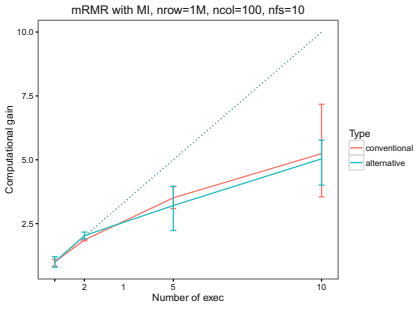
(a) mRMR scalability across the number of rows.



(b) mRMR scalability across the number of columns.



(c) mRMR scalability across the number of selected features.



(d) mRMR scalability across the number of nodes.

Fig. 4. Scalability performance of the mRMR distributed algorithm across number of rows, columns, selected features and nodes.

we are strictly constraint by the amount of memory available in the mapper task to scale across the number of columns. In Figs. 4a and b, we tested up to 10 million rows and up to one thousand columns, because very high-dimensional S/W tables raises memory errors in the cluster. Hence, even though we show the relative execution time, it would be incorrect to plot performances by increasing the number of observation (features).

The absolute execution time of mRMR MapReduce jobs with alternative encoding is generally 4–6 x faster than the respective jobs with conventional encoding.

6 Conclusion

In this work we investigated the design and scalability of mRMR algorithm in MapReduce. We proposed two implementations depending on the data layout, which can be easily interfaced in order to customize the feature score function. Despite Hadoop limitations for handling data with a large number of columns, the alternative data layout is a solution to store data from a phenomenon that has a very large number of features. In both conventional and alternative data layouts, we studied the scalability of mRMR in different settings: the number of rows, columns, selected features and nodes. Our experimental results illustrated the scalability of the proposed MapReduce implementations in a large variety of settings.

In the future, we intend to extend the approach with continuous features, and to provide an additional portfolio of built-in feature selection algorithms that work with the alternative encoding. While we design and implement known FSA for MapReduce, novel algorithms that directly take advantage of the distributed nature of the data will be investigated as well. We also plan to extend the scalability study to classification and network inference tasks.

Acknowledgement. The author CR acknowledges the funding of the BridgeIris project (RBC/13-PFS EH-11) supported by INNOVIRIS (Brussels Institute for the encouragement of scientific research and innovation) and The Belgian Kids' Fund. The authors YLB and GB acknowledge the funding of the Brufence project (Scalable machine learning for automating defense system) supported by INNOVIRIS (Brussels Institute for the encouragement of scientific research and innovation).

References

1. Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A.: Recent advances and emerging challenges of feature selection in the context of big data. Knowl. Based Syst. **86**, 33–45 (2015)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
3. Yeung, J.H., Tsang, C., Tsoi, K.H., Kwan, B.S., Cheung, C.C., Chan, A.P., Leong, P.H.: Map-reduce as a programming model for custom computing machines. In: 16th International Symposium on Field-Programmable Custom Computing Machines (FCCM 2008), pp. 149–159. IEEE (2008)

4. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. Proc. VLDB Endow. **2**(2), 1626–1629 (2009)
5. Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. Adv. Neural Inf. Process. Syst. **19**, 281 (2007)
6. Apache Mahout: Scalable machine learning and data mining. https://mahout.apache.org/
7. Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: MLlib: machine learning in apache spark. JMLR **17**(34), 1–7 (2016)
8. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
9. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)
10. López, F.G., Torres, M.G., Batista, B.M., Pérez, J.A.M., Moreno-Vega, J.M.: Solving feature subset selection problem by a parallel scatter search. Eur. J. Oper. Res. **169**(2), 477–489 (2006)
11. Melab, N., Cahon, S., Talbi, E.G.: Grid computing for parallel bioinspired algorithms. J. Parallel Distrib. Comput. **66**(8), 1052–1061 (2006)
12. de Souza, J.T., Matwin, S., Japkowicz, N.: Parallelizing feature selection. Algorithmica **45**(3), 433–456 (2006)
13. Garcia, D.J., Hall, L.O., Goldgof, D.B., Kramer, K.: A parallel feature selection algorithm from random subsets. In: Proceedings of the 17th European Conference on Machine Learning and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin (2006)
14. Guillén, A., Sorjamaa, A., Miche, Y., Lendasse, A., Rojas, I.: Efficient parallel feature selection for steganography problems. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5517, pp. 1224–1231. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02478-8_153
15. Singh, S., Kubica, J., Larsen, S., Sorokina, D.: Parallel large scale feature selection for logistic regression. In: SDM, pp. 1172–1183. SIAM (2009)
16. Peralta, D., Río, S., Ramírez, S., Triguero, I., Benítez, J.M., Herrera, F.: Evolutionary feature selection for big data classification: a MapReduce approach. In: Mathematical Problems in Engineering (2015)
17. Zhao, Z., Zhang, R., Cox, J., Duling, D., Sarle, W.: Massively parallel feature selection: an approach based on variance preservation. Mach. Learn. **92**(1), 195–220 (2013)
18. Sun, Z.: Parallel feature selection based on MapReduce. In: Wong, W.E., Zhu, T. (eds.) Computer Engineering and Networking. LNEE, vol. 277, pp. 299–306. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-01766-2_35
19. Ordozgoiti, B., Gómez Canaval, S., Mozo, A.: Massively parallel unsupervised feature selection on spark. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. CCIS, vol. 539, pp. 186–196. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23201-0_21
20. Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A.: Distributed feature selection: an application to microarray data classification. Appl. Soft Comput. J. **30**, 136–150 (2015)

21. Ramírez-Gallego, S., Mouriño-Talín, H., Martínez-Rego, D., Bolón-Canedo, V., Benítez, J.M., Alonso-Betanzos, A., Herrera, F.: An information theory-based feature selection framework for big data under apache spark. IEEE Trans. Syst. Man Cybern. Syst. **PP**(99), 1–13 (2017)
22. Brown, G., Pocock, A., Ming-Jie, Z., Luján, M.: Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. J. Mach. Learn. Res. **13**, 27–66 (2012)
23. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
24. Meyer, P.E., Lafitte, F., Bontempi, G.: minet: a R/bioconductor package for inferring large transcriptional networks using mutual information. BMC Bioinform. **9**, 461 (2008)
25. Reggiani, C., Le Borgne, Y.A., Bontempi, G.: Feature selection in high-dimensional dataset using MapReduce. ArXiv e-prints, September 2017
26. Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A comparison of join algorithms for log processing in MapReduce. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD 2010), New York, pp. 975–986. ACM (2010)
27. Sarma, A.D., Afrati, F.N., Salihoglu, S., Ullman, J.D.: Upper and lower bounds on the cost of a map-reduce computation. In: Proceedings of the VLDB Endowment, vol. 6, pp. 277–288. VLDB Endowment (2013)
28. Ahn, J., Jeon, Y.: Sparse HDLSS discrimination with constrained data piling. Comput. Stat. Data Anal. **90**, 74–83 (2015)
29. Jay, N.D., Papillon-Cavanagh, S., Olsen, C., Hachem, N., Bontempi, G., Haibe-Kains, B.: mRMRe: an R package for parallelized mRMR ensemble feature selection. Bioinformatics **29**(18), 2365–2368 (2013)
30. Karau, H., Konwinski, A., Wendell, P., Zaharia, M.: Learning Spark: Lightning-Fast Big Data Analytics, 1st edn. O'Reilly Media Inc., Sebastopol (2015)
31. 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 (NSDI 2012), Berkeley, p. 2. USENIX Association (2012)
32. Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A.: Feature Selection for High-Dimensional Data. AIFTA. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21858-8

Simultaneous Ensemble Generation and Hyperparameter Optimization for Regression

David Roschewitz(✉), Kurt Driessens(✉), and Pieter Collins(✉)

Maastricht University, P.O. Box 616, 6200 MD Maastricht, The Netherlands
d.roschewitz@student.maastrichtuniversity.nl,
{kurt.driessens,pieter.collins}@maastrichtuniversity.nl

Abstract. The development of advanced hyperparameter optimization algorithms, using e.g. Bayesian optimization, has encouraged a departure from hand-tuning. Primarily, this trend is observed for classification tasks while regression has received less attention. In this paper, we devise a method for simultaneously tuning hyperparameters and generating an ensemble, by explicitly optimizing parameters in an ensemble context. Techniques traditionally used for classification are adapted to suit regression problems and we investigate the use of more robust loss functions. Furthermore, we propose methods for dynamically establishing the size of an ensemble and for weighting the individual models. The performance is evaluated using three base-learners and 16 datasets. We show that our algorithms consistently outperform single optimized models and can outperform or match the performance of state of the art ensemble generation techniques.

Keywords: Bayesian optimization · Hyperparameter optimization
Ensemble generation · Regression

1 Introduction

Hyperparameter tuning *for regression* is sparsely covered in research and its combination with ensemble generation appears to be entirely absent. This although both techniques have been successfully applied to classification problems. Our research addresses this omission by examining methods to automatically generate ensembles with tuned hyperparameters for regression problems.

Naïve search methods have commonly been used for hyperparameter optimization of machine learning models. Grid search, for instance, evaluates hyperparameters on a grid with a predefined resolution. This, and its inefficiency in high-dimensional space, limits the usefulness for practical applications. Random search [2] partially alleviates this limitation, but unlike Bayesian optimization, it does not leverage all available information in the tuning process [22]. Researchers have expressed a need to explore both regression problems and deep learning in the context of hyperparameter optimization to encourage the departure from hand-tuning [3,9].

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 116–130, 2018.
https://doi.org/10.1007/978-3-319-76892-2_9

It is well accepted that ensemble methods, which consist of a combination of multiple base-learners, generally outperform single models for many types of problems [19]. Therefore, combining ensemble generation with hyperparameter tuning is a natural step for self-optimizing algorithms. Overproduce-and-select (OPAS) methods are a popular choice for ensemble construction, and can achieve best-in-class performance [6]. During an OPAS procedure, the ensemble is constructed selecting learned models from a library of predictors, which could be the models evaluated during a hyperparameter optimization procedure.

Lévesque et al. [15] devised a method that optimizes hyperparameters and simultaneously constructs the ensemble for classification problems. Essentially, it uses all previously trained models in the context of the ensemble for the Bayesian optimization step, which computes the hyperparameters of good base-models to add to the ensemble. They demonstrate that this way of generating ensembles can outperform OPAS methods, with no significant increase in runtime, and the benefit of not requiring a library of trained models.

In their investigation, Lévesque et al. showed that the optimization and ensemble generation step can be effectively combined. Their research, however, does not analyze the suitability of the method for regression problems, which require different treatment. Furthermore, critical elements such as ensemble size and the combination function of the ensemble were fixed. This paper will discuss and develop algorithms based on the research of Lévesque et al. [15] and releases the constraint of a fixed ensemble size. 16 small and medium sized publicly available data sets are used to evaluate the proposed solutions.

The paper is structured as follows: Sect. 2 formally introduces Bayesian hyperparameter optimization and ensemble generation approaches. Section 3 outlines our contributions, specifically modifications and extensions made to the algorithm. Section 4 presents the experimental set-up, results and analysis.

2 Hyperparameter Optimization

Hyperparameter optimization minimizes a function $f(\gamma)$, where γ is a set of hyperparameters in Γ, the hyperparameter space. In this context, $f(\gamma)$ can be evaluated by training a model with parameters γ, $M(\gamma)$, and computing its performance. This can be measured using a so-called loss function L. Previous observations, $\mathcal{D}$, of the performance of parameters γ can then be used for various optimization algorithms.

The following sections explain how Bayesian optimization is used to optimize $f(\gamma)$ and how ensembles can be generated from a set of trained models. Lastly the combined optimization and ensemble generation procedure, on which this research is based, is presented.

2.1 Bayesian Optimization

In contrast to naïve search methods such as grid or random search, Bayesian optimization uses *all* previous observations $\mathcal{D}$ to create a probabilistic model,

sometimes called a *surrogate function*, $\hat{f}(\gamma)$ of the objective $f(\gamma)$. A so-called *acquisition function* then computes the next point in Γ to evaluate. The model is updated with the performance of this point, and the two steps are repeated.

The surrogate function is the *posterior* distribution over the space of functions, induced by the *prior* and our observations $\mathcal{D}$. The prior captures our beliefs about the space of possible objective functions. For a more complete review of BO see, e.g., Brochu et al. [4] or for a broader overview see, e.g., Shahriari et al. [21].

Gaussian Process Prior. Gaussian process (GP) priors are a common choice for Bayesian optimization, due to their flexibility and tractability. A GP is a distribution over functions of type $f : \Gamma \to \mathbb{R}$, and when combined with observations $\mathcal{D}$ induces a posterior over functions. GPs are defined fully by their mean function $m : \Gamma \to \mathbb{R}$ and covariance function $k : \Gamma \times \Gamma \to \mathbb{R}$.

For hyperparameter optimization, the use of an automatic relevance detection (ARD) Matérn 5/2 kernel is suggested in literature [22].

$$k_{\mathrm{M52}}(x, x') = \theta_0 \exp(-\sqrt{5r^2(x, x')}) \\ (1 + \sqrt{5r^2(x, x')} + \frac{5}{3} r^2(x, x')). \tag{1}$$

$$r^2(x, x') = \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2. \tag{2}$$

The use of ARD kernels typically results in a total of $D+3$ GP parameters[1], θ: kernel length-scales $\theta_{1:D}$, kernel amplitude θ_0, observation noise v and constant mean m [22].

Acquisition Function. By construction of the GP prior, the surrogate $\hat{f}(\gamma)$ has both a predictive mean function $\mu(\gamma)$ and predictive variance function $\sigma(\gamma)$. The acquisition function $a : \Gamma \to \mathbb{R}^+$ can then be used to determine the *utility* of any point in Γ. The next set of hyperparameters to evaluate are then simply computed as $\gamma_{n+1} = \mathrm{argmax}_\gamma \; a(\gamma)$. Snoek et al. suggest using the Expected Improvement (EI) acquisition function, which provides a tradeoff between exploitation and exploration [22]:

$$a_{EI}(\gamma) = \sigma(\gamma)(\alpha(\gamma)\Phi(\alpha(\gamma)) + \phi(\alpha(\gamma))) \tag{3}$$

$$\alpha(\gamma) = \frac{f(\gamma_{\mathrm{best}}) - \mu(\gamma)}{\sigma(\gamma)} \tag{4}$$

where $\Phi(x)$ is the CDF and $\phi(x)$ the PDF of the standard normal distribution.

Since the mean and variance functions are dependent on the GP parameters θ, they can be represented as $\mu(\gamma; \theta)$ and $\sigma(\gamma; \theta)$. For the fully Bayesian treatment

[1] We employ the term 'GP parameters' to emphasize the difference between these and the hyperparameters subject to optimization in this paper.

of the GP parameters a so-called integrated acquisition function will be used throughout this paper. It is computed through a Monte Carlo estimate, using slice sampling for efficient computation of the required samples. See [22] and their additional material for further details on how the acquisition function is estimated.

2.2 Ensemble Construction from Optimization Output

Post-hoc ensemble generation (PHEG) is a natural way of constructing an ensemble from a library of trained models. PHEG is the *selection* stage of an OPAS method. In the context of optimizing hyperparameters, the history of all trained models serves as input for the ensemble generation.

The PHEG procedure works well in practice using a greedy selection criteria, which can be based on a variety of performance measures such as e.g., mean-squared-error (MSE) [6]. The procedure works as follows:

1. Begin with an empty ensemble.
2. Select the model which maximizes the ensemble performance.
3. Repeat 2 for a given number of iterations, until all models are added or until a point of diminishing returns is reached.

Caruana et al. [6] suggest modifying the procedure to allow selection with replacement. A noteworthy advantage of PHEG procedures, is that it can utilize the same performance function as the final benchmark, irrespective of how the library of models was acquired. When comparing our proposed algorithms to PHEG, we employ selection with replacement.

2.3 Simultaneous Ensemble Generation and Optimization

Hyperparameter optimization and ensemble construction are typically treated as separate procedures, with the possibility of coupling the techniques. Lacoste et al. [14] propose a bootstrapped round-robin technique, where each model of a fixed ensemble is optimized independently. Fundamentally, their ensemble sequential model-based optimization (ESMBO) procedure allows for more computationally efficient model optimization, but uses no information about the ensemble performance in the optimization.

In 2016 Lévesque et al. [15] outlined a simultaneous ensemble generation and optimization approach (SEGO). SEGO considers the performance of the ensemble, E, at every iteration. The loss function, introduced earlier, can be reformulated to evaluate the ensemble, not a single model.

Optimizing the hyperparameters of every model in E would make the objective space excessively high dimensional for large ensemble sizes. An elegant solution used by SEGO is to let the objective function $f(\gamma|E)$ be the loss of the ensemble if a model m trained with parameters γ is placed at index j of E. Hence, the *value* of parameters γ is measured *given* the current ensemble E.

$$f(\gamma|E, j) = L(E[j] \leftarrow M(\gamma)) \tag{5}$$

Throughout the SEGO algorithm, all trained models and their hyperparameter are stored in a history H and P respectively. The round-robin procedure optimizes the model at index $j = i \bmod n$ at iteration i and for ensemble size n. The loss of replacing $E[j]$ with every model in H is computed as l, using Formula 5. The loss of each hyperparameter in P can now be represented by l, and the two lists serve as input to the Bayesian optimization step, which then constructs the surrogate model $\hat{f}(\gamma)$.

The consequence of this type of loss calculation, is that for every iteration the loss must be computed $|H|$ times, resulting in runtime of $O(|H|m)$ for m prediction instances. This overhead is tolerated as the GP computation of the surrogate model contains a matrix inversion, which runs in $O(|H|^3)$. In addition, the assumption that training a learning algorithm is significantly more costly holds in practice.

Algorithm 1 formalizes the SEGO algorithm, but is adjusted to follow the notation of this paper. Note the explicit notion of the cross-validated loss computation, where the loss is averaged. In our research, the loss is estimated through 5-fold cross-validation. We employ *Spearmint* provided by Snoek et al. [22][2] as the implementation of Bayesian optimization for hyperparameter tuning. Constant Gaussian noise is assumed.

Algorithm 1. Simultaneous Ensemble Generation and Optimization

Input : b, n, $a()$, $L()$, $M()$
Output: Ensemble E; history of models H

$E, H, P \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** b **do**
  $j \leftarrow i \bmod n$;
  $l \leftarrow \text{cross-val}([L(E[j] \leftarrow m)]_{m \subset H})$;
  $\hat{f}(\gamma) \leftarrow \text{BO}(P, l)$;
  $\gamma_i \leftarrow \text{argmax}\ a_{EI}(\gamma; \hat{f}(\gamma))$;
  $m_i \leftarrow M(\gamma_i)$;
  $l \leftarrow l \cup \{\text{cross-val}(L(E[j] \leftarrow m_i))\}$;
  $H \leftarrow H \cup \{m_i\}$;
  $P \leftarrow P \cup \{\gamma_i\}$;
  $E[j] \leftarrow H[\text{argmin}_k l[k]]$;
end

The research performed by Lévesque et al. showed promising results: Their algorithm outperforms a single optimized model. In some cases SEGO also performed better than an ensemble constructed post-hoc from all models trained throughout the procedure [15]. If a loss function different from the final ensemble evaluation is used for the optimization, they suggest using the PHEG method on the history H from the SEGO procedure.

[2] Code available at https://github.com/JasperSnoek/spearmint.

SEGO was designed and tested for classification tasks only, an omission our research addresses. The authors also note that the requirement of a fixed ensemble size n should be investigated. Furthermore, we find that different loss functions as well as ensemble weighting functions should be explored. This could both improve the algorithm and also demonstrate its robustness. In the following section we introduce SEGO for regression, suitable loss functions, dynamic sizing approaches and a non-constant weighting technique.

3 SEGO for Regression

In order to apply the SEGO procedure to regression problems, the ensemble *integration function* (sometimes referred to as weighting or combination function) and the loss function must be chosen. A default choice for ensembles, is to average the results of all models. The ensemble prediction is therefore a linear combination of all model predictions, with weights $w_{1:n} = \frac{1}{n}$ for ensemble size n.

The residuals of a predictor can be defined as $r_i = \hat{y}_i - y_i$. A typical loss function for regression problems is the mean-squared-error (MSE): $MSE = \frac{1}{n}\sum_{i=1}^{n}(r_i)^2$. With integration and loss function defined, the baseline SEGO for regression (SEGOR) is established.

3.1 Loss Function

The fashion with which the loss of parameters given an ensemble, $f(\gamma|E)$, is computed is instrumental to the iterative optimization of the ensemble. Firstly, the cross-validated loss is used directly to select the next hyperparameters γ_{i+1}, and the loss is also used to choose which model to place at index j.

A known drawback of MSE is that it places disproportionately high weights on large errors. In the context of SEGOR this means a model might be selected if it reduces the error on a difficult-to-predict outlier, but in fact reduces the performance on most unseen instances.

Regularization techniques aim to reduce overfitting, usually by complementing the minimization function with a *regularization term*. Since we are not performing a straight-forward minimization, but using an iterative procedure, adding such a term is non-trivial. However, there exist robust loss functions that aim to reduce overfitting.

One such function is the *Huber loss*, a type of robust M-estimator. In 1964 Huber introduced the idea of a generalized M-estimator minimizing $\sum_{i=1}^{n}\rho(x_i)$ where ρ is some function on data x [11]. Huber loss is defined as a piecewise function of residuals whose behaviour is governed by $|r|$. This significantly reduces the impact of large errors, reducing the likelihood to overfit.

$$L_{\text{huber}}(r) = \begin{cases} \frac{1}{2}r^2 & \text{if } |r| < c \\ c(|r| - \frac{1}{2}c) & \text{otherwise} \end{cases}$$

Another, more extreme, type of robust loss function is the *Tukey bisquare*. In contrast to Huber loss, for large $|r|$, the loss is constant. This further reduces the weight given to large errors. Tukey's bisquare has been used successfully as a loss function for other regression problems [1].

$$L_{\text{tukey}}(r) = \begin{cases} \frac{c^2}{6}[1-(1-(\frac{r}{c})^2)^3] & \text{if } |r| < c \\ \frac{c^2}{6} & \text{otherwise} \end{cases}$$

The chosen values of c for Huber loss and Tukey's bisquare are $c = 1.345$ and $c = 4.685$, respectively. At these values of c, both loss functions are at least 95% as accurate as least-squares if the data is sampled from a normal distribution. Figure 1 highlights the differences in the loss functions.

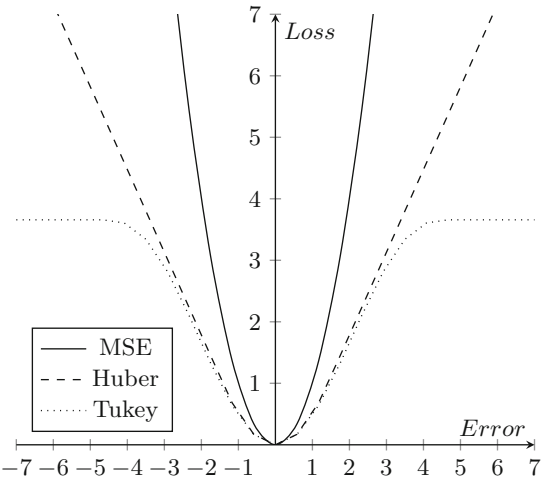


Fig. 1. Comparison of loss functions

3.2 Ensemble Size

Much of ensemble research has focused on selecting an ensemble from a library of models [5,12]. SEGOR currently optimizes parameters given a fixed ensemble size n. Such ensemble generating algorithms are sufficiently novel that there are no established approaches to dynamically adjust the ensemble size throughout the optimization procedure.

Lévesque et al. empirically selected $n = 12$ for their investigation, and our work confirms that this is a reasonable choice, but this might not hold for all types of base-models and datasets. We therefore devised multiple methods of dynamically generating the ensemble *throughout* the optimization process with *no* fixed size.

The fixed ensemble size method (F) will serve as the baseline. Note that the modifications to non-fixed ensemble sizes are performed *after* the BO step, and hence do not affect the input of the hyperparameter tuning. The first proposed method, *best-growing* (BG), initializes the ensemble as $E = \emptyset$, and can add any model $m \in H$, in addition to replacing the model at index j. If a model is added to the ensemble, the remainder of the ensemble is left unchanged. This means that at iteration i, i models are evaluated at index j, and i models are evaluated as additions. An additional set of losses, l_{grow} is computed:

$$l_{\text{grow}} = \text{cross-val}([L(E \cup m)]_{m \subset H}) \tag{6}$$

Similarly, the *best-dynamic* (BD) method allows for the deletion of a model in the ensemble in addition to growing:

$$l_{\text{shrink}} = \text{cross-val}(L(E[j] \leftarrow \emptyset)) \tag{7}$$

Each method then decides which action to perform: replace, grow (BG and BD only) or shrink (BD only).

3.3 Integration Function

In SEGO, as it was designed for classification, a majority voting ensemble *integration function* was used. As the ensemble prediction in SEGOR is already a linear combination of its component regressors, the weights $w_{1:n}$ for ensemble size n are suitable for optimization [20]. If we express the set of weights w_i for $i = 1 : n$ as w, then by finding $\text{argmin}_w L(\sum_{i=1}^{N}(r_i(x) * w_i))$ it is possible to find weights minimizing the validation loss, with an increased risk of overfitting the validation set, however. For the hyperparameter tuning step mean-weighting is used as otherwise the individual models' equal impact on the loss-evaluation is no longer guaranteed. We explore weighting using Python's SciPy implementation of BFGS [13].

4 Experiments

The objective of the experimentation is two-fold: Firstly, we want to demonstrate the applicability of SEGO to regression and secondly highlight the performance and robustness of our improvements. Three different types of predictors are used as base-models for hyperparameter tuning tested on 16 small to medium sized datasets. As each experiment requires training a significant number of models in addition to the optimization overhead, and computational resources were limited for this research, the scope of experimentation had to be restricted. Therefore every dataset was limited to 2999 instances. Each experiment is 5-fold cross validated and the number of Bayesian optimization iterations was set to 100 for every tested approach. The following shorthand will be used: *method - loss-function*, where s is a single model and f represents SEGOR with a fixed

ensemble size of 12; g and d are the BG and BD ensemble size methods from Sect. 3.2, respectively. The loss functions m, h and t correspond to MSE, Huber loss and Tukey's bisquare. Lastly, the suffix ph refers to the ensemble generated post-hoc using PHEG.

All regression models were chosen from the scikit-learn library [18]. The base-models and their hyperparameters are:

- DecisionTree (DT):
  max_depth, max_features, min_samples_split, min_samples_leaf
- MultiLayerPerceptron (MLP):
  learning_rate, activation, hidden_layer_sizes (max. 2 hidden layers)
- ElasticNet (EN):
  alpha, l1_ratio, max_iter

The datasets and the number of used features and instances can be viewed in Table 1. Unless cited otherwise, they were retrieved from the UCI Machine Learning Repository [16].

Table 1. Datasets

Dataset	Instances	Features
CPU (cpu)	209	6
Boston housing (bos)	506	13
White wine (ww)	2999	11
Red wine (rw)	1599	11
Chicago speed (csp) [24]	2999	3
MPG (mpg)	398	6
Power plant (pow)	2999	4
Solar flare (sf)	1066	23
Facebook comments (fp)	500	10
Air quality (aq)	2999	12
Concrete stength (cs)	1030	8
Cooling efficiency (ce)	768	8
Heating efficiency (he)	768	8
Math grades (mg)	395	56
Yacht resistance (yr)	308	6
Forest fire area (ffa)	517	22

A single predictor (s-m) optimized for 100 iterations and a post-hoc ensemble generated from its history (s-m-ph) will serve as a baseline. The performance is measured as the MSE of previously normalized values, irrespective of the loss function used. Furthermore, the diversity of the ensemble will be measured

by the mutual information (MI) of the individual models' predictions. In order to measure MI of an ensemble, the sum of all pairwise MI is taken, divided by the ensemble size squared [8]. Low values of MI represent ensembles with diverse predictions.

$$MI(Y^m; Y^n) = -\frac{1}{2} log(1 - \rho^2) \tag{8}$$

where Y^i is the prediction of model i, and ρ is the correlation between two predictors:

$$\rho = \frac{\sum_{i=1\ldots N}(y_i^m - \mu_Y^m)(y_i^n - \mu_Y^n)}{\sqrt{\sum_{i=1\ldots N}(y_i^m - \mu_Y^m)^2 (y_i^n - \mu_Y^n)^2}}. \tag{9}$$

4.1 Performance Results

In order to compare the different algorithms, their performance was measured for each of the base-models separately. We compare the combination of different sizing methods and loss functions.

For methods using loss functions other than MSE, a post-hoc ensemble is used to measure performance, as the PHEG procedure can utilize the same performance-function as the final benchmark. Therefore, post-hoc results are used where appropriate to ensure methods are not disadvantaged. An intuitive and insightful way to compare different algorithms is to observe their mean ranks, which are shown in Table 2.

Table 2. Mean ranks over 3 hyperparameter spaces

	s-m	f-m	g-m	d-m	s-m-ph	f-h-ph	g-h-ph	d-h-ph	f-t-ph	g-t-ph	d-t-ph
DT	10.19	4.12	4.56	5.50	6.19	5.69	5.88	7.00	4.81	5.44	6.62
MLP	9.38	7.25	4.81	3.88	5.56	6.31	6.50	5.69	5.88	4.69	6.06
EN	7.00	6.00	6.81	6.00	6.81	6.06	6.00	5.06	5.38	5.25	5.44

In order to measure performance differences we used two procedures. The first is a two-step process for comparing multiple methods simultaneously. First, a Friedman test is used to test whether there is a significant difference between all methods [7]. In principle, the test considers the mean rank, and is a non-parametric version of the well known ANOVA test. The p-values are 2.8×10^{-5}, 6.0×10^{-4} and 0.72 for DT, MLP and EN respectively. This suggests that for elastic nets, all methods perform similarly. Further investigation revealed that elastic nets performed worse in absolute terms compared to the other base types.

When the Friedman test shows significant differences, a post-hoc test is utilized to determine which methods differ from one another. The Nemenyi test can be used to compare all methods based on mean ranks [7], and includes a compensation for multiple comparisons. Our findings show that the Nemenyi test is more conservative, and is very sensitive to the methods selected for comparison.

For instance, *d-h-ph* no longer outperforms *s-m* at $p < 0.05$ for DT space. In the case of MLPs, *f-m*, *f-h-ph*, *g-h-ph*, *d-h-ph*, *f-t-ph* and *d-t-ph* are all above the critical p-value when compared to *s-m*. All performance differences are insignificant even at $p < 0.10$ for elastic nets.

Table 3 shows the results of the Nemenyi post-hoc test, comparing all proposed methods with the single optimized model *s-m*. All other comparisons values are insignificant at $p > 0.10$, and are not shown but were performed simultaneously.

Table 3. Nemenyi test p-values for both hyperparameter spaces

	f-m	g-m	d-m	s-m-ph	f-h-ph	g-h-ph	d-h-ph	f-t-ph	g-t-ph	d-t-ph
DT	0.00	0.00	0.00	0.03	0.00	0.00	0.13	0.00	0.00	0.04
MLP	0.77	0.00	0.00	0.05	0.24	0.33	0.06	0.10	0.00	0.15

The second procedure, the Wilcoxon signed-ranks test is a non-parametric paired test, which can compare the performance of two algorithms given multiple instances [7]. The methodology considers the absolute value of the performance difference, and is therefore more representative than mean ranks. Furthermore, it does not assume sampling from a normal distribution, as the paired t-test would.

In summary, all methods outperform the single optimized model, with $p < 0.05$ for both decision trees and multi-layer perceptrons. For elastic nets however, all methods show insignificant improvement at $p < 0.05$ in the pairwise comparison. Notable is the performance of *f-m* which outperforms *s-m-ph* at $p < 0.05$ for decision trees, therefore outperforming an ensemble generated using an OPAS method. Similarly *g-m* and *d-m* perform better than *f-m* for multi-layer perceptrons. This highlights the importance of non-fixed ensemble sizing for certain domains. Tables 4 and 5 show all pairwise p-values of the Wilcoxon tests, the structure of the tables are such that a value in row i and column j represents the p-value that algorithm i outperforms algorithm j.

We have excluded the weighted methods from the above comparisons, as we can observe weak performance, likely due to overfitting on the validation data or multi-collinearity among the predictors [17].

In conclusion, the proposed methods work well on the DT and MLP hyperparameter space, where the tunable parameters strongly affect the way the regressor learns. The sizing methods performed well, with *g-m* and *d-m* trading spots depending on the hyperparameter space. Robust loss functions did not improve generalization accuracy overall, and the Nemenyi test indicates they do not always perform significantly better than a single optimized regressor. We suspect this is related to the relatively small datasets. We believe given more diverse data or if the real loss function cannot be used for learning, robust loss functions would be more applicable.

Table 4. Pairwise Wilcoxon test p-values for decision trees

	s-m	f-m	g-m	d-m	s-m-ph	f-h-ph	g-h-ph	d-h-ph	f-t-ph	g-t-ph	d-t-ph
s-m		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
f-m	0.00		0.47	0.05	0.00	0.07	0.02	0.01	0.43	0.03	0.00
g-m	0.00	0.55		0.06	0.07	0.19	0.11	0.01	0.70	0.14	0.03
d-m	0.00	0.96	0.94		0.15	0.70	0.43	0.19	0.74	0.63	0.55
s-m-ph	0.00	1.00	0.94	0.86		0.72	0.70	0.23	0.83	0.57	0.74
f-h-ph	0.00	0.94	0.83	0.32	0.30		0.08	0.01	0.75	0.45	0.03
g-h-ph	0.00	0.99	0.90	0.59	0.32	0.93		0.05	0.86	0.74	0.49
d-h-ph	0.00	0.99	0.99	0.83	0.78	0.99	0.95		1.00	0.99	0.96
f-t-ph	0.00	0.59	0.32	0.28	0.19	0.26	0.15	0.00		0.47	0.01
g-t-ph	0.00	0.97	0.87	0.39	0.45	0.57	0.28	0.01	0.55		0.15
d-t-ph	0.00	1.00	0.97	0.47	0.28	0.97	0.53	0.05	0.99	0.86	

Table 5. Pairwise Wilcoxon test p-values for multi-layer-perceptrons

	s-m	f-m	g-m	d-m	s-m-ph	f-h-ph	g-h-ph	d-h-ph	f-t-ph	g-t-ph	d-t-ph
s-m		0.99	1.00	1.00	1.00	1.00	0.99	1.00	0.99	1.00	1.00
f-m	0.01		0.97	0.99	0.55	0.65	0.28	0.65	0.47	0.87	0.74
g-m	0.01	0.03		0.70	0.19	0.01	0.06	0.14	0.10	0.33	0.26
d-m	0.00	0.01	0.32		0.07	0.08	0.03	0.02	0.03	0.17	0.10
s-m-ph	0.00	0.47	0.83	0.94		0.53	0.45	0.78	0.61	0.89	0.88
f-h-ph	0.00	0.37	0.99	0.93	0.49		0.35	0.57	0.37	0.85	0.43
g-h-ph	0.01	0.74	0.95	0.98	0.57	0.67		0.28	0.53	0.94	0.55
d-h-ph	0.00	0.37	0.87	0.99	0.23	0.45	0.74		0.55	0.75	0.68
f-t-ph	0.01	0.55	0.91	0.98	0.41	0.65	0.49	0.47		0.65	0.41
g-t-ph	0.00	0.14	0.68	0.84	0.12	0.16	0.07	0.26	0.37		0.13
d-t-ph	0.00	0.28	0.75	0.91	0.13	0.59	0.47	0.33	0.61	0.88	

4.2 Diversity

We can utilize the MI diversity measure to explain the poor performance of the algorithms on the EN hyperparameter space. Table 6 highlights the much greater mean mutual information in the ensemble of elastic nets. This corresponds with the design of elastic nets, where the impact of different parameters has a limited effect on the prediction. Therefore, our hyperparameter tuning focus does not work well for such base-models, and the MI measure can be used for detecting this situation.

Table 6. Mean distance measures for the hyperparameter spaces

	dt	nn	en
Mean mut. info	1.35	1.42	4.28

As all of the aforementioned experiments were conducted with a budget of 100 iterations, the ensemble methods do not converge. We conducted an experiment using the 'rw' dataset for decision trees, with 300 iterations. The fixed-ensemble-size method *f-m* was tested for all sizes in the range $[4, 28]$, with sizes 25, 22 and 19 being the top performers. Our BG method *g-m* ranked 5th, with a final mean ensemble size of 27.76. Interestingly, a fixed size of 24 would have resulted in the 20th rank for *f-m*. This high sensitivity to the a-priori fixed ensemble size highlights the importance of the dynamic ensemble sizing methods. Due to computational limitations, we were unable to investigate true convergence.

5 Conclusion

We presented a method to simultaneously generate ensembles and tune the hyperparameters of its models for regression problems. Furthermore, we introduce robust loss functions and different methods of determining the size of the ensemble on-the-fly. For models with tunable hyperparameter spaces, our proposed techniques significantly outperform single regressors. The proposed sizing methods allow the algorithm to operate without a fixed a-priori ensemble size, a parameter which was shown to impact performance. The proposed robust loss functions have failed to exceed the performance of procedures using MSE, but tend to outperform single models. For models where hyperparameters only slightly affect the diversity in predictions, the suggested methods cannot significantly improve on a single tuned predictor.

Noteworthy is the finding that depending on the chosen base-learner and dataset, a different approach might be the most suitable, highlighted by differences found between decision trees and multi-layer-perceptrons. Most importantly, however, our research demonstrates the suitability of hyperparameter tuning to regression, and showcases the performance of automated meta-learning algorithms, specifically for ensemble generation with no fixed size.

Future Research

With more computational resources available, the proposed methods could be investigated on a greater breadth of larger datasets with more optimization iterations. The utilization of a diversity or complexity term during optimization has had mixed results in other applications [10], but could nonetheless be used to aid in dynamic ensemble sizing. For highly complex hyperparameter spaces, neural networks are suitable replacements for GPs [23]. Our research focused little on ensemble weighting, but other techniques such as stacking or regularized

weighting should not be discarded. Furthermore, the suitability of hyperparameter tuning and more advanced techniques such as SEGO(R) have not been extensively explored for deep neural networks, an area which we hope will see increased attention in the future.

Acknowledgements. We want to thank Mediaan for supporting this research and graciously providing compute resources.

References

1. Belagiannis, V., Rupprecht, C., Carneiro, G., Navab, N.: Robust optimization for deep regression. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2830–2838 (2015)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
3. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
4. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
5. Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: Sixth International Conference on Data Mining, ICDM 2006, pp. 828–833. IEEE (2006)
6. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 18. ACM (2004)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. Dutta, H.: Measuring diversity in regression ensembles. In: IICAI, vol. 9, 17p (2009)
9. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems, pp. 2962–2970 (2015)
10. Gu, S., Cheng, R., Jin, Y.: Multi-objective ensemble generation. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **5**(5), 234–245 (2015)
11. Huber, P.J., et al.: Robust estimation of a location parameter. Ann. Math. Stat. **35**(1), 73–101 (1964)
12. Johansson, U., Löfström, T., Boström, H.: Overproduce-and-select: the grim reality. In: 2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL), pp. 52–59. IEEE (2013)
13. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001). http://www.scipy.org/
14. Lacoste, A., Larochelle, H., Laviolette, F., Marchand, M.: Sequential model-based ensemble optimization. arXiv preprint arXiv:1402.0796 (2014)
15. Lévesque, J.C., Gagné, C., Sabourin, R.: Bayesian hyperparameter optimization for ensemble learning. In: Ihler, A., Janzing, D. (eds.) 2016 Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, pp. 437–446. AUAI Press, Arlington (2016)
16. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml

17. Mendes-Moreira, J., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: a survey. ACM Comput. Surv. (CSUR) **45**(1), 10 (2012)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
19. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
20. Seni, G., Elder, J.F.: Ensemble methods in data mining: improving accuracy through combining predictions. Synth. Lect. Data Min. Knowl. Discov. **2**(1), 1–126 (2010)
21. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: a review of Bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016)
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, pp. 2951–2959 (2012)
23. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., Adams, R.: Scalable Bayesian optimization using deep neural networks. In: International Conference on Machine Learning, pp. 2171–2180 (2015)
24. Speed camera violations, Chicago data portal. https://data.cityofchicago.org/Transportation/Speed-Camera-Violations/hhkd-xvj4/data

Comparison of Machine Learning Techniques for Multi-label Genre Classification

Mathijs Pieters and Marco Wiering(✉)

Institute of Artificial Intelligence and Cognitive Engineering,
University of Groningen, Groningen, The Netherlands
{m.t.pieters,m.a.wiering}@rug.nl

Abstract. We compare classic text classification techniques with more recent machine learning techniques and introduce a novel architecture that outperforms many state-of-the-art approaches. These techniques are evaluated on a new multi-label classification task, where the task is to predict the genre of a movie based on its subtitle. We show that pre-trained word embeddings contain 'universal' features by using the Semantic-Syntactic Word Relationship test. Furthermore, we explore the effectiveness of a convolutional neural network (CNN) that can extract local features, and a long short term memory network (LSTM) that can find time-dependent relationships. By combining a CNN with an LSTM we observe a strong performance improvement. The technique that performs best is a multi-layer perceptron, with as input the bag-of-words model.

Keywords: Natural language processing
Multi-label text classification · Movie subtitles · CNN model
LSTM network · Bag-of-words model

1 Introduction

Text classification is the task of assigning specific categories to documents, examples are spam detection and sentiment analysis. Naive Bayes, a technique based on applying Bayes' Theorem, is frequently used as a baseline method for text classification because it is relatively effective, fast, and easy to implement [11]. Numerous attempts have been made to tackle the poor assumptions of Naive Bayes [8,17].

Various types of neural networks have been developed throughout the years, many of these techniques are used for natural language processing (NLP) applications. A traditional method is the multilayer perceptron (MLP), trained on the bag-of-words (BoW) model [1]. The BoW model is a sparse representation of texts, ignoring both word order and semantic and syntactic features, treating texts as unordered sets of words. In order to capture the subtleties of language, we seek a dense representation that does capture these features. Many state-of-the-art word

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 131–144, 2018.
https://doi.org/10.1007/978-3-319-76892-2_10

embedding techniques [12,15] are based on the distributional hypothesis [3], stating that *linguistic items with similar distributions have similar meanings*. These dense representations capture multiple degrees of similarity [14], both semantic and syntactic, such that similar words have similar representations.

Convolutional neural networks (CNN) make use of the internal structure of the dense representation, both in the feature domain, and the temporal (word order) domain. CNN models have achieved remarkable results on various text classification tasks [5,21]. Whereas CNN models make use of the word order for a specific region size, recurrent neural networks (RNN) have the ability to capture long-term dependencies for texts of any length. More specifically, the Long Short-Term Memory (LSTM) architecture [4] is well suited for longer texts because of its ability to remember information for long periods of time.

In this paper, we introduce a novel dataset which we will use for multi-label text classification. We compare several state-of-the-art techniques, such as the concatenation-CNN and the LSTM network, with more traditional techniques. Furthermore, we introduce a novel architecture that applies a histogram on word embeddings, followed by an MLP. Unlike most research, we trained our own word embeddings, making our setup stand-alone.

In Sect. 2 we introduce our dataset, followed by Sect. 3 where we explain the used methods. The experimental setup is described in Sect. 4, and in Sect. 5 we show and discuss the results. We conclude the paper in Sect. 6 with a conclusion and a proposal for future work.

2 Dataset

The dataset used in the experiment is an intra-lingual movie subtitle corpus, collected by [9], and originates from OpenSubtitles[1]. We extracted the English corpus, and removed all tokens apart from the spoken text. Subsequently, we convert all words to lowercase and remove punctuation, see Fig. 1. We did not apply stop word removal or stemming. The total dataset consists of 44,171 subtitles, with in total 135,862,112 words and 920,705 unique words. Every subtitle is linked to at least one, and often multiple genres. In total the dataset contains subtitles with 27 different genres, ranging from animation and comedy, to documentary. Because every subtitle can have multiple genres, the classification task is considered a multi-label classification task. This should not be confused with multi-class classification, where every document has exactly one label. Multi-label classification is considered to be significantly more difficult, due to the vast amount of possible label combinations.

Because of the limited availability of computer power we will narrow our focus to the classification of the following genres: “Romance”, “Thriller”, and “Action”. This subset consists of 15,500 subtitles, with in total 48,998,774 words and 448,101 unique words. The distribution of the subtitle lengths is depicted in Fig. 2.

[1] http://www.opensubtitles.org/.

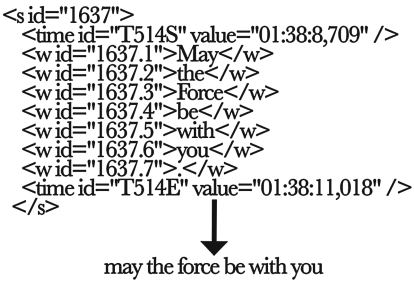


Fig. 1. Text preprocessing, single sentence example.

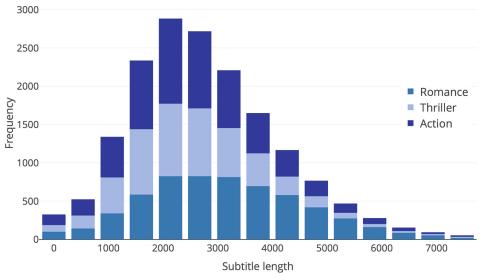


Fig. 2. Distribution of subtitle lengths

3 Methods

In this paper, we will differentiate between models that use the BoW model, and models that use word embeddings. For the first model we have two different methods, and for the latter we will discuss four methods.

3.1 Bag of Words

We will use the BoW model for both the Naive Bayes classifier and the multilayer perceptron. Let $\boldsymbol{d} = \{d_1, ..., d_n\}$ be a collection of documents, where d_{ij} denotes the number of occurrences of word i in document j. Furthermore, let $\boldsymbol{l} = \{l_1, ..., l_n\}$ be the according labels, where l_i is itself a set of possibly multiple labels.

Multinomial Naive Bayes Transformations. We will focus on the Multinomial version of the Naive Bayes model (MNB), where each word position is assumed to be independent of every other word. We will use several improvements proposed by [17], e.g. normalising the weight vectors.

To train the MNB model, we apply the transformations described in Eq. 1. For a test document t, with word i occurring t_i times, the document is labelled according to Eq. 2 for some threshold θ.

$$d_{ij} \overset{(1.1)}{=} d_{ij} \log \frac{\sum_k 1}{\sum_k 1_{i \text{ occurs in } k}} \qquad d_{ij} \overset{(1.2)}{=} \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}} \qquad w_{gi} \overset{(1.3)}{=} \log \sum_{j:g \in l_j} d_{ij} \tag{1}$$

$$w_g \overset{(2.1)}{=} \sum_i t_i w_{gi} \qquad w_g \overset{(2.2)}{=} \frac{w_g - \min w_g}{\max w_g - \min w_g} \qquad l(t) \overset{(2.3)}{=} \{g : w_g > \theta\} \tag{2}$$

In Eq. 1.1, we down-weight common words, a heuristic known as "inverse document frequency". Common words have little influence on the class of a document, but small variations can cause spurious correlations. Note that in most literature a "term frequency" heuristic precedes Eq. 1.1, we however found that this did not improve the accuracy. Therefore as shown in Eq. 1.1 we just use the term frequency. In order to prevent that document length affects the classification, we normalize every document according to Eq. 1.2. Finally, in Eq. 1.3 we add the weights of all documents belonging to the same genre.

For classification we first multiply each word frequency with the weight, as illustrated in Eq. 2.1. In standard multi-class classification, we could now assign a label to the class with the highest score. However, since the task is multi-label classification, we have to be able to assign multiple labels to a single document. We do this by first normalizing the weights according to Eq. 2.2, and then assign each label for which the weight is greater than the predefined threshold θ. By increasing θ we can trade-off recall for precision (defined in Sect. 5.1). We determine this threshold by means of the validation set.

Multi-layer Perceptron. The multi-layer perceptron (MLP) has been shown to be effective on a wide variety of tasks, despite its simplicity. We use a fully connected network, with two hidden layers. We use the ReLU activation function, and in every layer we apply L_2-normalisation before activation. The input of the MLP is again the BoW model, with the n most frequent words. Every word frequency is rescaled according to $d_{ij} = \log(1 + d_{ij})$, reducing the influence of frequently occurring words.

3.2 Skip-Gram Model

Many state-of-the-art techniques require dense word vectors as input. It is hypothesised that the techniques developed by e.g. [12] create dense word vectors that contain 'universal' features that can be used for various tasks. We will focus on the Skip-gram model [13]. In this model, each current word is used as an input, and the target is to predict the words that occur within a certain context c before and after the center word, as illustrated in Fig. 3. Furthermore, we use Negative sampling (NGE) as objective, where the task is to distinguish the target word from k negative samples drawn from a noise distribution. Since frequent words generally provide less information, we apply subsampling to all words as described in [13]. We train the model using all subtitles in our dataset, in Sect. 4.2 we denote the used hyperparameters. In order to explore the quality of the word vectors we use the Semantic-Syntactic Word Relationship test set, defined in [12]. This test set consists of five types of semantic questions and nine types of syntactic questions. The task is to predict a word, based on the relationship between three given words. An example for the semantic test is: "What word is similar to *Oslo* in the same way as *France* is similar to *Paris*?", the answer would be *Norway*. This test is performed by computing the vector $\boldsymbol{x} =$ *vector("france") - vector("paris") + vector("oslo")*, and finding the word that

has the smallest cosine distance to this vector $\boldsymbol{x}$ (different from the three question words). An answer is considered correct only if the closest word is identical to the word in the question. Table 1 shows the results on the word analogy task, indicating the effectiveness of the technique as well as generalizability of the used dataset. For the accuracy we denote both the percentage correct, and the number of correct classified pairs combined with the total number of pairs. Note that we used a subsection of the original test set, because some of the test words do not occur in our dataset. We evaluated 6,067 out of the original 8,869 semantic relations, for the syntactic relations we evaluated 10,300 out of 10,675 pairs. The results show that for the semantic relations the categories Common capital city and Man-Woman are learned very accurately, whereas Currency scores poorly. We expect that this is a result of the nature of movie subtitles, relationships (Man-Woman) and famous locations (Common capital city) play an important part in many movies, in contrast to currencies. The syntactic relations show a more balanced result, probably because all nine syntactic categories occur in spoken language.

Table 1. Results of semantic-syntactic word relationship test set.

Category	Accuracy
Semantic:	43.9 % (2665/6067)
Common capital city	86.6 % (433/506)
All capital cities	43.1 % (996/2310)
Currency	7.40 % (37/502)
City-in-state	35.4 % (824/2328)
Man-Woman	89.3 % (375/420)
Syntactic:	61.8 % (6362/10300)
Adjective to adverb	31.1 % (271/870)
Opposite	25.6 % (180/702)
Comparative	81.6 % (1087/1332)
Superlative	64.4 % (723/1122)
Present participle	62.7 % (622/992)
Nationality adjective	68.6 % (1044/1521)
Past tense	61.3 % (957/1560)
Plural nouns	82.1 % (1093/1332)
Plural verbs	44.3 % (385/869)

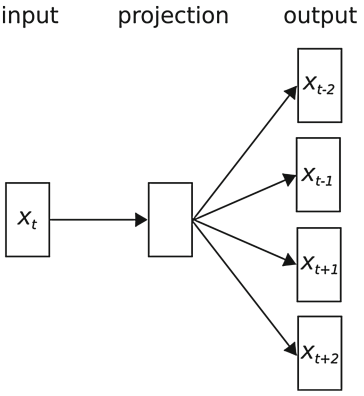


Fig. 3. The Skip-gram architecture, with a context size c of 2.

3.3 MLP on Histogram of Word Embeddings

Previous research has shown that first training a part of the model on an unsupervised task can reduce the training time and increase the accuracy on the supervised task [16]. Because the pre-trained word embeddings contain various

features, we expect that a basic model can find relationships between several features in order to learn a supervised task. Preliminary experiments have shown that taking scalar indicators (such as min, max, or mean) of a single feature over all words in combination with an MLP does not lead to satisfying results. Both the min and max operators can be affected by single, meaningless outliers, whereas the mean operator can potentially reduce significant positive and negative weights to a meaningless average. In order to capture more information we propose to use a histogram, where each word-embedding feature is described by a certain number of bins. Every bin denotes the relative frequency of a range of values for that specific feature. We will now describe the method used to convert a document to a word-embedding histogram, that subsequently can be used as an input for an MLP. Let every subtitle be consisting of n words, such that

$$X = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus ... \oplus \boldsymbol{x}_n \tag{3}$$

where $\boldsymbol{x}_t \in \mathbb{R}^k$ is the k-dimensional word embedding and $\oplus$ is the concatenation operator. Note that in most literature word embeddings are referred to by "words", we will use "concepts" because the word embeddings are actually the representation of the concept of a word, and not the word itself. The concatenation of the word embeddings results in a matrix $X \in \mathbb{R}^{n \times k}$, where n and k denote the number of words in the subtitle and dimension of the word embedding respectively. In order to use this matrix in combination with a histogram, we first need to scale the values such that we can use bins with a prefixed size and range. We normalize the matrix X according to

$$X_{ij} = \frac{X_{ij} - \min_i X_{ij}}{\max_i X_{ij} - \min_i X_{ij}} \tag{4}$$

We will now make a histogram along every word dimension, using s bins, where every bin has a width of size $1/s$. The range of the bins are denoted by $\{b_1 = [0, \frac{1}{s}), b_2 = [\frac{1}{s}, \frac{2}{s}), ..., b_s = [\frac{s-1}{s}, 1]\}$. For every bin b_l and every word dimension k we now calculate

$$H_{lk} = \text{card}(\{X_{jk} : X_{jk} \in b_l\}) \tag{5}$$

Subsequently, we calculate the L_1 norm

$$H_{lk} = \frac{H_{lk}}{\sum_{i=1}^{s} H_{ik}} \tag{6}$$

and calculate the z-score

$$Z_{lk} = \frac{H_{lk} - \mu}{\sigma} \tag{7}$$

where μ, and σ are the mean and standard deviation of all values in H respectively. The resulting matrix $Z \in \mathbb{R}^{s \times k}$ is then used as an input for an MLP.

3.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a feedforward neural network, originally used for image classification [7]. CNN models have shown to be effective on various NLP tasks, by utilising local features of the word embeddings [6]. We will now describe the CNN architecture. Let every document of length n be described by a sequence as defined in Eq. 3 (padded if necessary). Let $x_{i:i+j}$ denote the concatenation of concept x_i up to x_{i+j}. The convolutional filter $\boldsymbol{w} \in \mathbb{R}^{h \times k}$ is applied to a window of h concepts, which produces a new feature c_i. Note that k denotes again the word embedding size. We could in theory slide the convolution along the word-features too, there is however no reason to assume that any specific local relationships exist between concepts. The window of concepts $\boldsymbol{x}_{i:i+h-1}$ generates a new feature by

$$c_i = f(\boldsymbol{w} \circ \boldsymbol{x}_{i:i+h-1} + b) \tag{8}$$

where $\circ$ is the element-wise multiplication, $b \in \mathbb{R}$ is a bias term, and f is a non-linear function such as the sigmoid, hyperbolic tangent, etc. By applying this filter to all possible windows, we obtain a feature map $\boldsymbol{c} = [c_1, c_2, ..., c_{n-h+1}]$. Note that we can use multiple filters, with possibly different filter widths.

In order to capture the more significant events, we subsequently apply a max pooling operation on the feature map $\boldsymbol{c}$. Throughout the paper we will differentiate between two types of max pooling, namely max-over-time pooling and 1-D max pooling.

Max-over-time Pooling. The first technique extracts a single (maximum) scalar from each feature map. By using multiple convolutional filters, with varying filter widths, we obtain several features which are then passed on to a fully connected layer. This architecture was introduced by [6], and is referred to as concatenation-CNN (C-CNN). Whereas the architecture introduced by [6] uses a final softmax layer, we adapt the network for a multi-label problem by using a sigmoid activation output layer.

1-D Max Pooling. Max-over-time pooling reduces a feature map to a single feature, we can also reduce the feature map to several features, for different windows. In order to determine the maximum value for a window of size m we define

$$p_i = \max(c_{i:i+m-1}) \tag{9}$$

with $i = (1, 1+s, 1+2s, ...)$, where s denotes the size of the stride. In both the convolutional layer and the 1-D max pooling layer we can vary the stride, meaning that instead of moving the filter one step at the time, we move the filter several places per step. We use multiple filters for the same region, making it possible to learn complementary features from the same regions. With l filters, the generated l feature maps are combined to create a matrix $X \in \mathbb{R}^{l \times \lfloor (n-h-m+2)/s \rfloor}$. These feature maps are then used in combination with an LSTM network, as explained in Sect. 3.6.

3.5 Long Short-Term Memory Network

A Recurrent Neural Network (RNN) has the ability to capture time-dependent relations between words. It does this dynamically, without the use of fixed-size context windows. In particular, the Long Short-Term Memory Network (LSTM) [4] excels at tasks where long term dependencies are important. This network has received a lot of attention because of its capability of capturing important events throughout time series, and being relatively unsusceptible of gaps between important events. Given a sequence as described by Eq. 3, at time step t the LSTM network updates c_t and h_t with input x_t as follows

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{bmatrix} W \cdot \left[h_{t-1}, x_t \right] \tag{10}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t \tag{11}$$

$$h_t = o_t \circ \tanh(c_t) \tag{12}$$

where c_t and h_t are the memory and hidden state respectively, i_t, f_t, o_t, and $\hat{c}_t$ are the input gate vector, forget gate vector, output gate vector, and current cell state vector respectively. Note that in Eqs. 10 and 12 the functions sigm and tanh are applied element-wise. In order to map the output of the LSTM network to the output layer, we apply mean-over-time pooling on the output gate vectors o_t, meaning that we calculate the mean of all h_t values over all time steps t. Finally, the mean-over-time pooling is followed by a fully-connected layer with a sigmoid activation function.

The traditional LSTM network may have problems when the change of the parameters of one layer has an effect on the distribution of the input to all subsequent layers, also known as internal covariance shift. A solution proposed by [2], called Batch Normalized LSTM (BN-LSTM), normalizes both the input-to-hidden and hidden-to-hidden transformations by empirically estimating their means and standard deviations.

3.6 CNN-BN-LSTM

We discussed that CNN leverages the local features of words, whereas LSTM dominates in tasks where long term relations play a part. By combining the two techniques, we hope to get the best of both worlds. We start with applying a CNN layer, followed by a 1-D max pooling layer, as discussed in Sect. 3.4. The resulting matrix is then used as input for the LSTM network, such that there are $\lfloor (n - h - m + 2)/s \rfloor$ time steps, each with dimension l. Similar to the procedure discussed in Sect. 3.5, we subsequently apply mean-over-time pooling on the output gate vectors, together with a fully-connected layer with sigmoid activation. An example of this network is shown in Fig. 4.

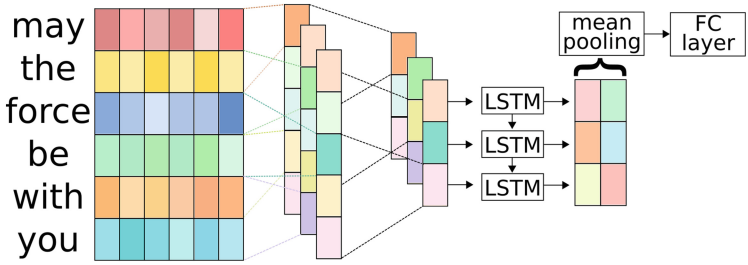


Fig. 4. Graphical representation of the CNN-BN-LSTM network. The hyperparameters of this example are as follows. The word-embedding size is 6. The convolutional layer uses a window size of 2, with a stride of 1, this is followed by 1-D max-pooling with a window size of 3, and a stride of 1.

4 Experimental Setup

4.1 Dataset

The proposed models are tested on the dataset introduced in Sect. 2. We split the dataset into a validation set and a train-test set of respectively 1500 and 14,000 subtitles, so that we tune the hyperparameters on the validation set and use cross validation on the train-test set. We use 7-fold cross validation in order to test the methods, the train set consists each time of 12,000 movies, the test set of 2,000 movies.

4.2 Hyperparameters and Training

The following hyperparameters are all determined by performing a grid search on the validation set. For the MNB model we only take into account words that occur more than 3 times. We use a classification threshold θ of 0.7 for the MNB model. For all other models we use a threshold value of 0.5.

For the BoW-MLP model we use the 50,000 most frequent words. The first hidden layer contains 512 nodes, the second layer 256. In both layers we apply the ReLU activation function, followed by dropout [19] with a dropout rate of 0.5.

Throughout all experiments we use a word embedding size of 300. We use static word embeddings, we thus apply no back propagation on the word embeddings in any of the experiments. We trained the word-embeddings on all subtitles, thus not only on the used subset for the multi-label classification task. The training was performed for 12 epochs, using a learning rate of 0.1, a mini-batch size of 16, a subsample threshold of 10^{-3}, a context size c of 5, and with 15 negative samples.

For the MLP-Histogram model we use 25 bins, followed by 128 hidden nodes in the first layer of the MLP, and 64 nodes in the second layer. Furthermore, in order to prevent overfitting we add Gaussian noise to the input with a mean

of 0, and a standard deviation of 0.02. Additionally, after each layer dropout is applied with a rate of 0.5. Finally, in each layer the ReLU activation function is applied.

The BN-LSTM model uses 300 hidden units, on both the input and output connections we use dropout with a rate of 0.2. We constrain the L_2-norm of the gradient to not exceed 10, this is known as gradient clipping.

For the CNN-BN-LSTM network we use similar LSTM hyperparameters, proceeded by a CNN. The CNN consists of 200 feature maps, with a window size of 8, a filter stride of 2, followed by a 1-D max pool filter of size 4, with a stride of 2. The activation function used in the CNN is the ReLU. Again we constrain the L_2-norm of the gradient to a maximum of 10.

In the C-CNN model we use filters of width 3,4 and 5, all with 128 feature maps. We apply dropout with a rate of 0.5, and constrain the L_2 norm again to 10.

For the CNN-BN-LSTM and the C-CNN model we pad the documents to a maximum length of 4000 words. In all models we use a mini-batch size of 20. We train the MLP-BoW and C-CNN for 6 epochs, all other models are trained for 10 epochs. We used the Adadelta update rule [20] for training, while shuffling the mini-batches.

Throughout all experiments (apart from training the word-embeddings) we anneal the learning rate α using exponential decay, defined by $\alpha = \alpha_0 r^{t/k}$, where α_0 is the initial learning rate, r is the decay rate, t is the iteration step, and k indicates the decay step, such that every k steps the learning rate is decayed. In all experiments we use a decay rate r of 0.97. For the MLP-Histogram, BN-LSTM, and CNN-BN-LSTM we used an initial learning rate of 0.1, for the MLP-BoW and C-CNN we use an initial learning rate of 0.005.

5 Results and Discussion

5.1 Metrics

In order to compare our models we will use the F_1 score, which takes into account both the recall and precision. Recall, precision, and the F_1 score for one label are respectively defined as:

$$\text{recall} = \frac{|\{\text{relevant labels}\} \cap \{\text{retrieved labels}\}|}{|\{\text{relevant labels}\}|} \tag{13}$$

$$\text{precision} = \frac{|\{\text{relevant labels}\} \cap \{\text{retrieved labels}\}|}{|\{\text{retrieved labels}\}|} \tag{14}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{15}$$

In order to calculate the final recall, precision, and F_1-score of the models, we calculate the mean scores over all three genres.

5.2 Results

The results of our models are listed in Table 2. Our baseline method (MNB) does not perform well. The model that performs best is the MLP-BoW model, with an average F_1-score of 0.77 ± 0.02. This is a significant higher result ($P < 0.005$) compared to the other models. The novel MLP-Histogram model achieves the second highest F_1-score. The BN-LSTM does not perform well on it own, however, in combination with a CNN layer (CNN-BN-LSTM) the model obtains the third best results. Finally, the C-CNN model is outperformed by all but two models.

Table 2. The results on the test set, after the specified number of epochs. Both the mean and standard deviation of the cross validation are displayed. We denote the recall, precision, and F_1-score for the three genres, together with the mean of the recall, precision, and F_1-score of the three genres.

Model	Romance			Thriller		
	Recall	Precision	F_1-score	Recall	Precision	F_1-score
MNB	0.93 ± 0.08	0.49 ± 0.14	0.64 ± 0.10	0.02 ± 0.01	0.67 ± 0.14	0.04 ± 0.03
MLP-BoW	0.75 ± 0.13	0.77 ± 0.04	0.76 ± 0.09	0.72 ± 0.01	0.77 ± 0.01	0.74 ± 0.08
MLP-Histogram	0.61 ± 0.03	0.72 ± 0.04	0.66 ± 0.02	0.78 ± 0.04	0.77 ± 0.02	0.77 ± 0.02
BN-LSTM	0.17 ± 0.13	0.72 ± 0.10	0.25 ± 0.14	0.76 ± 0.08	0.74 ± 0.05	0.75 ± 0.02
C-CNN	0.52 ± 0.17	0.69 ± 0.09	0.56 ± 0.10	0.77 ± 0.05	0.76 ± 0.09	0.76 ± 0.05
CNN-BN-LSTM	0.54 ± 0.05	0.74 ± 0.03	0.62 ± 0.04	0.76 ± 0.04	0.79 ± 0.02	0.77 ± 0.03
Model	Action			Mean		
	Recall	Precision	F_1-score	Recall	Precision	F_1-score
MNB	0.83 ± 0.11	0.71 ± 0.15	0.73 ± 0.08	0.59 ± 0.02	0.62 ± 0.05	0.47 ± 0.04
MLP-BoW	0.81 ± 0.06	0.82 ± 0.06	0.81 ± 0.05	0.76 ± 0.03	0.79 ± 0.03	**0.77 ± 0.02**
MLP-Histogram	0.80 ± 0.03	0.79 ± 0.06	0.79 ± 0.02	0.73 ± 0.01	0.76 ± 0.01	0.74 ± 0.01
BN-LSTM	0.75 ± 0.05	0.80 ± 0.06	0.77 ± 0.01	0.56 ± 0.04	0.75 ± 0.04	0.59 ± 0.04
C-CNN	0.75 ± 0.07	0.80 ± 0.08	0.77 ± 0.03	0.68 ± 0.06	0.75 ± 0.02	0.70 ± 0.04
CNN-BN-LSTM	0.78 ± 0.01	0.83 ± 0.05	0.80 ± 0.03	0.69 ± 0.03	0.78 ± 0.03	0.73 ± 0.03

5.3 Discussion

Our baseline model (MNB) does not perform well, the genre thriller has a very low recall and therefore a low F_1-score. The other two genres have however a very high recall (higher than all other models). We expect that the poor results on the genre thriller are caused by a combination of how the threshold is determined and the poor assumptions of the MNB model. We also experimented with n-grams, with n ranging from 1 to 3, but the performance decreased for n higher than 1.

The MLP-BoW model outperformed all other (more complex) models. This was in contrast with our expectations, because the model is relatively simple

compared to the other machine learning models. Not only is the F_1-score high, the training time was also relatively short. The fact that this model achieves the highest F_1-score could suggest that there exist some combinations of important 'indicator words' that are strong predictors for certain genres. We experimented with adding more layers to the network, but this had no significant positive effect on the results. Removing one layer had a negative effect on the accuracy.

Considering that the MLP-Histogram model only takes into account the relative frequency of word embedding feature values the model performs remarkably well. This is another illustration of the 'universal' features of word embeddings. Similar to the MLP-BoW model, the training time is relatively short. Furthermore, this newly proposed model performed best with using the word-embeddings.

The BN-LSTM model performs rather poorly. We expect that this is due to the length of the documents. A careful observation of Fig. 2 shows that the genre romance has relatively long subtitles. This could explain the poor results on this genre for models that are susceptible for document length. Although the BN-LSTM network does suffer less from vanishing gradients compared to other RNN networks, the network still has problems with documents of substantial length. Another explanation for the inadequacy of the BN-LSTM model could be that for this task word order is irrelevant and only the occurrence of certain words is important. Preliminary experiments have shown that stacking multiple BN-LSTM layers on top of each other had no effect on the final accuracy. The accuracy increases drastically with the use of batch normalization. Adding batch normalization also causes faster, more stable convergence. Furthermore, the model often diverged without the use of gradient clipping.

Contrary to the MNB model, we saw that for the C-CNN model the use of n-grams (by means of the filter widths) did increase the performance. Although the similar model introduced in [6] achieves state-of-the-art results on various tasks with a similar model, we find only moderate results on our task. One main difference is that the documents in the datasets used in [6] are significantly shorter compared to our dataset, making them less susceptible for outliers that can affect the max-over-time pooling.

By combining a CNN model with a BN-LSTM model (into the CNN-BN-LSTM model) we see a performance improvement compared to a separate C-CNN or BN-LSTM model. By combining the two methods we get the powerful feature extractor of the CNN model, and the capability of detecting long term dependencies of the LSTM model. The downside of this method is that even more hyperparameters have to be tuned. Exploratory research indicated that adding a CNN layer after the BN-LSTM or CNN-BN-LSTM model did not improve the accuracy.

6 Conclusion and Future Work

In this paper we described various techniques that can be used for multi-label classification of movie genres based on subtitles. First, we established a baseline using a multinomial naive Bayes (MNB) classifier combined with several

heuristics that "tackle the poor assumptions of MNB" [17]. We trained word embeddings on an unsupervised task, and showed that these embeddings contain indicative features for genre classification. We developed a novel architecture that combines a histogram of the word embeddings with an MLP. Despite the simple nature of this model it outperforms several more complex models. Both the C-CNN network and the BN-LSTM perform poorly on their own. However, by combining both techniques we observe a drastic increase in performance. The model that performs best is the MLP-BoW model, a surprising result given that many papers consider this network to be a baseline method.

We observed that simple models sometimes outperform more complex, state-of-the-art networks. The best network thus completely depends on the problem at hand. Therefore we would like to stress that exploring simpler text-classification methods is of great importance when a new dataset is studied. This directly relates to the principle of Occam's razor, stating that of all possible hypotheses, the one with the fewest assumptions should be used. When we decide to use a specific technique, we make certain assumptions about the data. A simple technique is less prone to overfit the data compared to a more complex technique, because it makes less assumptions about the data. With more assumptions, it is easier to choose parameters such that they only fit the observed data, and do not generalise well.

In follow-up work we would like to consider non-static word embeddings. In [6] it is shown that for certain tasks the performance improves when either non-static word embeddings, or a combination of both static and non-static word embeddings are used. Moreover, we would like to explore the use of random word embeddings and word embeddings trained by others, e.g. [13]. The final F_1-scores could be improved by using more advanced threshold techniques, and in future research the number of genres should be extended (up to 27). Finally, experiments on more datasets can be conducted, e.g. the Movie Review Sentiment dataset [10] and the Stanford Sentiment Treebank [18].

References

1. Clark, J., Koprinska, I., Poon, J.: A neural network based approach to automated e-mail classification. In: IEEE/WIC International Conference on Web Intelligence, pp. 702–705 (2003)
2. Cooijmans, T., Ballas, N., Laurent, C., Courville, A.C.: Recurrent batch normalization. CoRR, abs/1603.09025 (2016)
3. Harris, Z.: Distributional structure. Word **10**(23), 146–162 (1954)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
5. Johnson, R., Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058 (2014)
6. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)

8. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 4–15. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026666
9. Lison, P., Tiedemann. J.: OpenSubtitles 2016: Extracting large parallel corpora from movie and TV subtitles. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (2016)
10. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150 (2011)
11. McCallum, A., Nigam, K.: A comparison of event models for Naive Bayes text classification. In: AAAI 1998 Workshop on Learning for Text Categorization, vol. 752, pp. 41–48 (1998)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR, abs/1301.3781 (2013)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S, Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
14. Mikolov, T., Yih, S.W.-T., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 746–751 (2013)
15. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
16. Alec Radford, Rafal Józefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. CoRR, abs/1704.01444 (2017)
17. Rennie, J.D.M., Shih, L., Teevan, J., Karger, D.R.: Tackling the poor assumptions of Naive Bayes text classifiers. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 616–623 (2003)
18. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
19. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
20. Zeiler, M.D.: ADADELTA: An adaptive learning rate method. CoRR, abs/1212.5701 (2012)
21. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems, pp. 649–657 (2015)

Learning to Play Donkey Kong Using Neural Networks and Reinforcement Learning

Paul Ozkohen[1], Jelle Visser[1], Martijn van Otterlo[2], and Marco Wiering[1(✉)]

[1] University of Groningen, Groningen, The Netherlands
{p.m.ozkohen,j.visser.27}@student.rug.nl, m.a.wiering@rug.nl
[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
m.van.otterlo@vu.nl

Abstract. Neural networks and reinforcement learning have successfully been applied to various games, such as Ms. Pacman and Go. We combine multilayer perceptrons and a class of reinforcement learning algorithms known as actor-critic to learn to play the arcade classic Donkey Kong. Two neural networks are used in this study: the actor and the critic. The actor learns to select the best action given the game state; the critic tries to learn the value of being in a certain state. First, a base game-playing performance is obtained by learning from demonstration, where data is obtained from human players. After this off-line training phase we further improve the base performance using feedback from the critic. The critic gives feedback by comparing the value of the state before and after taking the action. Results show that an agent pre-trained on demonstration data is able to achieve a good baseline performance. Applying actor-critic methods, however, does usually not improve performance, in many cases even decreases it. Possible reasons include the game not fully being Markovian and other issues.

Keywords: Machine learning · Neural networks
Reinforcement learning · Actor-critic · Games · Donkey Kong
Platformer

1 Introduction

Games have been a prime subject of interest for machine learning in the last few decades. Playing games is an activity enjoyed exclusively by humans, which is why studying them in the pursuit of *artificial intelligence (AI)* is very enticing. Building software agents that perform well in an area that requires human-level intelligence would thus be one step closer to creating strong, or: general, AI, which can be considered one of the primary goals of the entire field.

The third author acknowledges support from the Amsterdam academic alliance (AAA) on data science.

B. Verheij and M. Wiering (Eds.): BNAIC 2017, CCIS 823, pp. 145–160, 2018.
https://doi.org/10.1007/978-3-319-76892-2_11

Reinforcement learning (RL) techniques have often been used to achieve success in creating game-playing agents [5,7]. RL requires the use of certain functions, such as a *policy function* that maps states to actions and a *value function* that maps states to values. The values of these functions could, for example, be stored in tables. However, most non-trivial environments have a large state space, particularly games where states are continuous. Unfortunately, tables would have to become enormous in order to store all the necessary function information. To solve this problem in RL, *function approximation* can be applied, often using *neural networks*. A famous recent example of this is the ancient board game Go, in which DeepMind's AI *AlphaGo* was able to beat the world's best players at their own game [7]. Besides traditional games, it was used to learn to play video games. For example, DeepMind used a combination of convolutional neural networks and Q-learning to achieve good gameplay performance at 49 different Atari games, and was able to achieve human-level performance on 29 of them [5]. That study shows how an RL algorithm can be trained purely on the raw pixel images. The upside of that research is that a good game-playing performance can be obtained without handcrafting game-specific features. The Deep Q-Network was able to play the different games without any alterations to the architecture of the network or the learning algorithms. However, the downside is that deep convolutional networks require exceptional amounts of computing power and time. Furthermore, one could speculate how well performance of each individual game could be improved by incorporating at least some game-relevant features. Still, it is impressive how the network could be generalized to very different games.

An alternative approach is to use hand-crafted game-specific features. One such game where this was successfully applied is Ms. Pac-Man, where an AI was trained to achieve high win rates using higher-order, game-specific features [3]. This approach shows that good performance can be obtained with a small amount of inputs, therefore severely reducing computation time.

In this paper we present an approach to machine learning in games that is more in line with the second example. We apply RL methods to a video game based on *Donkey Kong*, an old arcade game that was released in 1981 by Nintendo [4]. The game features a big ape called Donkey Kong, who captures princess Pauline and keeps her hostage at the end of each stage. It is up to the hero called Jumpman, nowadays better known as Mario, to climb all the way to the end of the level to rescue this damsel in distress. Besides climbing ladders, the player also has to dodge incoming barrels being thrown by Donkey Kong, which sometimes roll down said ladders.

This game provides an interesting setting for studying RL. Unlike other games, Donkey Kong does not require expert strategies in order to get a decent score and/or get to the end of the level. Instead, *timing* is of the utmost importance for surviving. One careless action can immediately lead Mario to certain death. The game also incorporates unpredictability, since barrels often roll down ladders in a random way. The intriguing part of studying this game is to see whether RL can deal with such an unpredictable and timing-based continuous environment. We specifically focus on the very first level of the Donkey Kong

game, as this incorporates all the important elements mentioned above while also making the learning task simpler. Other levels contain significantly different mechanics, such as springs that can launch Mario upwards if he jumps on it, or vertically moving platforms. We do not consider these mechanics in this research.

For this study we used a specific RL technique called *actor-critic* [9]. In each in-game step, the *actor* (player) tries to select the optimal action to take given a game state, while the *critic* tries to estimate the given state's value. Using these state-value estimates, the critic gives feedback to the actor, which should improve the agent's performance while playing the game. More specifically, we employ a variant of actor-critic: the *actor-critic learning automaton* (ACLA) [14].

Both the actor and the critic are implemented in the form of a *multilayer perceptron* (MLP). Initializing the online learning with an untrained MLP would be near-impossible: the game environment is too complex and chaotic for random actions to lead to good behavior (and positive rewards). In order to avoid this, both the actor and the critic are trained offline on demonstration data, which is collected from a set of games being played by human players.

The main question this paper seeks to answer is: is a combination of neural networks and actor-critic methods able to achieve good gameplay performance in the game Donkey Kong? In the next sections we will first define the domain and its features, after which we discuss our machine learning setup and methodology and we conclude with results and discussion.

2 The Domain: A Donkey Kong Implementation

A framework was developed that allows the user to test several RL techniques on a game similar to the original Donkey Kong. The game itself can be seen in Fig. 1 and was recreated from scratch as a Java application.

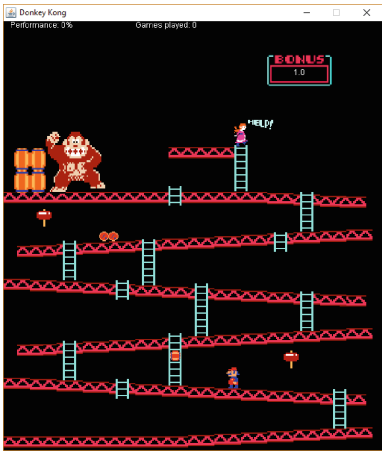


Fig. 1. Recreation of Donkey Kong.

The goal of the game is to let the player reach the princess at the top of the level. The agent starts in the lower-left corner and has to climb ladders in order to ascend to higher platforms. In the top-left corner, we find the game's antagonist Donkey Kong, who throws barrels at set intervals. The barrels roll down the platforms and fall down when reaching the end, until they disappear in the lower-left of the screen. When passing a ladder, each barrel has a 50% chance of rolling down the ladder, which adds a degree of unpredictability to the barrel's course. The player touching a barrel results in an instant loss (and "game over"), while jumping over them nets a small score. Additionally, two power-ups (hammers) can be picked up by the player when he collides with them by either a walking or jumping action, which results in the barrels being destroyed upon contact with the agent, netting a small score gain as well. This powerup is temporary. The agent can execute one out of seven actions: *walking* (left or right), *climbing* (up or down), *jumping* (left or right) or *doing nothing* (standing still). The game takes place in a 680 × 580 window. Mario moves to the left and right at a speed of 1.5 pixels, while Mario climbs at a speed of 1.8 pixels. A jump carries Mario forward around 10 pixels, which implies it requires many actions to reach the princess from the initial position.

While this implementation of the game is quite close to the original game, there are several differences between the two versions of the game:

- The game speed of the original is slower than in the recreation.
- The barrels are larger in the original. To reduce the difficulty of our game, we made the barrels smaller.
- The original game contains an oil drum in the lower-left corner which can be ignited by a certain type of barrel. Upon ignition, the barrel produces a flame that chases the player. This has been entirely left out in the recreation.
- The original game consists of several different levels. The recreation only consist of one level, which is a copy of the first level from the original.
- The original game uses some algorithm for determining whether a barrel will go down a ladder or not, which appears to be based on the player's position relative to the barrel and the player's direction. The code of the original is not available, so instead we opted for a simple algorithm where the barrels' odds of rolling down a ladder is set to be simply 50% at any given time.

The built environment supports *manual* mode, in which a human player can interact with the game, and two *automated* modes in which an MLP is used to control Mario (either only using an actor network, or learning with a critic). While there are a few notable differences between the original game and our recreation both versions are still quite similar. It is therefore reasonable to assume that any AI behavior in the recreation would translate to the original.

3 Generalization: Multilayer Perceptrons

The actor and critic are implemented in the form of an MLP, a simple feed-forward network consisting of an input layer, one or more hidden layers and an output layer. Like the game itself, the MLP was built from scratch in *Java*, meaning no external packages were used.

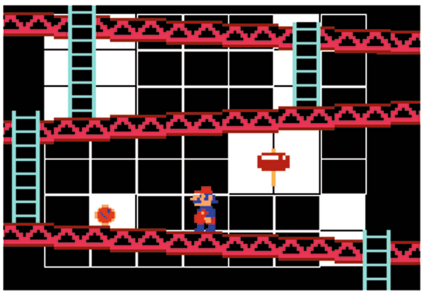


Fig. 2. Visualization of the vision grid that tracks objects directly around the agent, granting Mario local vision of his immediate surroundings. Note that while only one grid can be distinguished, there are actually three vision grids stacked on top of each other, one for each object type.

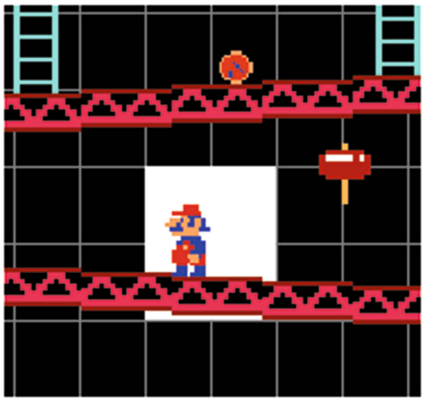


Fig. 3. Visualization of the level-wide grid that tracks the current location of the agent. While not visible in this image, the grid spans the entire game environment.

3.1 Feature Construction for MLP Input

This section provides an overview of how inputs for the MLPs are derived from the game state. Two algorithms employ several varieties of grids that are used to track the location of objects in the game. Each cell in each grid corresponds to one input for the MLP. Besides these grids, several additional inputs provide information about the current state of the game.

There are three types of objects in the game that the agent can interact with: *barrels*, *powerups* and *ladders*. We use three different vision grids that keep track of the presence of these objects in the immediate surroundings of Mario. A similar method was used by Shantia et al. [6] for the game *Starcraft*.

First of all, the MLP needs to know how to avoid getting killed by barrels, meaning it needs to know where these barrels are in relation to Mario. Barrels that are far away pose no immediate threat. This changes when a barrel is on the same platform level as Mario: at this point, Mario needs to find a way to avoid a collision with this barrel. Generally, this means trying to jump over it. Barrels on the platform level above Mario need to be considered as well, as they could either roll down a ladder or fall down the end of the platform level, after which they become an immediate threat to the agent. The second type of objects, ladders, are the only way the agent can climb to a higher platform level, which is required in order to reach the goal. The MLP therefore needs to know if there are any ladders nearby and where they are. Finally, the powerups provide

the agent the ability to destroy the barrels, making Mario invincible for a short amount of time. The powerups greatly increase the odds of survival, meaning it's important that the MLP knows where they are relative to Mario.

In order to track these objects, we use a set of three grids of 7×7 cells, where each grid is responsible for tracking one object type. The grids are fixed on Mario, meaning they move in unison. During every time step, each cell detects whether it's colliding with the relevant object. Cells that contain an object are set to 1.0, while those that do not are set to 0.0. This results in a set of $3 \times 49 = 147$ Boolean inputs. The princess is always above the player, while barrels that are below the player pose no threat whatsoever. We are therefore not interested in what happens in the platform levels *below* the agent, since there rarely is a reason to move downwards. Because of this, these *vision grids* are not centered around the agent. Instead, five of the seven rows are above the agent while there is only one row below. An example of the vision grid is shown in Fig. 2.

The MLP requires knowledge of the location of the agent in the environment. This way it can relate outputs (i.e. player actions) to certain locations in the map. Additionally, this knowledge is essential for estimating future rewards by the critic, which will be explained further in Sect. 5. The agent's location in the game is tracked using a 20×20 grid that spans the entire game environment. Like the vision grid, each cell in the agent tracking grid provides one boolean input. The value of a cell is 1.0 if the agent overlaps with it, 0.0 if it does not. This *agent tracking grid* provides $20 \times 20 = 400$ Boolean inputs. An example tracking grid can be seen in Fig. 3.

There are some additional features, such as Booleans that track whether Mario is currently jumping or climbing. The total amount of features is the sum of 147 vision grid cells, 400 agent tracking grid cells and 4 additional inputs, resulting in 551 in total. The four additional inputs are extracted from the game state as follows:

- A boolean that tracks whether the agent can climb (i.e. is standing close enough to a ladder). This prevents the agent from trying to climb while this is not possible.
- A boolean that tracks whether the agent is currently climbing. This prevents the agent from trying to do any other action besides climbing while on a ladder.
- A boolean that tracks whether the agent currently has an activated powerup. This is used to teach the MLP that it can destroy barrels while under the influence of a powerup, as opposed to having to jump over them.
- A real decimal number in the range [0, 1] that tracks how much time a powerup has been active. We compute it as the ratio $\frac{t}{d}$ between the time passed since the powerup was obtained (t) and the total time a powerup remains active (d).

3.2 MLP Output

For the **actor** the output layer consists of seven neurons, each neuron representing one of the seven possible player actions: moving left or right, jumping left

or right, climbing up or down, or standing still. During training using demonstration data, the target pattern is encoded as a *one-hot vector*: the target for the output neuron corresponding to the action taken has a value of 1.0, while all other targets are set to 0.0. During gameplay, the MLP picks an action based on *softmax* action selection [9]. Here, each action is given a probability based on its activation. Using a *Boltzmann distribution*, we can transform a vector a of length n, consisting of real output activation values, into a vector $\sigma(a)$ consisting of n real values in the range $[0, 1]$. The probability for a single output neuron (action) i is calculated as follows:

$$\sigma(a_i) = \frac{e^{a_i/\tau}}{\sum_{j=1}^{n} e^{a_j/\tau}} \text{ for i} = 1, \ldots, \text{n} \tag{1}$$

where τ is a positive real temperature value which can be used to induce exploration into action selection. For $\tau \to \infty$, all actions will be assigned an equal probability, while for $\tau \to 0$ the action selection becomes purely *greedy*. During each in-game timestep, each output neuron in the actor-MLP is assigned a value using Eq. 1. This value stands for the probability that the actor will choose a certain action during this timestep. The output layer of the **critic** consists of one numerical output, which is a *value estimation* of a given game state. This will be explained further in Sect. 5.2.

3.3 Activation Functions

Two different activation functions were used for the hidden layers: the sigmoid function and the Rectified Linear Unit (ReLU) function. Given an activation a, the sigmoid output value $\sigma(a)$ of a neuron is:

$$\sigma(a) = 1/(1 + e^{-a}) \tag{2}$$

The ReLU output value is calculated using:

$$\sigma(a) = max(0, a) \tag{3}$$

Both activation functions are compared in order to achieve the best performance for the MLP. This will be elaborated upon in Sect. 6.

4 Learning from Demonstration

RL alone is sometimes not enough to learn to play a complex game. Hypothetically, we could leave out offline learning and initialize both the actor and the critic with an untrained MLP, which the critic would have to improve. In a game like Donkey Kong however, this would lead to initial behavior to consist of randomly moving around without getting even remotely close to the goal. In other words: it would be hard to near-impossible for the actor to reach the goal state, which is necessary for the critic to improve gameplay behavior. This means

that we need to *pre-train* both the actor and the critic in order to obtain a reasonable starting performance. For this, we utilized *learning from demonstration* (LfD) [1]. A dataset of input and output patterns for the MLP was created by letting the first two authors play 50 games each. For each timestep, an input pattern is extracted from the game state as explained before. Additionally, the action chosen by the player at that exact timestep (and the observed reward) is stored. The critic uses the reward to compute a target value as is explained later. All these corresponding input-output patterns make up the data on which the MLPs are pre-trained.

5 Reinforcement Learning Framework

Our game is modeled as a *Markov Decision Process* (MDP), which is the framework that is used in most RL problems [9,13]. An MDP is a tuple $\langle S, A, P, R, \gamma \rangle$, where S is the set of all states, A is the set of all actions, $P(s_{t+1}|s_t, a)$ represents the transition probabilities of moving from state s_t to state s_{t+1} after executing action a and $R(s_t, a, s_{t+1})$ represents the reward for this transition. The discount factor γ indicates the importance of future rewards. Since in Donkey Kong there is only one main way of winning the game, which is saving the princess, the future reward of reaching her should be a very significant contributor to the value of a state. Furthermore, as explained in Sect. 2, the agent does not move very far after each action selection. When contrasted with the size of the game screen, this means that around 2000 steps are needed to reach the goal, where 7 actions are possible at each step, leading to a very challenging environment. For these reasons, the discount factor γ is set to 0.999, in order to cope with this long horizon, such that values of states that are, for example, a 1000 steps away from the goal still get a portion of the future reward of saving the princess. A *value function* $V(s_t)$ is defined, which maps a state to the expected value of this state, indicating the usefulness of being in that state. Besides the value function, we also define a *policy function* $\pi(s_t)$ that maps a state to an action. The goal of the RL is to find an *optimal* policy $\pi^*(s_t)$ such that an action is chosen in each state in order to maximize the obtained rewards in the future. The environment is assumed to satisfy the *Markov property*, which assumes that the history of states is not important to determine the probabilities of state transitions. Therefore, the transition to a state s_{t+1} depends only on the current state s_t and action a_t and not on any of the previous states encountered.

In our Donkey Kong framework, the decision-making agent is represented by Mario, who can choose in each state one of the seven actions to move to a new state, where the state is uniquely defined by the combination of features explained earlier. Like in the work done by Bom et al. [3], we use a fixed reward function based on specific in-game events. Choosing actions in certain states can trigger these events, leading to positive or negative rewards. We want the agent to improve its game-playing performance by altering its policy. Rewards give an indication of whether a specific action in a specific state led to a good or a bad outcome. In Donkey Kong, the ultimate goal is to rescue the princess at

the top of the level. Therefore, the highest positive reward of 200 is given in this situation. One of the challenging aspects of the game is the set of moving barrels that roll around the level. Touching one of these barrels will immediately kill Mario and reset the level, so this behavior should be avoided at all costs. Therefore, a negative reward of −10 is given, regardless of the action chosen by Mario. Jumping around needlessly should be punished as well, since this can lead Mario into a dangerous state more easily. For example, jumping in the direction of an incoming barrel can cause Mario to land right in front of it, with no means of escape left. The entire set of events and the corresponding rewards are summarized in Table 1.

Table 1. Game events and their corresponding rewards. A 'needless' jump penalty is only given if the agent jumped, but did not jump over a barrel nor did the agent pick up a powerup.

Event	Reward
Save princess	+200
Jumping over a barrel	+3
Destroy barrel with powerup	+2
Pick up powerup	+1
Getting hit by barrel	−10
Needless jump	−20

5.1 Temporal Difference Learning

Our RL algorithms are a form of *temporal difference* (TD) learning [8,9]. The advantage of TD methods is that they can immediately learn from the raw experiences of the environment as they come in and no model of the environment needs to be learned. This means that we can neglect the P-part of the MDP tuple explained earlier. TD methods allow learning updates to be made at every time step, unlike other methods that require the end of an episode to be reached before any updates can be made (such as *Monte Carlo* algorithms). Central to TD methods is the value function, which estimates the value of each state based on future rewards that can be obtained, starting at this state. Therefore, the value of a state s_t is the expected total sum of discounted future rewards starting from state s_t:

$$V(s_t) = E\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\Big] \tag{4}$$

Here, s_t is the state at time t, γ is the discount factor and R_{t+k+1} is the reward at time $t+k+1$. We can take the immediate reward observed in the next state out of the sum, together with its discount factor:

$$V(s_t) = E\Big[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^{k+1} R_{t+k+2}\Big] \tag{5}$$

We observe that the discounted sum in Eq. 5 is equal to the definition of the value function $V(s_t)$ in Eq. 4, except one time step later into the future. Substituting Eq. 4 into Eq. 5 gives us the final value function prediction target:

$$V(s_t) = E\Big[R_{t+1} + \gamma V(s_{t+1})\Big] \tag{6}$$

Therefore, the predicted value of a state is the reward observed in the next state plus the discounted next state value.

5.2 Actor-Critic Methods

Actor-critic methods are based on the TD learning idea. However, these algorithms represent both the policy and the value function separately, both with their own weights in a neural network or probabilities/values in a table. The policy structure is called the actor, which takes actions in states. The value structure is called the critic, which criticizes the current policy being followed by the actor. The structure of the actor-critic model is illustrated in Fig. 4.

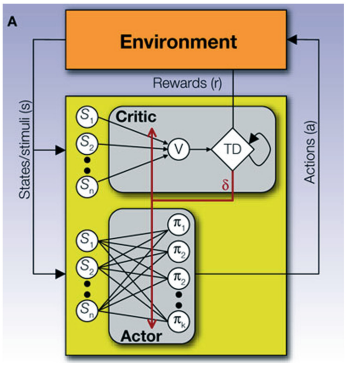


Fig. 4. The architecture of actor-critic methods [10].

The environment presents the representation of the current state s_t to both the actor and the critic. The actor uses this input to compute the action to execute, according to its current policy. The actor then selects the action, causing the agent to transition to a new state s_{t+1}. The environment now gives a reward

based on this transition to the critic. The critic observes this new state and computes its estimate for this new state. Based on the reward and the current value function estimation, both R_{t+1} and $\gamma V(s_{t+1})$ are now available to be incorporated into both making an update to the critic itself, as well as computing a form of feedback for the actor. The critic looks at the difference of the values of both state s_t and s_{t+1}. Together with the reward, we can define the feedback δ_t at time t, called the TD error, as follows:

$$\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{7}$$

When a terminal state is encountered (hit by a barrel or saving the princess) the value of the next state, $\gamma V(s_{t+1})$, is set to 0. The tendency to select an action has to change, based on the following update rule [9]:

$$h(a_t|s_t) = h(a_t|s_t) + \beta\delta_t, \tag{8}$$

where $h(a_t|s_t)$ represents the tendency or probability of selecting action a_t at state s_t and β is a positive step-size parameter between 0 and 1.

In the case of neural networks, both the actor and the critic are represented by their own multilayer perceptron. The feedback computed by the critic is given to the actor network, where the weights of the output node of the actor corresponding to the chosen action are directly acted upon. The critic is also updated by δ_t. Since the critic approximates the value function $V(s_t)$ itself, the following equation (where the updated V' is computed from V):

$$\begin{aligned} V'(s_t) &= V(s_t) + \delta_t, \\ &= V(s_t) + R_{t+1} + \gamma V(s_{t+1}) - V(s_t) \end{aligned}$$

is reduced to:

$$V'(s_t) = R_{t+1} + \gamma V(s_{t+1}), \tag{9}$$

which is, once again, the value function target for the critic. We can see that as the critic keeps updating and improves its approximation of the value function, $\delta_t = V'(s_t) - V(s_t)$ should converge to 0, which decreases the impact on the actor likewise, which can converge to a (hopefully optimal) policy.

We employ the actor-critic algorithm called *actor-critic learning automaton* [14]. This algorithm functions in the same basic way as standard actor-critic methods, except in the way the TD error is used for feedback. As explained before, standard actor-critic methods calculate the feedback δ_t and use this to alter the tendency to select certain actions by changing the parameters of the actor. ACLA does not use the exact value of δ_t, but only looks at whether or not an action selected in the previous state was good or bad. Therefore, instead of the value, the sign of δ_t is used, and a one-hot vector is used as the target.

6 Experiments and Results

In our experiments we define the performance as the percentage of games where the agent was able to reach the princess: $\frac{gamesWon}{gamesPlayed}$. In the first experiment,

the parameters for the MLP trained using learning from demonstration were optimized in order to achieve a good baseline performance. We then perform 10 runs of 100 games to see how the optimized actor performs without any RL. For the second experiment, we compare the performance of only the actor versus an actor trained with ACLA for 5 different models. Between each model, the performance of the Actor-MLP is varied: we do not only want to see if ACLA is able to improve our best actor, but we want to know whether it can increase the performance of lower-performing actors as well.

6.1 Model Selection for RL

During the RL experiments, the ACLA algorithm was applied to a few different actor networks. The networks were selected based on their performance on the 10×100 games. For example, the first model we considered is an Actor trained with the combination of the best parameters for the sigmoid activation function, found as a result of a separate parameter optimization phase. We consider two networks using the sigmoid activation functions and two networks using the ReLU activation function. The fifth model differs from the other 4: this model is only pre-trained for 2 epochs. This small amount of pre-training means that the model is quite bad, leaving much room for possible improvement by the critic. Besides model 5, the two sigmoid models were trained until a minimum change in error between epochs of 0.00005 was reached, while the two ReLU models had a minimum change threshold of 0.0007. The reason that the ReLU models' threshold is higher than the Sigmoid models', is that preliminary results have shown that the error did not decrease further after extended amounts of training for MLPs using ReLU. Table 2 displays and details all 5 models that we considered and tried to improve during the RL trials together with their performance and standard error (SE).

Table 2. Details of the 5 models that were used in the RL trials. The performance means the % of trials in which Mario gets to the Princess in 100 games. The results are averaged over 10 simulations with MLPs trained from scratch.

Model	N hidden layers	N hidden nodes	Learning rate	Activation function	Performance
1	2	200	0.01	Sigmoid	56.6% (SE: 1.08)
2	1	50	0.001	Sigmoid	29.9% (SE: 1.08)
3	1	100	0.005	ReLU	48.6% (SE: 2.02)
4	2	50	0.001	ReLU	50.6% (SE: 1.46)
5	1	80	0.01	Sigmoid	12.6% (SE: 0.90)

6.2 Online Learning Experiments

This section explains how the RL trials were set up. Each of the 5 models is trained during one ACLA session. This learning session lasts 1000 games, where

the temperature of the Boltzmann distribution starts at a value of 8. This temperature is reduced every 200 games, such that the last 200 games are run at the lowest temperature of 4. Preliminary results showed that most networks performed best at this temperature. The ACLA algorithm is applied at every step, reinforcing positive actions. The learning rate of the actor is set to 0.0001, so that ACLA can subtly push the actor into the right direction. The critic also uses a learning rate of 0.0001. Such low learning rates are required to update the approximations (that were already trained well on the demonstration data) *cautiously*. Setting the learning rate too high causes the networks to become unstable. In this event, state values can become very negative, especially when the actor encounters a lot of negative rewards.

After the 1000-games training sessions, the performance of the actors trained with ACLA were compared to the performance of the actors before training with ACLA. For each of the 5 models, both actors were tested in 10×100 games, both with a fixed temperature of 4. The results of the trained actor performances are shown in Table 3. The final results are shown in Fig. 5, where the performance of each model's actor versus the model's actor trained with ACLA are shown.

Table 3. Results of the models trained with ACLA on 10 runs

Statistic	Model 1	Model 2	Model 3	Model 4	Model 5
MEAN	45.8%	31.2%	44.5%	20.8%	26.4%
SE	1.45	1.46	0.76	1.34	1.59

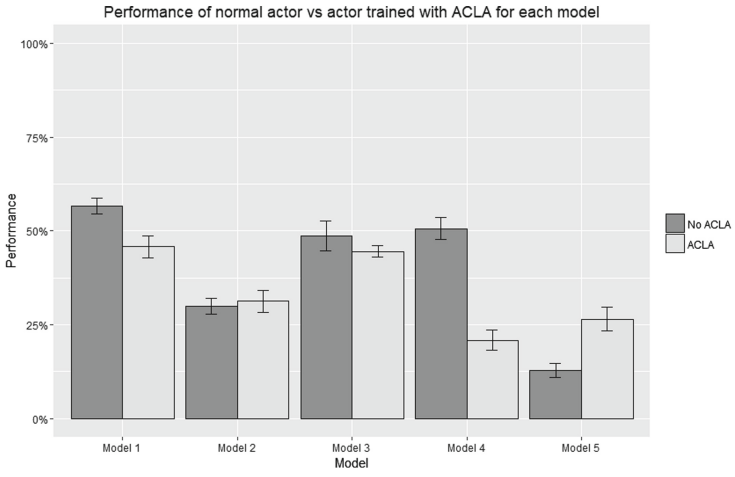


Fig. 5. Performances of the actor trained with vs. without ACLA for each model. The error bars show two standard errors (SE) above and below the mean

6.3 Analysis of Results

Looking at Fig. 5, the differences in performance can be seen for each model, together with standard error bars which have a length of $4 \times SE$. From this figure, we see that the error bars of models 2 and 3 overlap. This might indicate that these differences in performances are not significant. The other 3 models do not have overlapping error bars, suggesting significance. In order to test for significance, we use a nonparametric Wilcoxon rank sum test, since the performance scores are not normally distributed. The Wilcoxon rank sum test confirms a significant effect of ACLA on models 1 ($W = 41.5$, $p < 0.05$), 4 ($W = 100$, $p < 0.05$) and 5 ($W = 0$, $p < 0.05$), but not on models 2 ($W = 41.5$, $p > 0.05$) and 3 ($W = 27$, $p > 0.05$). These results seem to confirm the observations made earlier with respect to the error bars in Fig. 5.

6.4 Discussion

Using parameter optimization, we were able to find an MLP that is able to obtain a reasonable baseline performance by using learning from demonstration. The best model, model 1, was able to achieve an average performance of winning the game-level of 56.6%. In addition to this, several MLPs were trained with different parameter settings, resulting in a total of 5 neural net models. The performance of these 5 models varies based on how robustly the actor-critic method is able to improve these models.

While the performance achieved by an actor that is only trained offline is reasonable, ACLA does not usually seem to be able to improve this any further. Even worse, the actor's performance can start to decline over time. Only a model that is barely pre-trained on demonstration data can obtain a significant improvement. We therefore conclude that a combination of neural nets and actor-critic is in most cases not able to improve on a reasonable policy that was obtained through learning from demonstration.

7 Conclusions

We have presented our Donkey Kong simulation and our machine learning experiments to learn good gameplay. We have employed LfD to obtain reasonable policies from human demonstrations, and experimented with RL (actor-critic) to learn the game in an online fashion. Our results indicate that the first setting is more stable, but that the second setting has possibly still potential to improve automated gameplay. Overall, for games such as ours, it seems that LfD can go a long way if the game can be learned from relevant state-action examples. It may well be that for Donkey Kong, in our specific level, the right actions are clear from the current game state and additional delayed reward aspects play a less influencing role, explaining the lesser effect of RL in our experiments. More research is needed to find out the relative benefits of LfD and RL. Furthermore, in our experiments we have focused on the global performance measure of

percentage of games won. Further research could focus on more finegrained performance measures using the (average) reward, and experiment with balancing the various (shaping) rewards obtained for game events (see Table 1).

Future research could result in better playing performance than those obtained in this research. Actor-critic methods turned out to not be able to improve the performance of the agent. Therefore, other reinforcement learning algorithms and techniques could be explored, such as Q-learning [12], advantage learning [2] or Monte Carlo methods. A recent method has been introduced called the Hybrid Reward Architecture, which has been applied to Ms. Pac-Man to achieve a very good performance [11]. Applying this method to Donkey Kong could yield better results. Additionally, it would be interesting to see whether having more demonstration data positively affects performance. Since we only focused on the very first level of the game, further research is needed to make the playing agent apply its learned behavior to different levels and different mechanics.

References

1. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Proceedings of the International Conference on Machine Learning, pp. 12–20 (1997)
2. Baird, L.: Residual algorithms: reinforcement learning with function approximation. In: Proceedings of the Twelfth International Conference on Machine Learning, pp. 30–37 (1995)
3. Bom, L., Henken, R., Wiering, M.: Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs. In: IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (2013)
4. Donkey Kong fansite wiki. http://donkeykong.wikia.com/wiki/Nintendo. Accessed Sept 2017
5. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
6. Shantia, A., Begue, E., Wiering, M.: Connectionist reinforcement learning for intelligent unit micro management in Starcraft. In: The 2011 International Joint Conference on Neural Networks, pp. 1794–1801 (2011)
7. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
8. Sutton, R.S.: Learning to predict by the methods of temporal differences. Mach. Learn. **3**(1), 9–44 (1988)
9. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
10. Takahashi, Y., Schoenbaum, G., Niv, Y.: Silencing the critics: understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an actor/critic model. Front. Neurosci. **2**(1), 86–99 (2008)
11. van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., Tsang, J.: Hybrid reward architecture for reinforcement learning (2017). https://arxiv.org/abs/1706.04208

12. Watkins, C.J.: Learning from delayed rewards. Ph.D. Thesis, University of Cambridge, England (1989)
13. Wiering, M., van Otterlo, M.: Reinforcement Learning: State of the Art. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3
14. Wiering, M.A., Van Hasselt, H.: Two novel on-policy reinforcement learning algorithms based on TD(λ)-methods. In: 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, pp. 280–287 (2007)

Author Index

Bex, Floris 32
Bontempi, Gianluca 101
Bosse, Tibor 61
Brandt, Armin 76

Collins, Pieter 116

de Weerd, Harmen 86
de Weerdt, Mathijs 16, 46
Driessens, Kurt 116

Goedschalk, Linford 61

Le Borgne, Yann-Aël 101

Otte, Marco 61
Ozkohen, Paul 145

Pieters, Mathijs 131
Polevoy, Gleb 16, 46
Prakken, Henry 32

Reggiani, Claudio 101
Renooij, Silja 32
Roos, Nico 1
Roschewitz, David 116

Schulze-Bonhage, Andreas 76

van der Gaag, Linda C. 32
van Otterlo, Martijn 145
van Vugt, Marieke 76
Visser, Jelle 145

Wiering, Marco 131, 145
Wieten, Remi 32
Wiggers, Bram 86