

Lecture Notes in Operations Research

Panos Pardalos

Ilias Kotsireas

William J. Knottenbelt

Stefanos Leonardos *Editors*

Mathematical Research for Blockchain Economy

4th International Conference


MARBLE 2023, London, United Kingdom

 Springer

Lecture Notes in Operations Research

Editorial Board Members

Ana Paula Barbosa-Povoa, *University of Lisbon, Lisboa, Portugal*

Adiel Teixeira de Almeida , *Federal University of Pernambuco, Recife, Brazil*

Noah Gans, *The Wharton School, University of Pennsylvania, Philadelphia, USA*

Jatinder N. D. Gupta, *University of Alabama in Huntsville, Huntsville, USA*

Gregory R. Heim, *Mays Business School, Texas A&M University, College Station, USA*

Guowei Hua, *Beijing Jiaotong University, Beijing, China*

Alf Kimms, *University of Duisburg-Essen, Duisburg, Germany*

Xiang Li, *Beijing University of Chemical Technology, Beijing, China*

Hatem Masri, *University of Bahrain, Sakhir, Bahrain*

Stefan Nickel, *Karlsruhe Institute of Technology, Karlsruhe, Germany*

Robin Qiu, *Pennsylvania State University, Malvern, USA*

Ravi Shankar, *Indian Institute of Technology, New Delhi, India*

Roman Slowiński, *Poznań University of Technology, Poznan, Poland*

Christopher S. Tang, *Anderson School, University of California Los Angeles, Los Angeles, USA*

Yuzhe Wu, *Zhejiang University, Hangzhou, China*

Joe Zhu, *Foies Business School, Worcester Polytechnic Institute, Worcester, USA*

Constantin Zopounidis, *Technical University of Crete, Chania, Greece*

Lecture Notes in Operations Research is an interdisciplinary book series which provides a platform for the cutting-edge research and developments in both operations research and operations management field. The purview of this series is global, encompassing all nations and areas of the world.

It comprises for instance, mathematical optimization, mathematical modeling, statistical analysis, queueing theory and other stochastic-process models, Markov decision processes, econometric methods, data envelopment analysis, decision analysis, supply chain management, transportation logistics, process design, operations strategy, facilities planning, production planning and inventory control.

LNOR publishes edited conference proceedings, contributed volumes that present firsthand information on the latest research results and pioneering innovations as well as new perspectives on classical fields. The target audience of LNOR consists of students, researchers as well as industry professionals.


Panos Pardalos · Ilias Kotsireas ·
William J. Knottenbelt · Stefanos Leonardos
Editors


Mathematical Research for Blockchain Economy


4th International Conference MARBLE 2023
London, United Kingdom

Editors

Panos Pardalos 
Department of Industrial and Systems
Engineering
University of Florida
Gainesville, FL, USA

William J. Knottenbelt 
Department of Computing
Imperial College London
London, UK

Ilias Kotsireas 
CARGO Lab
Wilfrid Laurier University
Waterloo, ON, Canada

Stefanos Leonardos 
Department of Informatics
King's College London
London, UK

ISSN 2731-040X

ISSN 2731-0418 (electronic)

Lecture Notes in Operations Research

ISBN 978-3-031-48730-9

ISBN 978-3-031-48731-6 (eBook)

<https://doi.org/10.1007/978-3-031-48731-6>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature
Switzerland AG 2023

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

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

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

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

Paper in this product is recyclable.

Preface

This volume presents the proceedings of the 4th International Conference on Mathematical Research for Blockchain Economy (MARBLE 2023) that was held in London, United Kingdom from July 11 to 13, 2023.

The 4th MARBLE conference took place as an in-person event and featured an exciting programme of research papers, keynote talks and a tutorial, in line with MARBLE's goal to provide a high-profile, cutting-edge platform for mathematicians, computer scientists and economists to present the latest advances and innovations related to the quantitative and economic aspects of blockchain technology. In this context, the Technical Programme Committee has accepted 12 research papers for publication and presentation on themes including mining incentives, game theory, decentralized finance, central government digital coins and stablecoins, automated market makers, blockchain infrastructure and security. The technical programme also features keynotes by the following distinguished speakers: Dr. Garrick Hileman, Tara Annison (Twinstake), Artur Sepp (Clearstar), Mark Morton (Scilling Digital Mining), Dr. Jiahua Xu (University College London & DLT Science Foundation), Thomas Erdösi (CF Benchmarks), Dr. Alexander Freier (Catholic University of Cordoba, University College London and Energiequelle) and Juan Ignacio Ibañez (University College London and DLT Science Foundation).

We thank all authors who submitted their innovative work to MARBLE 2023. In addition, we thank all members of the Technical Programme Committee and other reviewers, everyone who submitted a paper for consideration, the General Chairs, Prof. William Knottenbelt and Prof. Panos Pardalos; the Organization Chair, Jas Gill; the Programme Chairs, My T. Thai and Stefanos Leonardos; the Publication Chair, Ilias Kotsireas; the Web and Publicity Chairs, Kai Sun and Gemma Ralton; and other members of the Centre for Cryptocurrency Research and Engineering who have contributed in many different ways to the organization effort, particularly Katerina Koutsouri. Finally, we are grateful to our primary sponsor, the Brevan Howard Centre for Financial Analysis, for their generous and ongoing support.

London, UK
July 2023

William J. Knottenbelt
Ilias Kotsireas
Stefanos Leonardos
Panos Pardalos

Contents

Deep Reinforcement Learning-Based Rebalancing Policies for Profit Maximization of Relay Nodes in Payment Channel Networks	1
<i>Nikolaos Papadis and Leandros Tassioulas</i>	
Game-Theoretic Randomness for Proof-of-Stake	28
<i>Zhuo Cai and Amir Goharshady</i>	
Incentive Schemes for Rollup Validators	48
<i>Akaki Mamagishvili and Edward W. Felten</i>	
Characterizing Common Quarterly Behaviors in DeFi Lending Protocols	62
<i>Aaron Green, Michael Giannattasio, Keran Wang, John S. Erickson, Oshani Seneviratne, and Kristin P. Bennett</i>	
Blockchain Transaction Censorship: (In)secure and (In)efficient?	78
<i>Zhipeng Wang, Xihan Xiong, and William J. Knottenbelt</i>	
An Automated Market Maker Minimizing Loss-Versus-Rebalancing	95
<i>Conor McMenamin, Vanesa Daza, and Bruno Mazorra</i>	
Profit Lag and Alternate Network Mining	115
<i>Cyril Grunspan and Ricardo Pérez-Marco</i>	
Oracle Counterpoint: Relationships Between On-Chain and Off-Chain Market Data	133
<i>Zhimeng Yang, Ariah Klages-Mundt, and Lewis Gudgeon</i>	
Exploring Decentralized Governance: A Framework Applied to Compound Finance	152
<i>Stamatis Papangelou, Klitos Christodoulou, and George Michoulis</i>	
A Mathematical Approach on the Use of Integer Partitions for Smurfing in Cryptocurrencies	169
<i>Bernhard Garn, Klaus Kieseberg, Ceren Çulha, Marlene Koelbing, and Dimitris E. Simos</i>	
Bigger Than We Thought: The Upbit Hack Gang	178
<i>Qishuang Fu, Dan Lin, and Jiajing Wu</i>	
Staking Pools on Blockchains	187



Deep Reinforcement Learning-Based Rebalancing Policies for Profit Maximization of Relay Nodes in Payment Channel Networks

Nikolaos Papadis¹ and Leandros Tassioulas²

¹ Nokia Bell Labs, Murray Hill, NJ 07974, USA
nikos.papadis@nokia-bell-labs.com

² Department of Electrical Engineering & Yale Institute for Network Science,
Yale University, New Haven, CT 06520, USA
leandros.tassioulas@yale.edu

Abstract. Payment channel networks (PCNs) are a layer-2 blockchain scalability solution, with its main entity, the payment channel, enabling transactions between pairs of nodes “off-chain,” thus reducing the burden on the layer-1 network. Nodes with multiple channels can serve as relays for multihop payments by providing their liquidity and withholding part of the payment amount as a fee. Relay nodes might after a while end up with one or more unbalanced channels, and thus need to trigger a rebalancing operation. In this paper, we study how a relay node can maximize its profits from fees by using the rebalancing method of submarine swaps. We introduce a stochastic model to capture the dynamics of a relay node observing random transaction arrivals and performing occasional rebalancing operations, and express the system evolution as a Markov Decision Process. We formulate the problem of the maximization of the node’s fortune over time over all rebalancing policies, and approximate the optimal solution by designing a Deep Reinforcement Learning (DRL)-based rebalancing policy. We build a discrete event simulator of the system and use it to demonstrate the DRL policy’s superior performance under most conditions by conducting a comparative study of different policies and parameterizations. Our work is the first to introduce DRL for liquidity management in the complex world of PCNs.

Keywords: Payment channel networks · Lightning Network · Rebalancing · Submarine swaps · Deep reinforcement learning · Soft actor-critic · Optimization · Discrete event simulation · Control

Work done while N. Papadis was with the Department of Electrical Engineering and the Yale Institute for Network Science, Yale University, CT 06520, USA.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
P. Pardalos et al. (Eds.): MARBLE 2023, 2024.
https://doi.org/10.1007/978-3-031-48731-6_1

1 Introduction

Blockchain technology enables trusted interactions between untrusted parties, with financial applications like Bitcoin and beyond, but with also known scalability issues [8]. Payment channels are a layer-2 development towards avoiding the long confirmation times and high costs of the layer-1 network: they enable nodes that want to transact quickly, cheaply and privately to do so by depositing some balances to open a payment channel between themselves, and then trustlessly shifting the same total balance between the two sides without broadcasting their transactions and burdening the network. Connected channels create a Payment Channel Network (PCN), via which two nodes not sharing a channel can still pay one another via a sequence of existing channels. Intermediate nodes in the PCN function as relays: they forward the payment along its path and collect relay fees in return. As transactions flow through the PCN, some channels get depleted, causing incoming transactions to fail because of insufficient liquidity on their path. Thus, the need for channel rebalancing arises.

In this paper, we study the rebalancing mechanism of submarine swaps, which allows a blockchain node to exchange funds from on- to off-chain and vice versa. Since a swap involves an on-chain transaction, it takes some time to complete. Taking this into account, we formulate the following optimal rebalancing problem as a Markov Decision Process (MDP): For a node relaying traffic across multiple channels, determine an optimal rebalancing strategy over time (i.e. when and how much to rebalance as a function of the transaction arrival rates observed from an unknown distribution and the confirmation time of an on-chain transaction), so that the node can keep its channels liquid and its profit from relay fees can be maximized.

More specifically, our *contributions* are the following:

- We develop a stochastic model that captures the dynamics of a relay node with two payment channels under two timescales: a continuous one for random discrete transaction arrivals in both directions from distributions unknown to the node, and a discrete one for dispatching rebalancing operations.
- We express the system evolution in our model as an MDP with continuous state and action spaces and time-dependent constraints on the actions, and formulate the problem of relay node profit maximization.
- We approximate the optimal policy of the MDP using Deep Reinforcement Learning (DRL) by appropriately engineering the states, actions and rewards and tuning a version of the Soft Actor-Critic algorithm.
- We develop a discrete event simulator of the system, and use it to evaluate the performance of the learning-based as well as other heuristic rebalancing policies under various transaction arrival conditions and demonstrate the superiority of our policy in a range of regimes.

In summary, our paper is the first to formally study the submarine swap rebalancing mechanism and to introduce a DRL-based method for channel rebalancing in particular, and for PCN liquidity management in general.

2 Background

2.1 Payment Channel Networks and the Need for Rebalancing

A payment channel (Fig. 1) is created between two nodes N_1 and N_2 after they deposit some capital to a channel-opening on-chain transaction. After this transaction is confirmed, the nodes can transact completely *off-chain* (i.e. in the channel) without broadcasting their interactions to the layer-1 network, and without the risk of losing funds, thanks to a cryptographic safety mechanism. The sum of their two balances in the channel remains constant and is called the channel capacity. A transaction of amount α from N_1 to N_2 will succeed if the balance of N_1 at that moment suffices to cover it. In this case, the balance of N_1 is reduced by α and the balance of N_2 is increased by α .

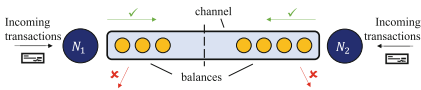


Fig. 1. A payment channel between nodes N_1 and N_2 and current balances of 3 and 4

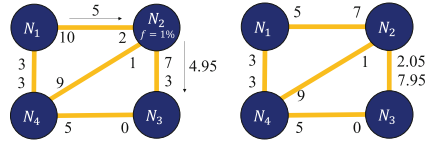


Fig. 2. Processing of a transaction in a payment channel network: before (left) and after (right)

As pairs of nodes create channels, a payment channel network (Fig. 2) is formed, over which multihop payments are possible: Consider a transaction of amount 5 from N_1 to N_3 via N_2 . Note that the amount 5 includes the fees that will have to be paid on the way, e.g. 1% at each intermediate node. In the N_1N_2 channel, N_1 's local balance is reduced by 5 and N_2 's local balance is increased by 5. In the N_2N_3 channel, N_2 's local balance is reduced by $5 - \text{fees} = 4.95$ and N_3 's local balance is increased by 4.95. N_2 's total capital in all its channels before the transaction was $2 + 1 + 7 = 10$, while after it is $7 + 1 + 2.05 = 10.05$, so N_2 made a profit of 0.05 by acting as a relay. If one of the outgoing balances did not suffice, then the transaction would fail end-to-end, thanks to a smart contract mechanism, the Hashed Time-Lock Contract (HTLC). The role of relay nodes is fundamental for the continuous operation of a PCN. The most prominent PCN currently is the Lightning Network [26] built on top of Bitcoin. More details on PCN operation can be found in [23].

Depending on the demand a payment channel is facing in its two directions, funds might accumulate on one side and deplete on the other, due to a combination of factors (see Appendix A for details). The resulting imbalance is undesirable, as it leads to transaction failures and loss of profit from relay fees, thus creating the need for rebalancing mechanisms.

2.2 The Submarine Swap Rebalancing Mechanism

In this work, we study submarine swaps, introduced in [4] and used commercially by Boltz¹ and Loop.² At a high level, a *submarine swap* works as follows (Fig. 3): Node N_1 owns some funds in its channel with node N_2 , and some funds on-chain. At time t_0 , the channel N_1N_2 is almost depleted on N_1 's side (balance = 5). N_1 can start a *swap-in* by paying an amount (50) to a Liquidity Service Provider (LSP)—a wealthy node with access to both layers—via an *on-chain* transaction, and the LSP will give this amount back (reduced by a 10% swap fee, so 45) to N_1 *off-chain* via a path that goes through N_2 . The final amount that is added at N_1 (and subtracted at N_2) is $45 - \epsilon$ due to the relay fees spent on its way from the LSP. Thus, at time t_1 the channel will be almost perfectly balanced. The reverse procedure is also possible (a *reverse submarine swap* or *swap-out*) in order for a node to offload funds from its channel, by paying the server off-chain and receiving funds on-chain. More details on the submarine swap technical protocol can be found in Appendix B.

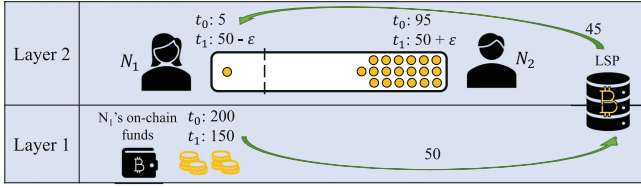


Fig. 3. A submarine swap (swap-in)

The node has to make an important tradeoff: not rebalance a lot to avoid paying swap fees but forfeit relay fees of transactions dropped due to imbalance, or vice versa. This motivates the problem of demand-aware, timely dispatching of swaps by a node aiming to maximize its total fortune.

3 Problem Formulation

3.1 System Evolution

In this section, we introduce a stochastic model of a PCN relay node N that has two channels, one with node L and one with node R , and wishes to maximize its profits from relaying payments from L to R and vice versa (Fig. 4). Let $b_{LN}(\tau)$, $b_{NL}(\tau)$, $b_{NR}(\tau)$, $b_{RN}(\tau)$ be

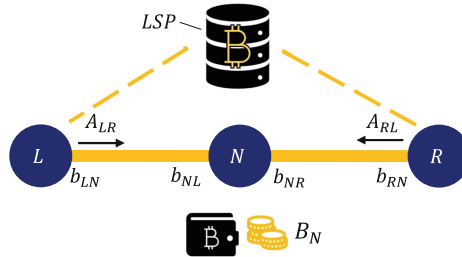


Fig. 4. System model

¹ <https://boltz.exchange>.

² <https://lightning.engineering/loop>.

the channel balances and $B_N(\tau)$ be the on-chain amount of N at time τ . Let C_n be the total capacity of the channel Nn , $n \in \mathcal{N} \triangleq \{L, R\}$. Events happen at two timescales: a continuous one for arriving transactions, and a discrete one for times when the node is allowed to rebalance.

The Transaction Timescale Transactions arrive as a marked point process and are characterized by their direction (L -to- R or R -to- L), time of arrival and amount. We consider node N to not be the source or destination of any transactions itself, but rather to only act as a relay. At each moment in continuous time (denoted by τ), (at most) one transaction arrives in the system. All transactions are admitted, but some fail due to insufficient balances.

Let $f(\alpha)$ be the fee that a transaction of amount α pays to a node that relays it. We assume all nodes charge the same fees. f can be any fixed function with $f(0) = 0$. In practice, for $\alpha > 0$, $f(\alpha) = f_{\text{base}} + f_{\text{prop}} \cdot \alpha$, where the base fee f_{base} and the proportional fee f_{prop} are constants. Let $A_{LR}(\tau)$, $A_{RL}(\tau)$ be the externally arriving amounts coming from node L in the L -to- R direction and from node R in the R -to- L direction at time τ respectively, each drawn from a distribution that is fixed but unknown to node N . An arriving transaction of amount $A_{LR}(\tau) = \alpha$ is feasible if and only if there is enough balance in the L -to- R direction in both channels, i.e. $b_{LN}(\tau) \geq \alpha$ and $b_{NR} \geq \alpha - f(\alpha)$, and similarly for the R -to- L direction. The successfully processed amounts by N at time τ are³:

$$S_{LR}(\tau) = \begin{cases} A_{LR}(\tau) & , \text{ if } A_{LR}(\tau) \leq b_{LN}(\tau) \text{ and } A_{LR}(\tau) - f(A_{LR}(\tau)) \leq b_{NR}(\tau) \\ 0 & , \text{ otherwise} \end{cases}$$

and symmetrically for $S_{RL}(\tau)$.

The profit of node N at time τ is $f(S_{LR}(\tau)) + f(S_{RL}(\tau))$, and the lost fees (from transactions that potentially failed to process) are $f(A_{LR}(\tau) - S_{LR}(\tau)) + f(A_{RL}(\tau) - S_{RL}(\tau))$. The balance processes at time τ evolve as follows (the on-chain amount $B_N(\tau)$ is not affected by the processing of off-chain transactions; channel NR behaves symmetrically):

$$\begin{aligned} b_{LN}(\tau) &\rightarrow b_{LN}(\tau) + (S_{RL}(\tau) - f(S_{RL}(\tau))) - S_{LR}(\tau) \\ b_{NL}(\tau) &\rightarrow b_{NL}(\tau) + S_{LR}(\tau) - (S_{RL}(\tau) - f(S_{RL}(\tau))) \end{aligned}$$

The Rebalancing Decision (Control) Timescale The evolution of the system can be controlled by node N using submarine swap rebalancing operations. Rebalancing may start at times $t_i = i \cdot T_{\text{check}}$, $i = 0, 1, \dots$, and takes a (fixed) time T_{conf} to complete (on average 10 min for Bitcoin). We consider the case

³ Since in the sequel we focus on the discrete and sparse time scale of the periodic times at which the node rebalances, we make the fair assumption (as e.g. in [2]) that off-chain transactions are processed instantaneously across their entire path and do not fail in their subsequent steps after they cross the two channels (if a transaction were to fail outside the two channels, it can be viewed as of zero value by the system).

where $T_{\text{check}} \geq T_{\text{conf}}$ (to avoid having concurrent rebalancing operations in the same channel that could be combined into one). The system state is defined only for the discrete rebalancing decision timescale as the collection of the off- and on-chain balances:

$$S(t_i) = (b_{LN}(t_i), b_{NL}(t_i), b_{NR}(t_i), b_{RN}(t_i), B_N(t_i)) \quad (1)$$

At each time t_i , node N can decide to request a swap-in or a swap-out in each channel. Call the respective amounts $r_L^{\text{in}}(t_i), r_L^{\text{out}}(t_i), r_R^{\text{in}}(t_i), r_R^{\text{out}}(t_i)$. At any time t_i , in a given channel, either a swap-in or a swap-out or nothing will be requested, but not both a swap-in and a swap-out.⁴

Let $F_{\text{swap}}^{\text{in}}(\alpha)$ and $F_{\text{swap}}^{\text{out}}(\alpha)$ be the swap fees that the LSP charges for an amount α for a swap-in and a swap-out respectively, where $F_{\text{swap}}^{\text{in}}(\cdot)$ and $F_{\text{swap}}^{\text{out}}(\cdot)$ are any nonnegative functions with $F_{\text{swap}}(0) = 0$. For ease of exposition, we let all types of fees the node will have to pay (relay fees for the off-chain part, on-chain miner fees, server fees) be both part of the above swap fees, and be the same for both swap-ins and swap-outs when a net amount r_{net} is transferred from on- to off-chain or vice versa: $F_{\text{swap}}^{\text{in}}(r_{\text{net}}) = F_{\text{swap}}^{\text{out}}(r_{\text{net}}) = F_{\text{swap}}(r_{\text{net}}) \triangleq r_{\text{net}}F + M$, where the proportional part F includes the server fee and off-chain relay fees, and M includes the miner fee and potential base fees.

Note that the semantics of the swap amounts r are such that they represent the amount that will move *in the channel* (and not necessarily the net change in the node's fortune). As a result of this convention and based on the swap operation as described in the following paragraph, the amount r^{in} of a swap-in coincides with the net amount $r_{\text{net}}^{\text{in}}$ by which the node's fortune decreases (as r^{in} does not include the swap fee), while the amount r^{out} of a swap-out includes the swap fee and the net amount by which the node's fortune decreases is $r_{\text{net}}^{\text{out}} = \phi^{-1}(r^{\text{out}})$, where $\phi(r_{\text{net}}) \triangleq r_{\text{net}} + F_{\text{swap}}(r_{\text{net}})$, and ϕ^{-1} is the generalized inverse function of $\phi(\cdot)$ (it always exists: $\phi^{-1}(y) = \min\{x \in \mathbb{N} : \phi(x) = y\}$). For our $F_{\text{swap}}(\cdot)$ it is $\phi(r_{\text{net}}) = r_{\text{net}}(1 + F) + M$ for $r_{\text{net}} > 0$, $\phi(0) = 0$, so $\phi^{-1}(y) = (y - M)/(1 + F)$ for $y > 0$ and $\phi^{-1}(0) = 0$.

A Submarine Swap Step-by-Step We now describe how a rebalancing operation on the Nn channel is affecting the system state. First, we describe a swap-in of amount r_n^{in} initiated by node N to refill N 's local balance in the Nn channel:

- At time t_i , node N locks the net rebalancing amount plus fees and subtracts it from its on-chain funds: $B_N \rightarrow B_N - (r_n^{\text{in}} + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}))$
- At time $t_i + T_{\text{conf}}$, the on-chain transaction is confirmed, so the LSP sends a payment of r_n^{in} to node N off-chain.⁵ The payment reaches node n :
 - If $b_{nN} \leq r_n^{\text{in}}$ (i.e. n does not have enough balance to forward it), then the off-chain payment fails. The on-chain funds are unlocked and refunded back to the on-chain amount: $B_N \rightarrow B_N + (r_n^{\text{in}} + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}))$

⁴ Nodes L and R are considered passive: they perform no swap operations themselves.

⁵ The LSP is a well-connected node owning large amounts of liquidity, so we reasonably assume that it can always find a route from itself to N , possibly via splitting the amount across multiple paths.

- Otherwise (if the transaction is feasible), n forwards the payment to N : $b_{nN} \rightarrow b_{nN} - r_n^{\text{in}}$ and $b_{Nn} = b_{Nn} + r_n^{\text{in}}$

A swap-out of amount r_n^{out} , initiated by node N to offload some of its local balance to the chain, works as follows:

- At time t_i , node N locks the net rebalancing amount plus fees and sends it to the LSP via the off-chain network: $b_{Nn} \rightarrow b_{Nn} - r_n^{\text{out}}$. Note that r_n^{out} includes the fees.
- At time $t_i + T_{\text{conf}}$, the on-chain transaction is confirmed, so node N receives the funds on-chain: $B_N \rightarrow B_N + \phi^{-1}(r_n^{\text{out}})$, and the funds are also unlocked in the channel and pushed towards the remote balance: $b_{nN} \rightarrow b_{nN} + r_n^{\text{out}}$

Rebalancing Constraints Based on the steps just described, swap operations will succeed if and only if their amounts satisfy the following constraints:

- Rebalancing amounts must be non-negative:

$$r_n^{\text{in}}(t_i), r_n^{\text{out}}(t_i) \geq 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \quad (2)$$

- A swap-in and a swap-out cannot be requested in the same channel at the same time:

$$r_n^{\text{in}}(t_i) \cdot r_n^{\text{out}}(t_i) = 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \quad (3)$$

- The swap-out amounts (which already include the swap fees) must be greater than the fees themselves:

$$r_n^{\text{out}}(t_i) - F_{\text{swap}}^{\text{out}}(r_n^{\text{out}}(t_i)) \geq 0 \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \quad (4)$$

- The respective channel balances must suffice to cover the swap-out amounts (which already include the swap fees):

$$r_n^{\text{out}}(t_i) \leq b_{Nn}(t_i) \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \quad (5)$$

- The on-chain balance must suffice to cover the total swap-in amount plus fees:

$$\sum_{n \in \mathcal{N}} (r_n^{\text{in}}(t_i) + F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}(t_i))) \leq B_N(t_i) \text{ for all } i \in \mathbb{N} \quad (6)$$

State Evolution Equations Now we are able to write the complete state evolution equations. The amounts added to each balance due to successful transactions during the interval (t_i, t_{i+1}) are:

$$d_{NL}^{(t_i, t_{i+1})} \triangleq \int_{\tau \in (t_i, t_{i+1})} \left(S_{LR}(\tau) - (S_{RL}(\tau) - f(S_{RL}(\tau))) \right) d\tau,$$

similarly for $d_{NR}^{(t_i, t_{i+1})}$, and $d_{nN}^{(t_i, t_{i+1})} \triangleq -d_{Nn}^{(t_i, t_{i+1})}$. Then for actions taken subject to the constraints (2)–(6), the state evolves as follows:

$$\begin{aligned} b_{nN}(t_{i+1}) &= b_{nN}(t_i) + d_{nN}^{(t_i, t_{i+1})} - (r_n^{\text{in}}(t_i) - z_n(t_i)) + r_n^{\text{out}}(t_i) \\ b_{Nn}(t_{i+1}) &= b_{Nn}(t_i) + d_{Nn}^{(t_i, t_{i+1})} + (r_n^{\text{in}}(t_i) - z_n(t_i)) - r_n^{\text{out}}(t_i) \\ B_N(t_{i+1}) &= B_N(t_i) - \sum_{n \in \mathcal{N}} \left(r_n^{\text{in}}(t_i) - F_{\text{swap}}^{\text{in}}(r_n^{\text{in}}(t_i)) \right) + \sum_{n \in \mathcal{N}} \phi^{-1}(r_n^{\text{out}}(t_i)) + \sum_{n \in \mathcal{N}} w_n(t_i) \end{aligned}$$

where $z_n(t_i)$ and $w_n(t_i)$ are the refunds of the swap-in amount off- and on-chain respectively in case a swap-in operation fails:

$$z_n(t_i) = r_n^{\text{in}}(t_i) \mathbb{1}\{b_{nN}(t_i) + d_{nN}^{(t_i, t_i + T_{\text{conf}})} < r_n^{\text{in}}(t_i)\} \quad (7)$$

$$w_n(t_i) = z_n(t_i) + F_{\text{swap}}^{\text{in}}(z_n(t_i)) \quad (8)$$

3.2 Writing the Problem as a Markov Decision Process

The objective function the node wishes to maximize in the real world is its total fortune both in the channels and on-chain (another equivalent objective is discussed in Appendix C). The fortune increase due to the action (the 4-tuple $r(t_i)$ taken at step t_i is:

$$D(t_i, r(t_i)) \triangleq \left(\sum_{n \in \mathcal{N}} b_{Nn}(t_{i+1}) + B_N(t_{i+1}) \right) - \left(\sum_{n \in \mathcal{N}} b_{Nn}(t_i) + B_N(t_i) \right)$$

A control policy $\pi = \{(t_i, r^\pi(t_i))\}_{i \in \mathbb{N}}$ consists of the times t_i and the corresponding actions $r^\pi(t_i) = (r_L^{\text{in}}(t_i), r_L^{\text{out}}(t_i), r_R^{\text{in}}(t_i), r_R^{\text{out}}(t_i))$ taken from the set of allowed actions $\mathcal{R} = [0, C_L]^2 \times [0, C_R]^2$, and belongs to the set of admissible policies

$$\Pi = \{ \{(t_i, r(t_i))\}_{i \in \mathbb{N}} \text{ such that } r(t_i) \in \mathcal{R} \text{ for all } i \in \mathbb{N} \}$$

Ultimately, the goal of node N is to find a rebalancing policy that maximizes the long-term average expected fortune increase D over all admissible policies:

$$\text{maximize}_{\pi \in \Pi} \lim_{H \rightarrow \infty} \frac{1}{t_H} \sum_{i=0}^H \mathbb{E} [D(t_i, r^\pi(t_i))]$$

subject to (2)–(6).

4 Heuristic and Reinforcement Learning-Based Policies

In this section, we describe the steps we took in order to apply DRL to approximately solve the formulated MDP. We first outline two heuristic policies, which we will use later to benchmark our DRL-based solution.

4.1 Heuristic Policies

Autoloop [7, 18] is a policy that allows a node to schedule automatic swap-ins (resp. swap-outs) if its local balance falls below a minimum (resp. rises above a maximum) threshold expressed as a percentage of the channel’s capacity.⁶ The initiated swap is of amount equal to the difference of the local balance from the midpoint, i.e. the average of the two thresholds. The pseudocode can be found in Algorithm 1. We expect Autoloop to be suboptimal with respect to profit maximization in certain cases, as it does not take the expected demand into account and thus possibly performs rebalancing at times when it is not necessary.

Algorithm 1: Autoloop rebalancing policy

Input: *state* as in Eq. (1)
Parameters: T_{check} , *low*, *high*

```

1 every  $T_{\text{check}}$  do
2   foreach neighbor  $n \in \mathcal{N}$  do
3      $\text{midpoint} = C_n \cdot (\text{low} + \text{high})/2$ 
4     if  $b_{Nn} < \text{low} \cdot C_n$  then
5       | Swap-in amount =  $\text{midpoint} - b_{Nn}$ 
6     else if  $b_{Nn} > \text{high} \cdot C_n$  then
7       | Swap-out amount =  $b_{Nn} - \text{midpoint}$ 
8     else
9       | Perform no action

```

This motivates us to define another heuristic policy that incorporates the empirical demand information. We call this policy Loopmax, as its goal is to rebalance with the maximum possible amount and as infrequently as possible (without sacrificing transactions), based on the demand. Loopmax keeps track of the total arriving amounts, and estimates the net change of each balance per unit time using the difference of the total amounts that arrived in each direction:

$$\hat{A}_{LN}^{\text{net}}(\tau) = -\hat{A}_{NL}^{\text{net}}(\tau) \triangleq \frac{1}{\tau} \int_{t \in [0, \tau]} \left(A_{RL}(t) - f(A_{RL}(t)) - A_{LR}(t) \right) dt \quad (9)$$

$$\hat{A}_{RN}^{\text{net}}(\tau) = -\hat{A}_{NR}^{\text{net}}(\tau) \triangleq \frac{1}{\tau} \int_{t \in [0, \tau]} \left(A_{LR}(t) - f(A_{LR}(t)) - A_{RL}(t) \right) dt \quad (10)$$

For each channel, we first calculate its estimated time to depletion (*ETTD*) or saturation (*ETTS*), depending on the direction of the net demand and the current balances, and using this time we dispatch a swap of the appropriate type not earlier than $T_{\text{check}} + T_{\text{conf}}$ before depletion/saturation, and of the maximum

⁶ The original Autoloop algorithm defines the thresholds in terms of the node’s inbound liquidity in a channel. We adopt an equivalent balance-centric view instead.

possible amount. The rationale is that if e.g. $ETTD \geq T_{\text{check}} + T_{\text{conf}}$, the policy can leverage this fact to postpone starting a swap until the next check time, since until then no transactions will have been dropped. If $ETTD < T_{\text{check}} + T_{\text{conf}}$ though, the policy should act now, as otherwise it will end up dropping transactions during the following interval of $T_{\text{check}} + T_{\text{conf}}$. The maximum possible swap-out is constrained by the local balance at that time, while the maximum possible swap-in is constrained by the remote balance at that time⁷ and the on-chain amount: an on-chain amount of B_N can support (by including fees) a net swap-in amount of at most $\phi^{-1}(B_N)$. The pseudocode can be found in Algorithm 2. Compared to Autoloop, Loopmax has the advantage that it rebalances only when it is absolutely necessary and can thus achieve savings in swap fees. On the other hand, Loopmax's aggressiveness can lead it to extreme rebalancing decisions when traffic is quite skewed in a particular direction (e.g. it can do a swap-in of almost the full capacity, which is very likely to fail due to randomness in the transaction arrivals). A small modification we can use on top of Algorithm 2 to alleviate this is to define certain safety margins of liquidity that Loopmax should always leave intact on each side of the channel, so that incoming transactions do not find it depleted due to a large pending swap.

Algorithm 2: Loopmax rebalancing policy

Input: *state* as in Eq. (1)

Parameters: T_{check}

```

1 every  $T_{\text{check}}$  do
2   Update  $\{\hat{A}_{Nn}^{\text{net}}\}_{n \in \mathcal{N}}$  according to Eqs. (9)–(10)
3   foreach neighbor  $n \in \mathcal{N}$  do
4     if  $\hat{A}_{Nn}^{\text{net}} < 0$  then
5        $ETTD = b_{Nn} / |\hat{A}_{Nn}^{\text{net}}|$  /* estimated time to depletion */
6       if  $ETTD < T_{\text{check}} + T_{\text{conf}}$  then
7         Swap-in amount =  $\max\{\phi^{-1}(B_N), b_{nN}\}$  /* maximum possible
8         swap in */
9       else
10        Perform no action
11    else if  $\hat{A}_{Nn}^{\text{net}} > 0$  then
12       $ETTS = b_{nN} / \hat{A}_{Nn}^{\text{net}}$  /* estimated time to saturation */
13      if  $ETTS < T_{\text{check}} + T_{\text{conf}}$  then
14        Swap-out amount =  $b_{Nn}$  /* maximum possible swap out */
15      else
16        Perform no action
17    else
18      Perform no action
  
```

⁷ Actually, it is constrained by the remote balance at the time of the swap-in's completion. We will improve this later using estimates of future balances.

4.2 Deep Reinforcement Learning Algorithm Design

Having formulated the problem as an MDP, we now need to find an (approximately) optimal policy. The problem is challenging for a number of reasons:

- The problem dynamics are not linear.
- The state and action spaces are continuous and thus tabular approaches are not applicable.
- There are time-dependent constraints on the actions.
- Choosing to not rebalance at a specific time requires special treatment, as otherwise the zero action will be sampled from a continuous action space with zero probability.

To tackle these challenges, we resort to approximate methods, and specifically Reinforcement Learning (RL). In the standard RL framework, an agent makes decisions based on a policy represented as a probability distribution over states and actions: $p : p(s, a) \rightarrow [0, 1]$, with $p(s, a)$ being the probability that action a will be taken when the environment is in state s . Since our problem has continuous state and action spaces and the policy cannot be stored in tabular form, we need to use function approximation techniques. Neural networks serve well the role of function approximators in many applications [5, 20]. Some algorithms appropriate for this type of problems are Deep Deterministic Policy Gradient (DDPG) [19] and Soft Actor-Critic (SAC) [15]. We decided to use the latter as DDPG is known to exhibit extreme brittleness and hyperparameter sensitivity [11]. We now describe our methodology around how we engineer our DRL algorithm based on the vanilla SAC in order to arrive at a solution that deals with all the above challenges.

For the RL agent’s environment, we use as state the five balances (off- and on-chain) and the estimates of the remote balances at the time of the swap completion, each normalized appropriately: by the respective channel’s capacity, or by a total target fortune in the on-chain amount’s case. Thus, our state space is $[0, 1]^7$. As actions, instead of the 4-tuple of Sect. 3, we use an equivalent (due to (3)) 2-tuple (r_L, r_R) , i.e. a single variable for each channel that can take both positive (swap-in) and negative (swap-out) values. Before the raw sampled action is applied, it undergoes some processing described in the sequel.

Raw actions are sampled from the entire continuous action space; thus the zero action will be selected with zero probability. In reality, though, performing zero rebalancing in a channel when a swap is not necessary is important for minimizing the costs, and an action the agent should learn to apply. To make the zero action selectable with positive probability, and at the same time prevent the agent from performing swaps too small in size, we force the respective applied action to be zero if the raw action coordinate is less than a threshold ρ_0 (e.g. 20%) of the channel capacity. Moreover, the vanilla SAC algorithm [15] operates on an action space that is a compact subset of \mathbb{R}^k for all decision times. In our case, though, the allowed actions vary due to the time-dependent constraints (2)–(6). We therefore define the action space to be $[-1, 1]^2$, where each coordinate denotes the percentage not of the entire channel capacity, but of the maximum

amount available for the respective type of swap at that moment. We now focus on deriving these maximum amounts from the constraints.

All constraints are decoupled per channel, except for (6). However, we observe that given some traffic, mostly in the L -to- R direction or mostly in the R -to- L direction or equal in both directions, the local balances of node N will either deplete in one channel and accumulate in the other, or accumulate in both, but never both deplete. Thus, a swap-in in both channels in general will not be a good action. Therefore, for the RL solution’s purposes we can split (6) into two constraints, one for each channel, with the right-hand side of each being the entire amount $B_N(t_i)$. In case the agent does take the not advisable decision of swap-ins in both channels and their sum exceeds the on-chain amount, one of the two will simply fail.

Another useful observation is that when a swap-in is about to complete time T_{conf} after it was requested, the remote balance in the respective channel needs to suffice (otherwise the swap-in will fail and a refund will be triggered as in Eqs. (7)–(8)):

$$r_n^{\text{in}}(t_i) \leq b_{nN}(t_i) + d_{nN}^{(t_i, t_i + T_{\text{conf}})} \text{ for all } i \in \mathbb{N}, n \in \mathcal{N} \quad (11)$$

We calculate an estimate $\hat{b}_{nN}(t_i + T_{\text{conf}})$ of the right-hand side of (11) based on the past history, with the details of the calculation given in Appendix D.1. Let $\rho_{\text{min}}^{\text{out}} \triangleq M/(1 - F)$ be the minimum solution of (4). As long as $\rho_0 C_n \gg \rho_{\text{min}}^{\text{out}}$, which should hold in practice as $\rho_{\text{min}}^{\text{out}}$ is very small, we can write all constraints (2)–(6), (11) in terms of the 2-tuple (r_L, r_R) as follows:

$$r_n \in \left[-b_{nN}, \min\{\hat{b}_{nN}(t_i + T_{\text{conf}}), \phi^{-1}(B_N(t_i)), C_n\} \right], n \in \mathcal{N}$$

The described mapping of raw actions (sampled from the distribution on the entire action space) to the finally applied actions is summarized in Table 1.

Table 1. Mapping of raw actions sampled from the learned distribution to final swap amounts requested for channel Nn , $n \in \mathcal{N}$

Raw action $r_n \in [-1, 1]$	Corresponding absolute amount \tilde{r}_n	Final requested swap amount
$r_n < 0$	$ r_n b_{nN}$	Swap out $\tilde{r}_n \mathbb{1}\{\tilde{r}_n \geq \rho_0 C_n\}$
$r_n \geq 0$	$r_n \min\{\hat{b}_{nN}(t_i + T_{\text{conf}}), \phi^{-1}(B_N(t_i)), C_n\}$	Swap in $\tilde{r}_n \mathbb{1}\{\tilde{r}_n > \rho_0 C_n\}$