Optimizing Virtual Channels in Payment Channel Networks

Network Architecture Project

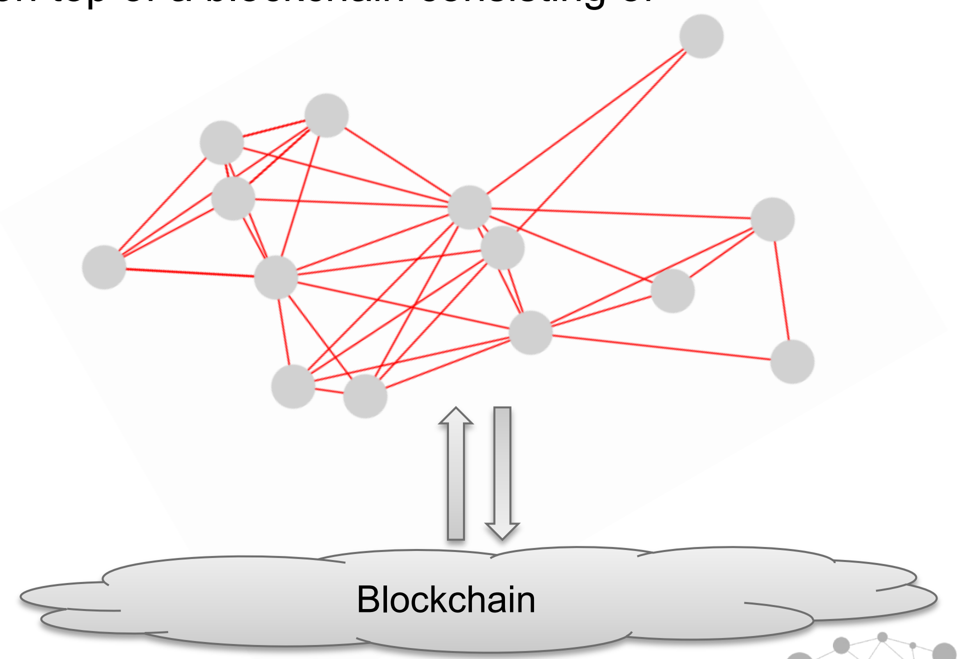
Yannik Kopyciok, supervisor: Iosif Salem

Payment Channel Network

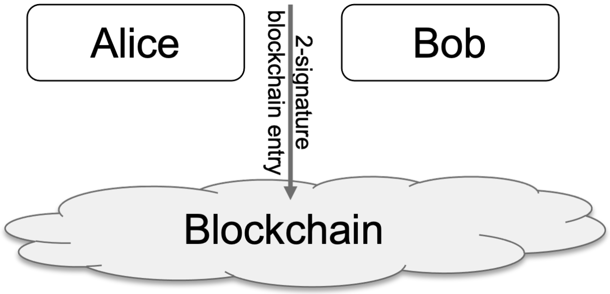
Decentralized second network layer on top of a blockchain consisting of various payment channels.

Off-chain transactions

→ Scalable, fast, and cost efficient



Why is not used by everyone?



Establishment of PC locks funds on the blockchain

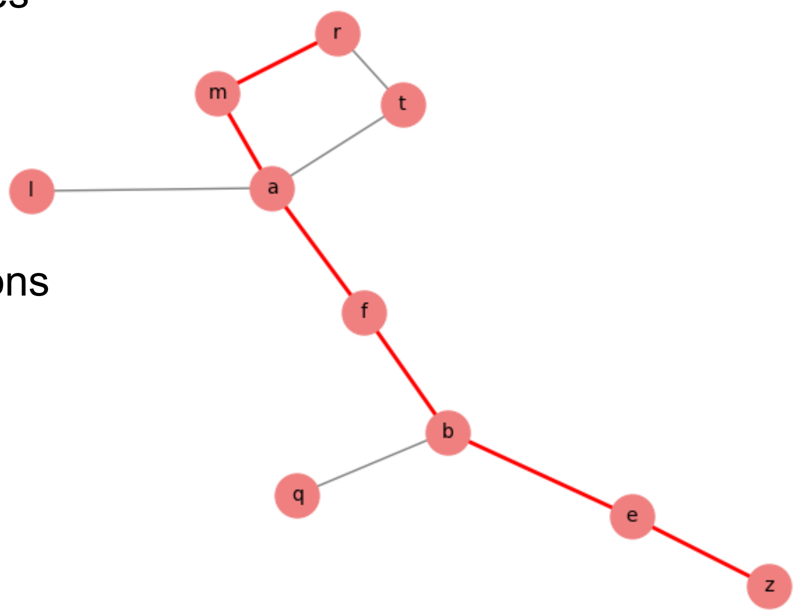
→ Determines capacity of the PC

Potentially multiple intermediates

→ Routing fees of a path

→ Possible security issues

- Data hoarding
- Transaction cancellations



Virtual Channels

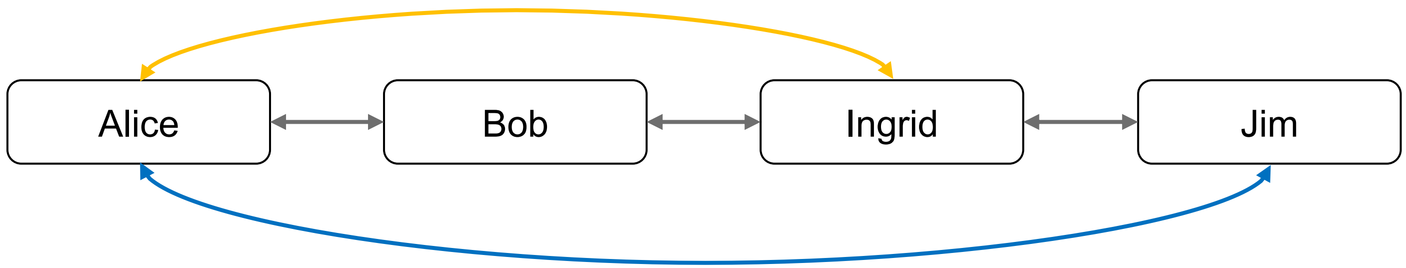
Bridge over one or multiple nodes

→ "as if they have a direct PC"

→ Reduced latency

→ Routing fees avoided

→ Payment details stay secret



Why is not used by everyone?

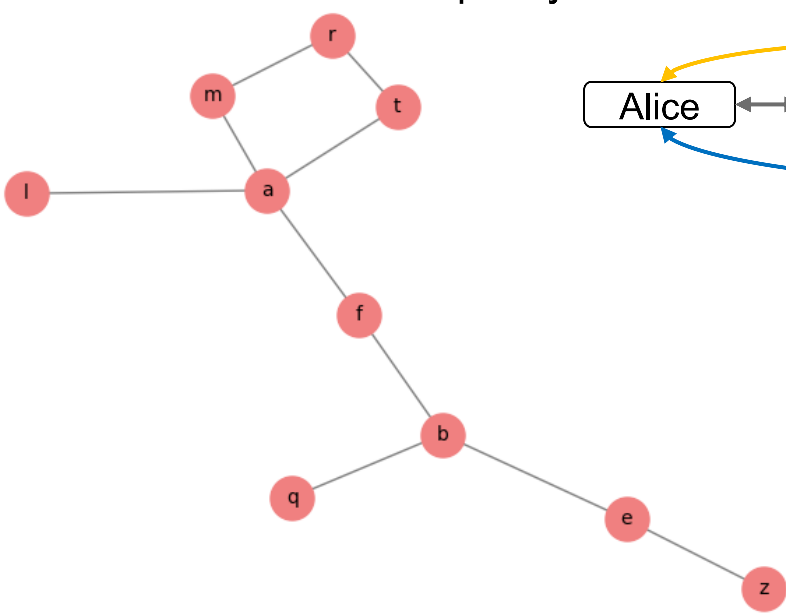
Concept of VCs is a fairly new research topic

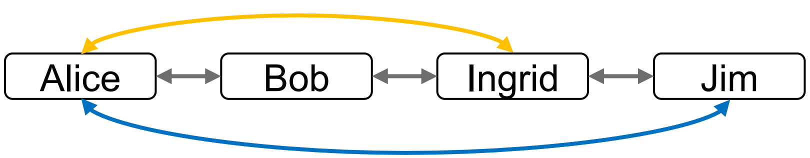
Establishment requires the collaboration of **all** intermediates

Collaboration comes at a cost

→ base creation cost and

→ proportional cost in relation to the capacity





What VC creation strategy to follow?

NP-hard problem of what VCs should be created when

→ Proof in a yet unpublished paper which presents

- → an empirical solution
- → and an Integer Linear Program in theory

Goal of the project is to find an exact and optimal solution by building this ILP.

VC optimization meets GurobiPy

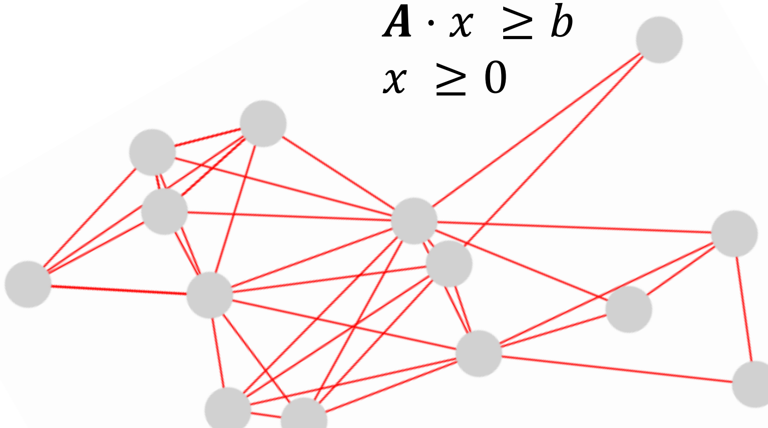
GUROBI OPTIMIZATION* state of the art linear optimization problem solver

Objective: Minimization of fees

C1: Transaction path uniqueness
C2: Transaction success rate
C3: Capacity restriction
C4: VC existence
C5: VC capacity
C6: Known adversaries

ILP in matrix form:

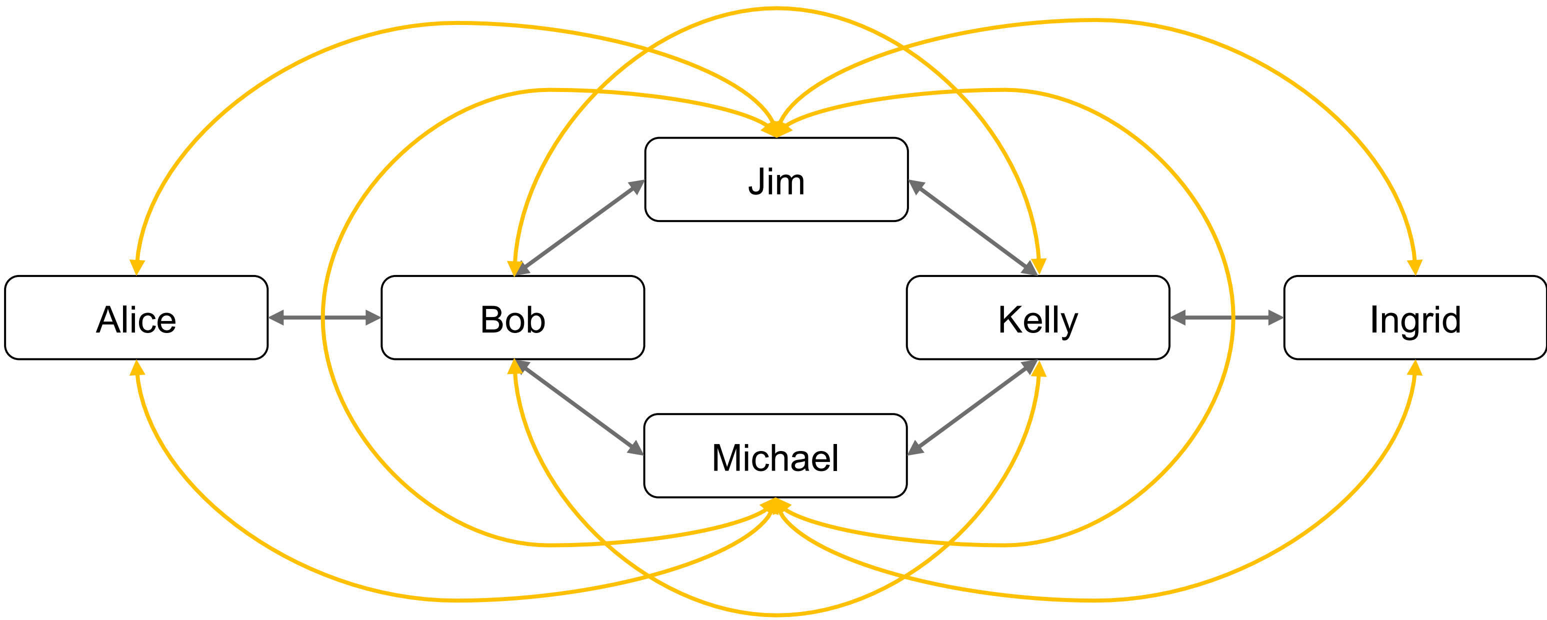
$$\begin{aligned} &\min \boldsymbol{c}^T \cdot x \\ &\text{such that} \\ &\boldsymbol{A} \cdot x \geq b \\ &x \geq 0 \end{aligned}$$



*https://www.gurobi.com

Input - Example

Baseline
Graph(6, 6)
5 transactions
Paths = 10



Level 0
Graph(6, 14)
5 transactions
VC = 8
Paths = 152

Technische Universität Berlin

Integer Linear Program – Example

$$\min \boldsymbol{c}^T \cdot x \quad \text{such that} \quad \boldsymbol{A} \cdot x \geq b, \quad x \geq 0$$

Objective: Costminimization
C1: Transaction path uniqueness
C2: Transaction success rate
C3: Capacity restriction
C4: VC existence
C5: VC capacity
C6: Known adversaries

Objective
x-vector
c^T

[BINARY … BINARY BINARY … BINARY INTEGER … INTEGER]

path_variable *vc_existence* *vc_capacity*

[0. 0. 1. 0. 0. 0. 0. … 0. 0. 0. 0. 1. ... 0. 0. 0. ... 0. 400. 0.]

[routing_fee(path) … vc.base_fee ... vc.prop_fee]

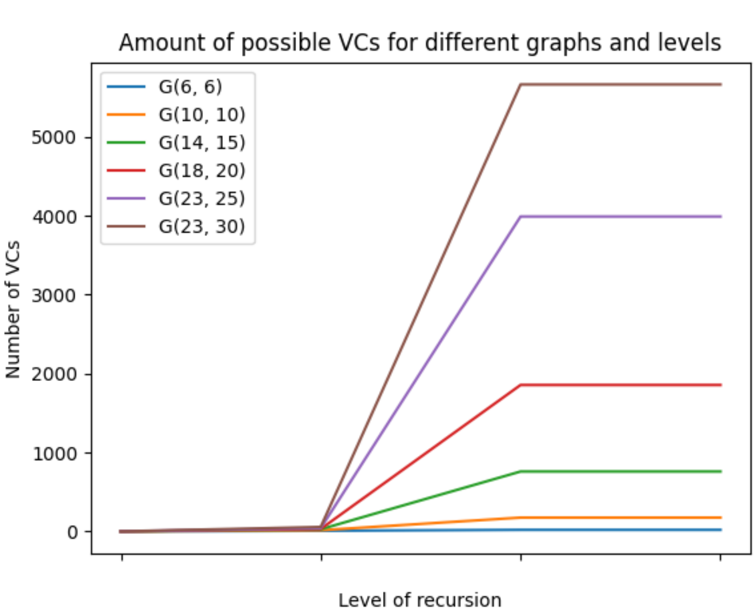
Constraints
A-matrix (sparse)

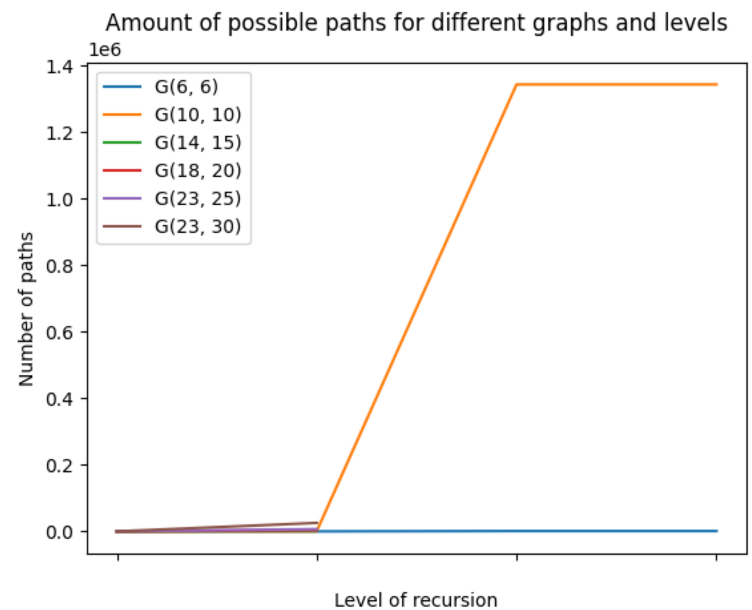
$$
\begin{array}{cc}
(C1\,, path; vc; vc) & -1 \\
\vdots\ T - times & \vdots \\
(C2\,, path; vc; vc) & 1 \\
(C3\,, path; vc; vc) & -(trans + fees); base; prop \\
\vdots\ E\ - times & \vdots \\
(C4\,, path; vc; vc) & -1 \\
\vdots\ VC\ - times & \vdots \\
(C5\,, path; vc; vc) & -(trans + fees); 1 \\
\vdots\ VC\ - times & \vdots \\
(C6\,, path; vc; vc) & 0
\end{array}
\qquad
\begin{bmatrix}
-1 \\
\vdots \\
c_{tr} * T \\
-pc.capacity \\
\vdots \\
0 \\
\vdots \\
0 \\
\vdots \\
0
\end{bmatrix}
$$

Constraints
b/rhs-vector

INET

Challenge



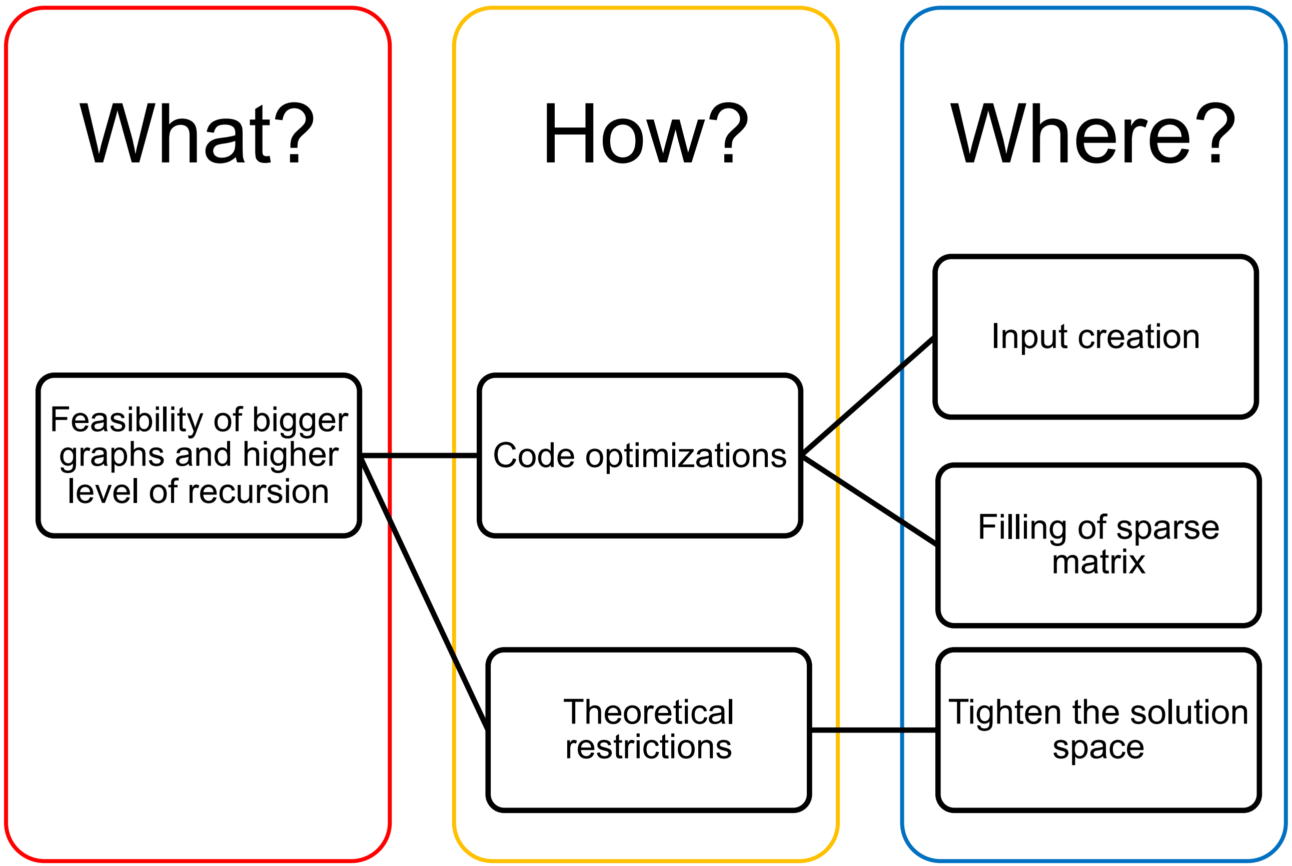


Graph(6, 6) Level 1
VCs = 20
Paths = 1,116

Graph(10, 10) Level 1
VCs = 175
Paths = 1,343,227

LN(8k, 33k)

Next steps



What?
How?
Where?
Feasibility of bigger graphs and higher level of recursion
Code optimizations
Theoretical restrictions
Input creation
Filling of sparse matrix
Tighten the solution space

Questions?