# *ReViNE*: Reallocation of Virtual Network Embedding to Eliminate Substrate Bottleneck

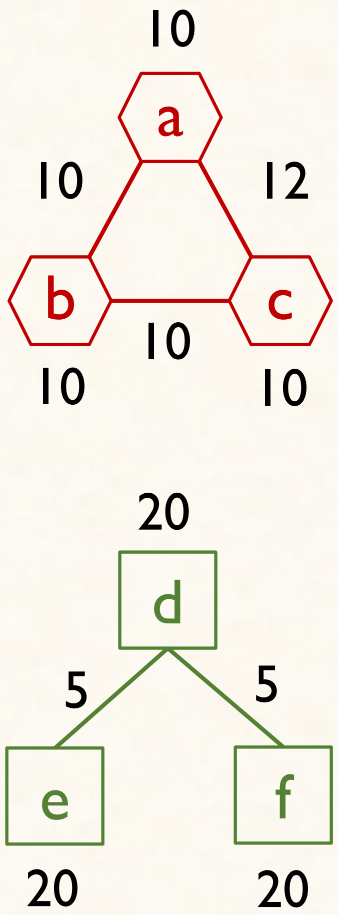
Shihabur R. Chowdhury,
Reaz Ahmed, Nashid Shahriar,
Aimal Khan, Raouf Boutaba

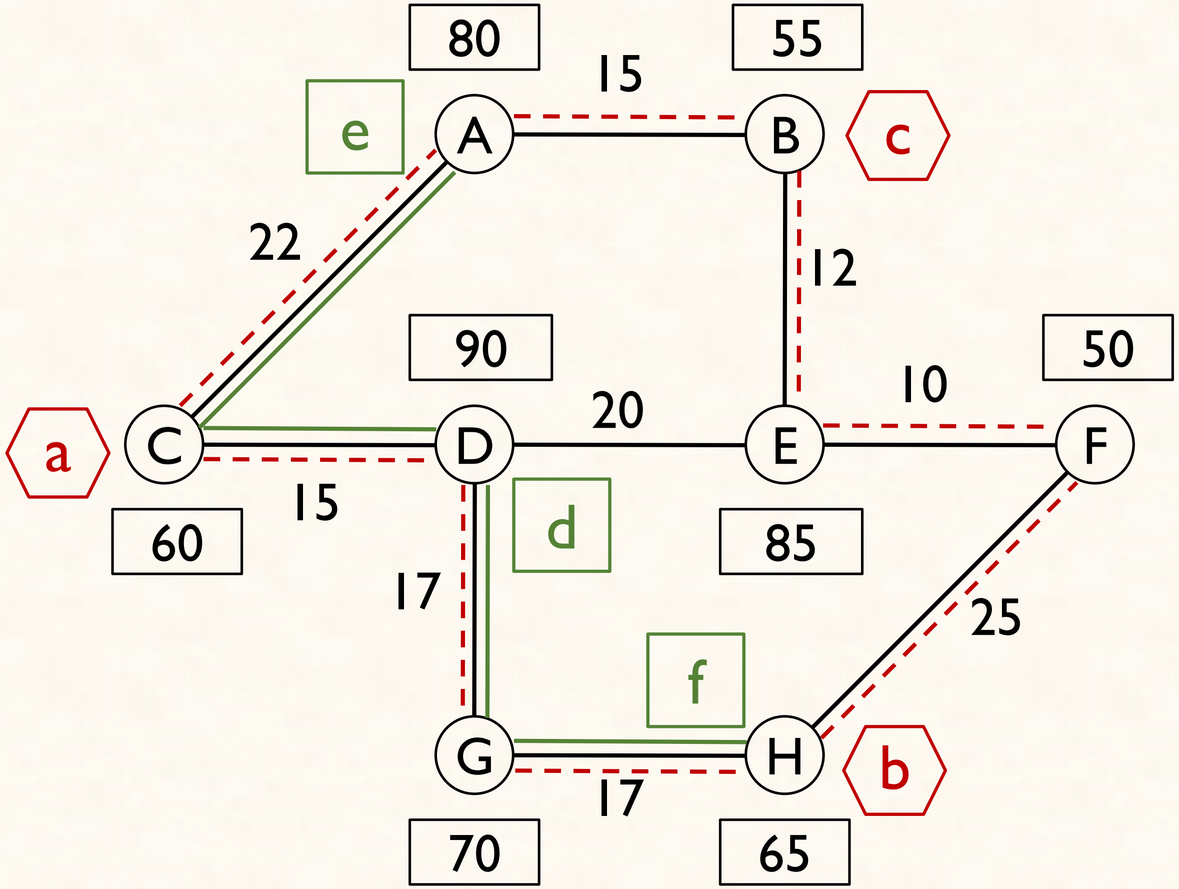
UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
David R. Cheriton School
of Computer Science

Jeebak Mitra,
Liu Liu

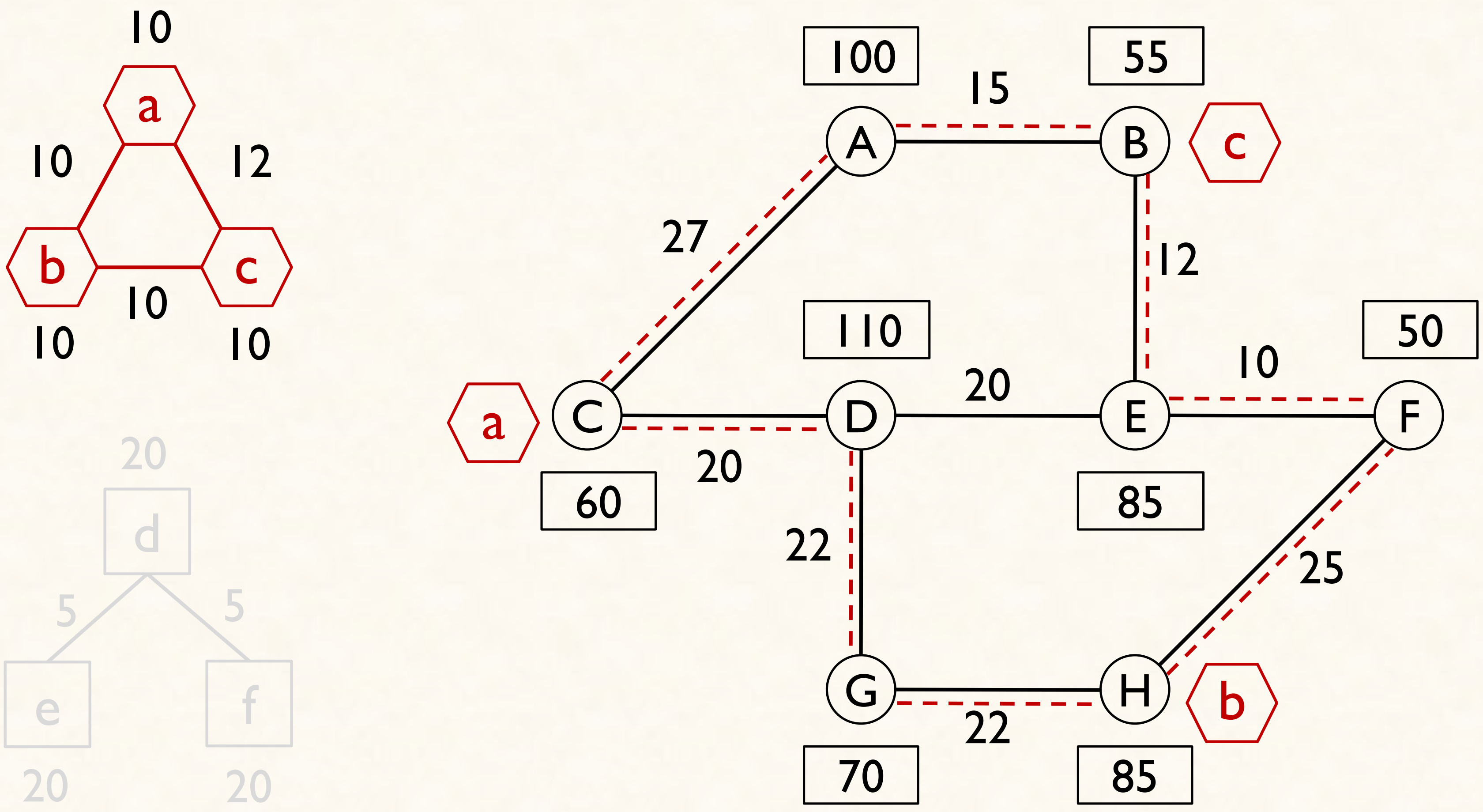
HUAWEI

# Virtual Network Embedding (VNE)





# Virtual Network Embedding (VNE)



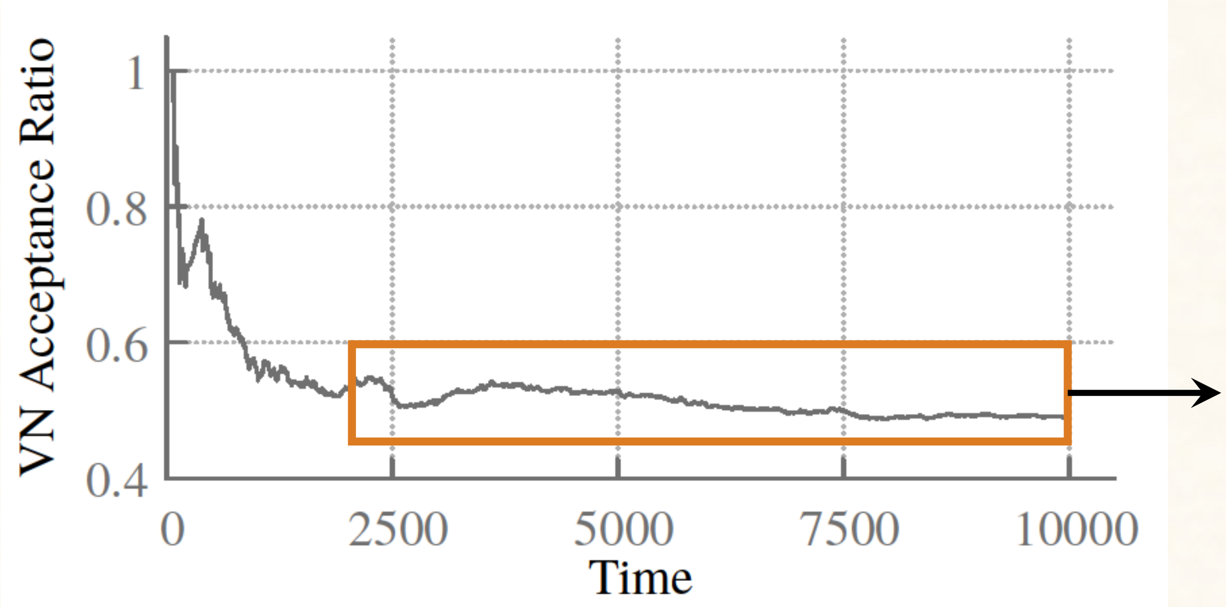
# Impact of Dynamicity: An Empirical Study

*300+* SNodes, *900+* SLinks. (*AS6461*), *4 – 8* VNodes/VN (*50%* conn. pr.)

Poisson Arrival (*10VNs/100 T.U.*), Exponential Lifetime (*1000 T.U.*)

Optimal embedding that minimizes total bandwidth consumption

# Impact of Dynamicity: An Empirical Study



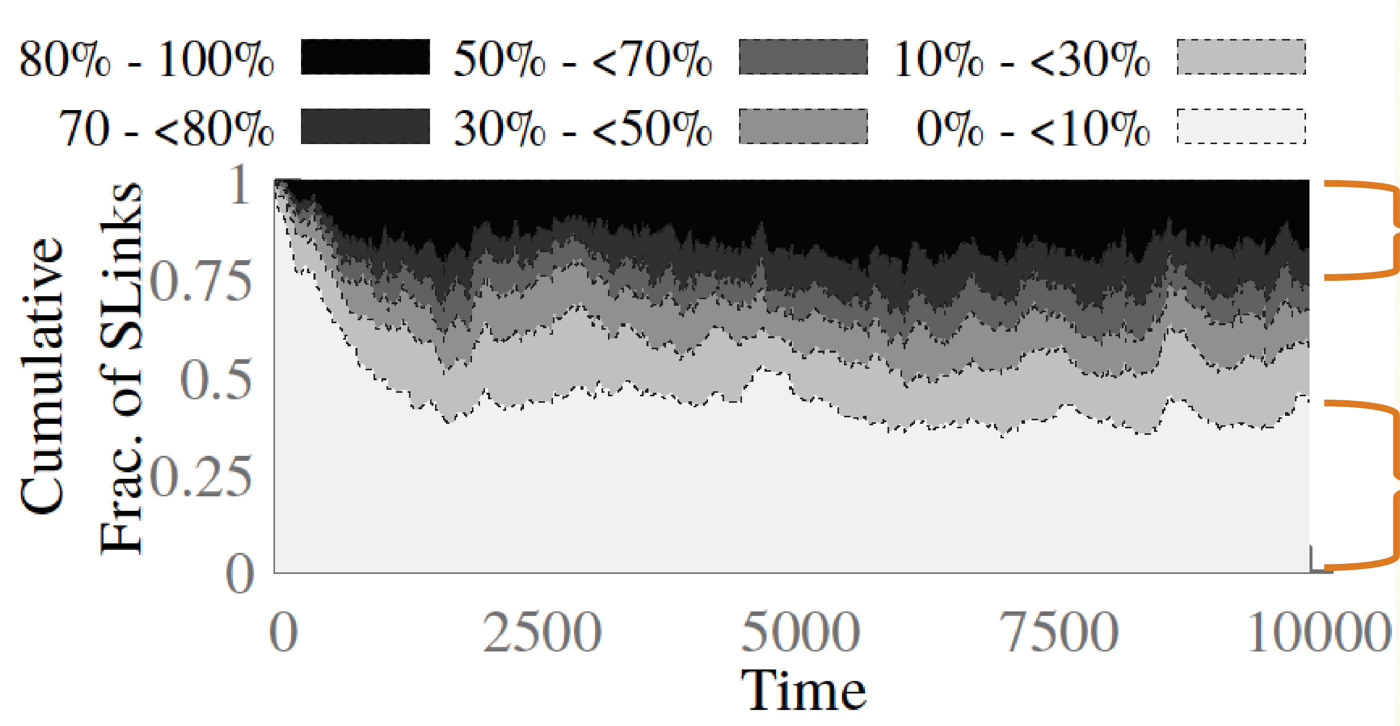
*Acceptance Ratio capped at ~50%*

*300+* SNodes, *900+* SLinks. (*AS6461*), *4 – 8* VNodes/VN (*50%* conn. pr.)

Poisson Arrival (*10VNs/100 T.U.*), Exponential Lifetime (*1000 T.U.*)

Optimal embedding that minimizes total bandwidth consumption

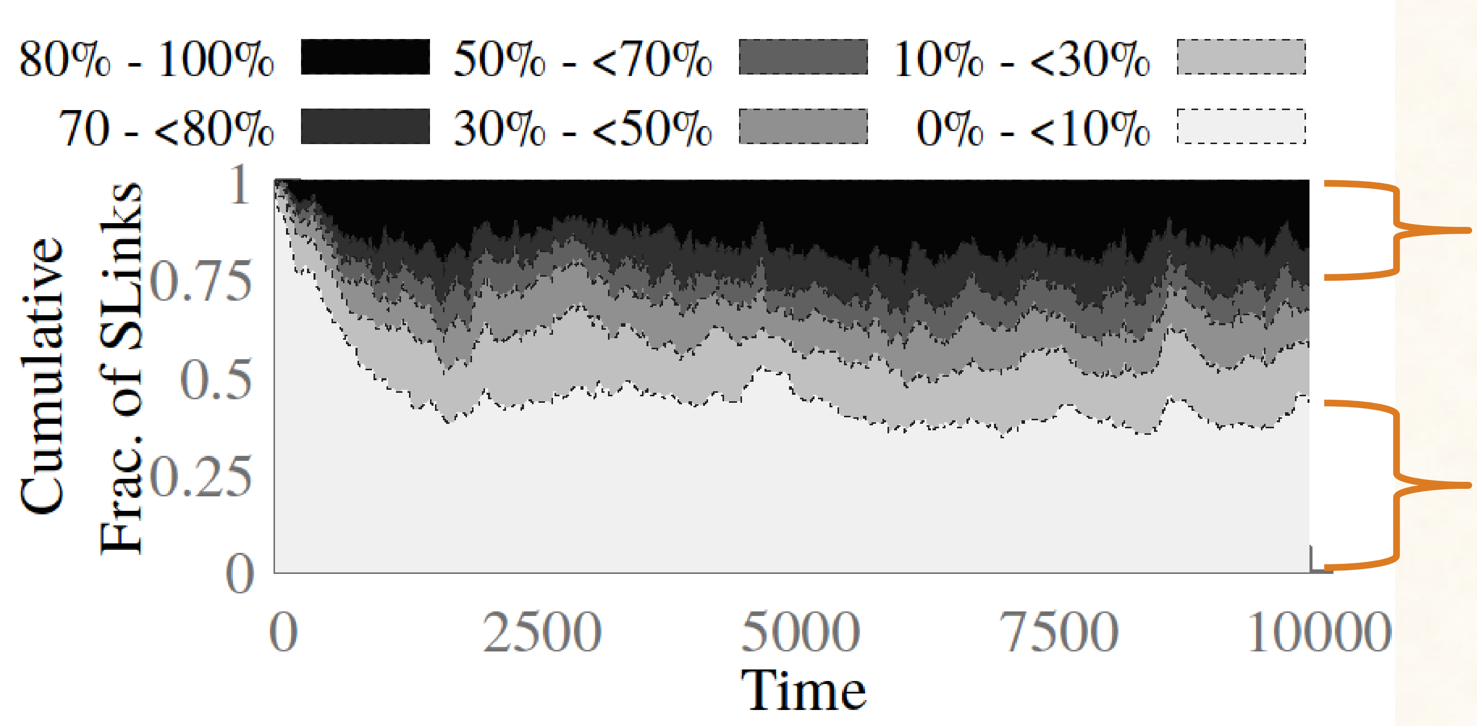
# Impact of Dynamicity: An Empirical Study



*~30% links utilized >= 70%*

*~40% links utilized <= 10%*

# Impact of Dynamicity: An Empirical Study



*~30% links utilized >= 70%*

*~40% links utilized <= 10%*

*Skewed Substrate Link Utilization* impacts *Acceptance Ratio* !!

## Key Question:

How to cope with the dynamicity in Network Virtualization when little or no information about the future is available?

## (One Possible) Answer:

Periodically adjust the embedding to eliminate “*bottlenecks*” and “*optimize resource usage*”

# The Problem

Reallocation of Virtual Network Embedding (ReViNE)

Given a Substrate Network and a set of embedded Virtual Networks

# The Problem

Reallocation of Virtual Network Embedding (ReViNE)

Given a Substrate Network and a set of embedded Virtual Networks

Migrate Virtual Nodes to New Substrate Nodes

# The Problem

## Reallocation of Virtual Network Embedding (ReViNE)

Given a Substrate Network and a set of embedded Virtual Networks

Migrate Virtual Nodes to New Substrate Nodes

Migrate Virtual Links to New Substrate Paths

# The Problem

## Reallocation of Virtual Network Embedding (ReViNE)

Given a Substrate Network and a set of embedded Virtual Networks

Migrate Virtual Nodes to New Substrate Nodes

Migrate Virtual Links to New Substrate Paths

Objective: Eliminate Substrate Bottlenecks* and Minimize Resource Usage**

* Links with utilization >= θ%

** In our case, bandwidth consumed by virtual links

# Our Proposal

A suit of solutions to ReViNE

ReViNE-OPT

ILP-based optimal solution*
(NP-Hard)

ReViNE-FAST

Simulated Annealing-based
heuristic

* Details is in the paper

# Do We Need A Heuristic?

## Computing Optimal Solution is Very Expensive

*H/W Configuration:* 8x10 Core Intel Xeon E5 CPU, 1TB RAM

*Observed limits for ILP*: 50 – 100 Node SN with < 60VNs took several hours and several 10s of GB RAM

## ILP Can Yield Impractical Solutions

- A practical solution contains a sequence of operations to reach the re-optimized state (also satisfy *make-before-break* constraint)
- Not possible to model in ILP. Final state obtained from ILP can be unreachable without violating *make-before-break* constraint.

# Do We Need A Heuristic?

## Computing Optimal Solution is Very Expensive

*H/W Configuration:* 8x10 Core Intel Xeon E5 CPU, 1TB RAM

*Observed limits for ILP*: 50 – 100 Node SN with < 60VNs took several hours and several 10s of GB RAM

## ILP Can Yield Impractical Solutions

- A practical solution contains a sequence of operations to reach the re-optimized state (also satisfy *make-before-break* constraint)
- Not possible to model in ILP. Final state obtained from ILP can be unreachable without violating *make-before-break* constraint.

# Do We Need A Heuristic?

Computing Optimal Solution is Very Expensive

*H/W Configuration:* 8x10 Core Intel Xeon E5 CPU, 1TB RAM

*Observed limits:* 50 – 100 Node SN with < 60VNs took several hours and several 10s of GB RAM

ILP Can Yield Impractical Solutions

- A practical solution contains a sequence of operations to reach the re-optimized state (also satisfy *make-before-break* constraint)
- Hard to model in ILP. Final state obtained from ILP can be unreachable without violating *make-before-break* constraint.

# Heuristic Design

## Our Objectives are Conflicting

Minimize Bottleneck Links

Minimize Bandwidth Usage

# Heuristic Design

## Our Objectives are Conflicting

Minimize Bottleneck Links

Distribute load across substrate links

Paths can become longer

Minimize Bandwidth Usage

# Heuristic Design

## Our Objectives are Conflicting

### Minimize Bottleneck Links

Distribute load across substrate links

Paths can become longer

### Minimize Bandwidth Usage

Route Virtual Links on Shorter Paths

Substrate links on shorter paths can become bottlenecks

# Heuristic Design

## Our Objectives are Conflicting

Minimize Bottleneck Links

Distribute load across substrate links

Paths can become longer

Minimize Bandwidth Usage

Route Virtual Links on Shorter Paths

Substrate links on shorter paths can become bottlenecks

CONFLICT !!

# Heuristic Design

## Our Objectives are Conflicting

Minimize Bottleneck Links

Distribute load across substrate links

Paths can become longer

Minimize Bandwidth Usage

Route Virtual Links on Shorter Paths

Substrate links on shorter paths can become bottlenecks

CONFLICT !!

Instead of an one-shot algorithm, *use a meta-heuristic (Simulated Annealing)* to explore the solution space and find a balance.

# Simulated Annealing: Neighborhood Generation

## Bottleneck Substrate Link Reconfiguration

Select a bottleneck substrate link and reroute virtual links using that bottleneck link until it is no longer a bottleneck.

## Virtual Node Migration

Randomly select a VN and re-embed a random virtual node and incident virtual links.

## Virtual Link Migration

Randomly select a VN and reroute a randomly selected virtual link.

# Simulated Annealing: Neighborhood Generation

## Bottleneck Substrate Link Reconfiguration

Select a bottleneck substrate link and reroute virtual links using that bottleneck link until it is no longer a bottleneck.

## Virtual Node Migration

Randomly select a VN and re-embed a random virtual node and incident virtual links.

## Virtual Link Migration

Randomly select a VN and reroute a randomly selected virtual link.

# Simulated Annealing: Neighborhood Generation

## Bottleneck Substrate Link Reconfiguration

Select a bottleneck substrate link and reroute virtual links using that bottleneck link until it is no longer a bottleneck.

## Virtual Node Migration

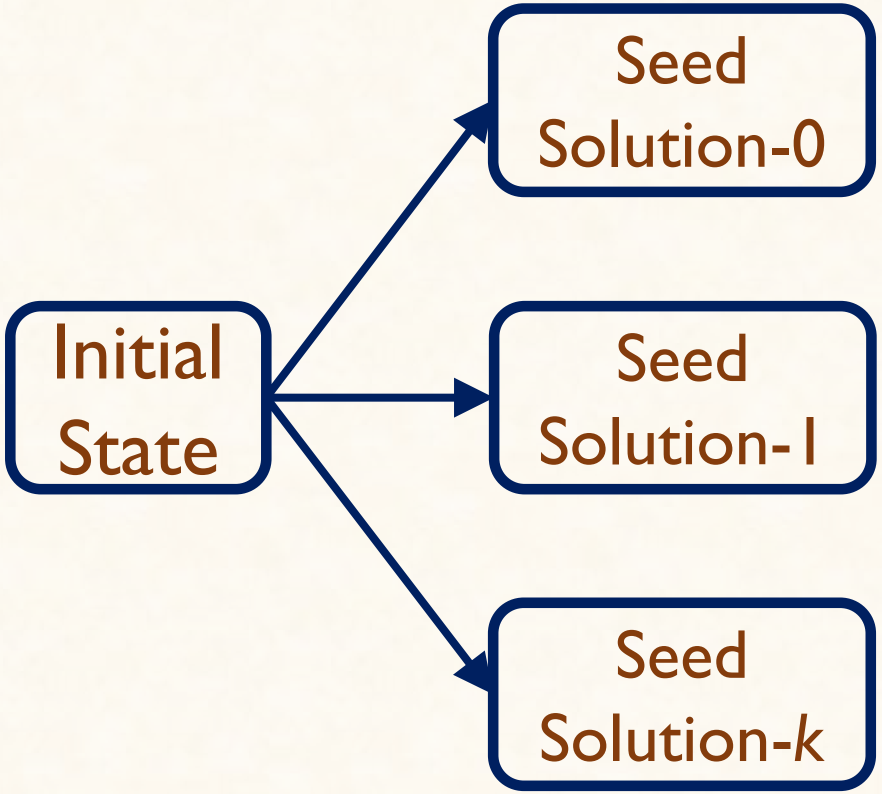
Randomly select a VN and re-embed a random virtual node and incident virtual links.

## Virtual Link Migration

Randomly select a VN and reroute a randomly selected virtual link.

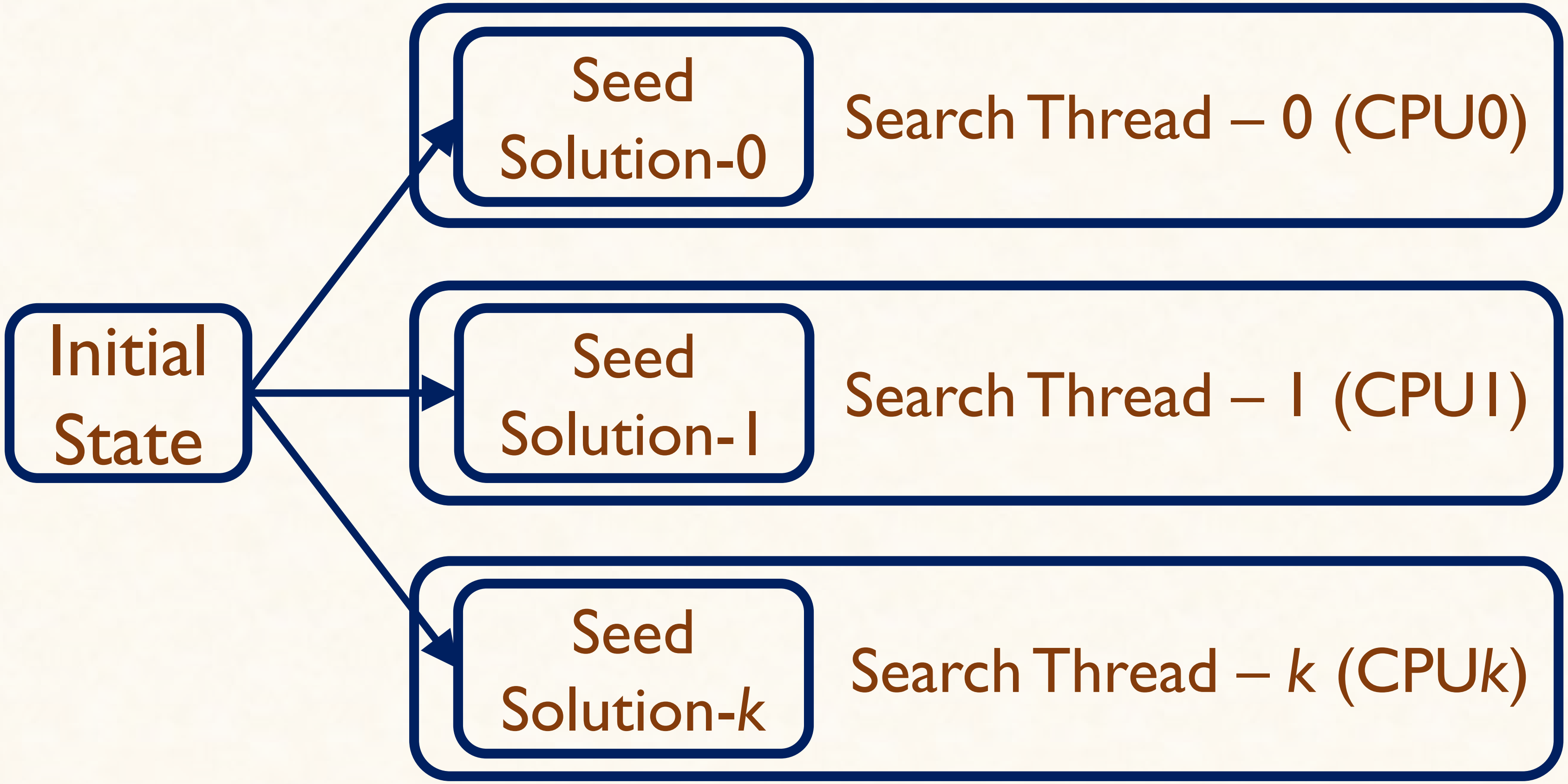
# Exploiting Multi-core CPU

## Parallel Simulated Annealing Searches



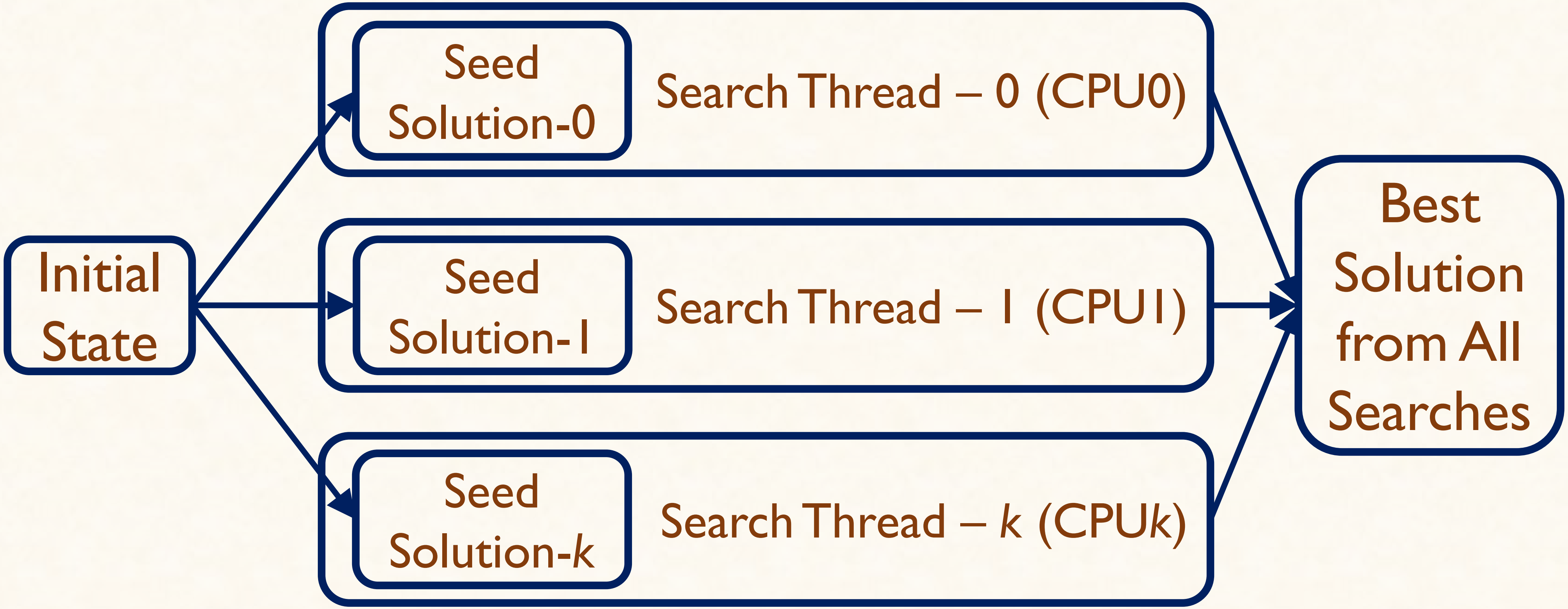
# Exploiting Multi-core CPU

## Parallel Simulated Annealing Searches



# Exploiting Multi-core CPU

## Parallel Simulated Annealing Searches



Initial State

Seed Solution-0

Search Thread – 0 (CPU0)

Seed Solution-1

Search Thread – 1 (CPU1)

Seed Solution-*k*

Search Thread – *k* (CPU*k*)

Best Solution from All Searches

# Evaluation: Setup

- ReViNE-FAST compared with ReViNE-OPT and SA-realloc*
- Parameters
  - 50 – 100 node synthetic substrate network
  - Larger test cases with 1000 node (heuristic only comparison)
  - Mean degree between 3.6 – 4
  - Mean substrate link utilization 60% - 80%
  - Bottleneck substrate link threshold 70% - 90%

* Masti, S,. *et al.* "Simulated Annealing Algorithm for Virtual Network Reconfiguration", *8th Euro-NGI Conference on Next Generation Internet*, IEEE, 2012, pp. 95-102.

# ReViNE-FAST Performance Highlights

Within ~*19% of optimal* (ReViNE-OPT) on avg.

~*3x less cost* compared to SA-realloc on avg.

~*5% more VNs accepted* on avg. when combined with optimal VN embedding algorithm

# Summary

ReViNE is one possible way to address the dynamicity in VN arrival/departure

ReViNE-FAST, a simulated annealing based heuristic performs ~*19% within the optimal* (empirically evaluated)

ReViNE-FAST performs ~*3x better* than S.O.A Simulated Annealing-based heuristic

# Questions?

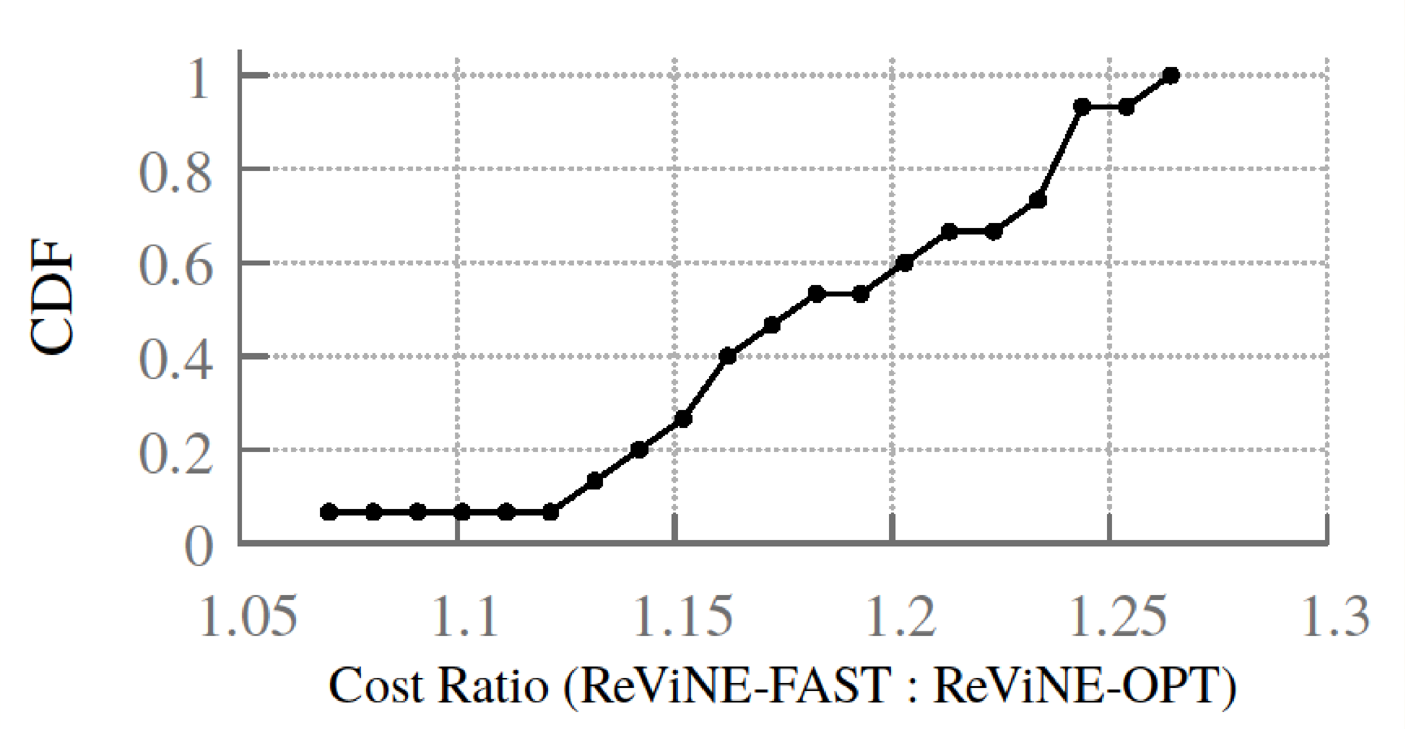
Source Code

CPLEX: https://github.com/srcvirus/vne-reallocation-cplex

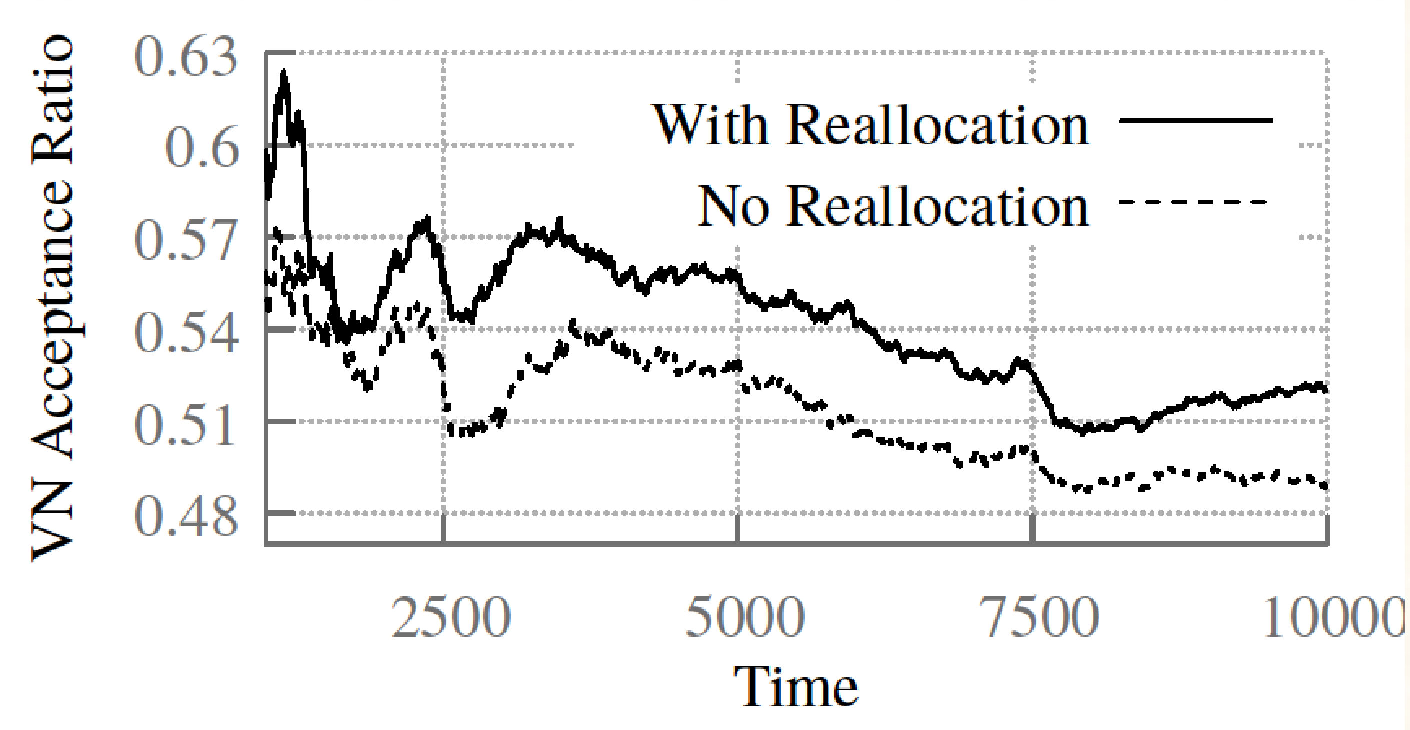
Simulated Annealing: https://github.com/srcvirus/vne-reallocation-sa

# Backup

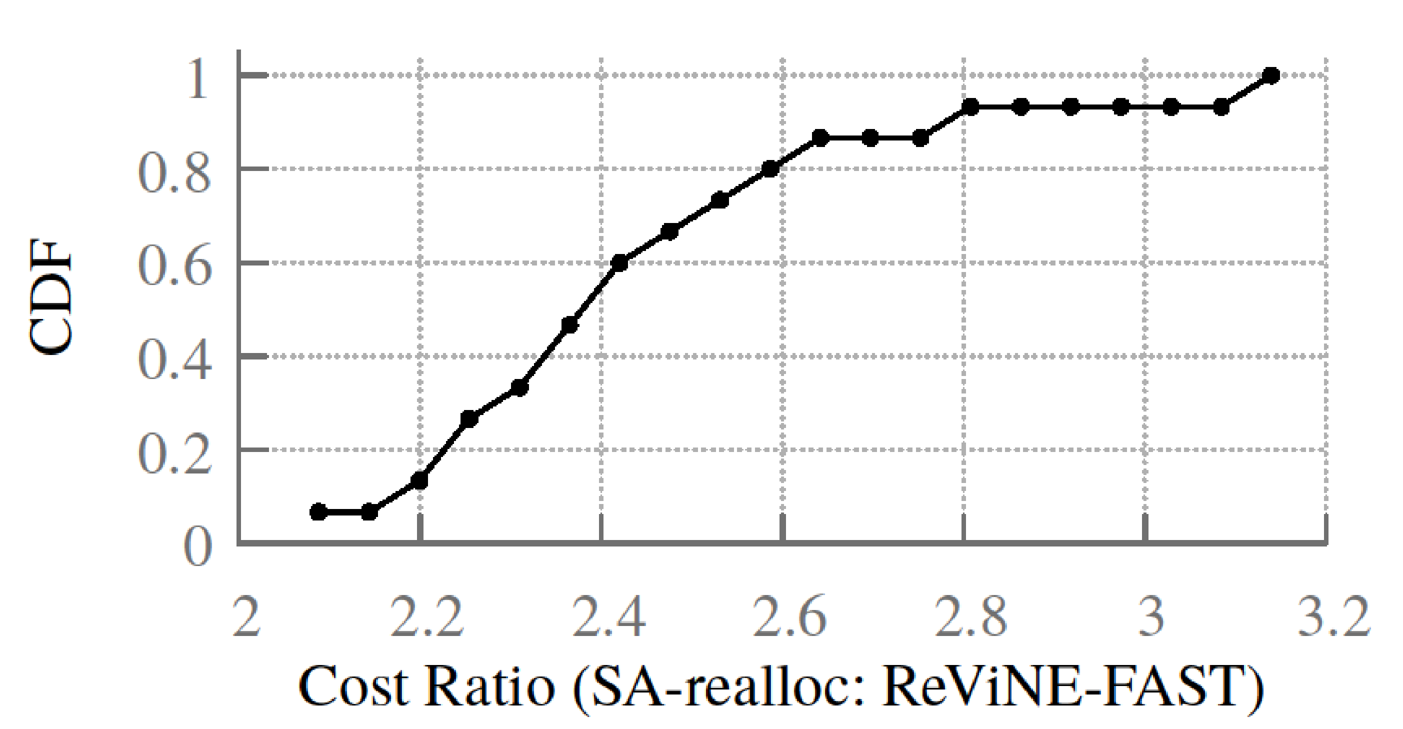
# ReViNE-FAST vs ReViNE-OPT



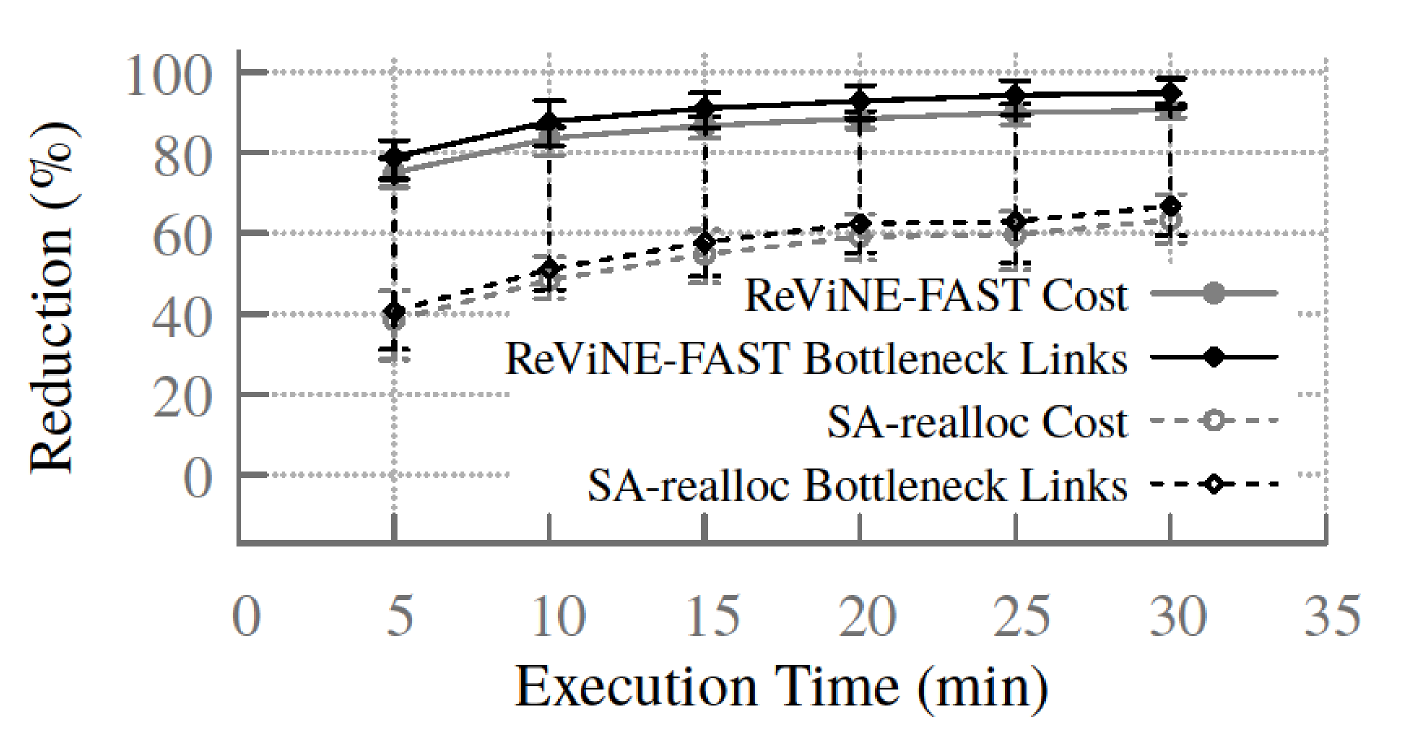
# Impact of Reallocation



# ReViNE-FAST vs SA-Realloc (Large Cases)



# ReViNE-FAST Convergence (Large Cases)



# State-of-the-art

*Reactive One-shot Approaches*: Reallocate VNs when a new VN cannot be embedded [1][2]

*Proactive One-shot Approaches*: Periodically reallocate VNs [3][4]

*Meta-heuristic Approaches*: Simulated Annealing [5], Particle Swarm Optimization [6]

[1] Y. Zhu *et al.*, "Algorithms for assigning substrate network resources to virtual network components", IEEE INFOCOM, 2006.

[2] M. Yu, *et al.* "Rethinking virtual network embedding: substrate support for path splitting and migration", ACM SIGCOMM CCR, 38(2), 2008, pp. 17–29.

[3] N. F. Butt, *et al.* "Topology-awareness and reoptimization mechanism for virtual network embedding", Int. Conf. on Research in Networking 2010.

[4] P. N. Tran, et al., "Optimal mapping of virtual networks considering reactive reconfiguration," IEEE CloudNet, 2012.

[5] S. Masti, *et al.* "Simulated Annealing Algorithm for Virtual Network Reconfiguration", 8th Euro-NGI Conf. on Next Generation Internet, IEEE, 2012.

[6] Y. Yuan, *et al.* ,"Discrete particle swarm optimization algorithm for virtual network reconfiguration," Int. Conf. in Swarm Intelligence, 2013.