University of Glasgow

Genetic Algorithms for Service Function Chain Deployment

Theviyanthan Krishnamohan (Thivi)

Second-year PhD student at the University of Glasgow.

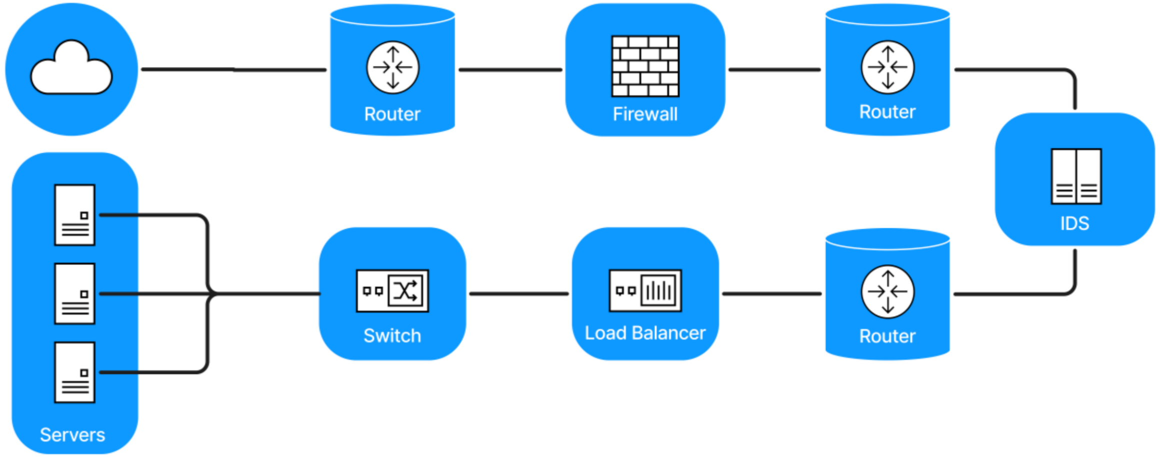
t.krishnamohan.1@research.gla.ac.uk

Background

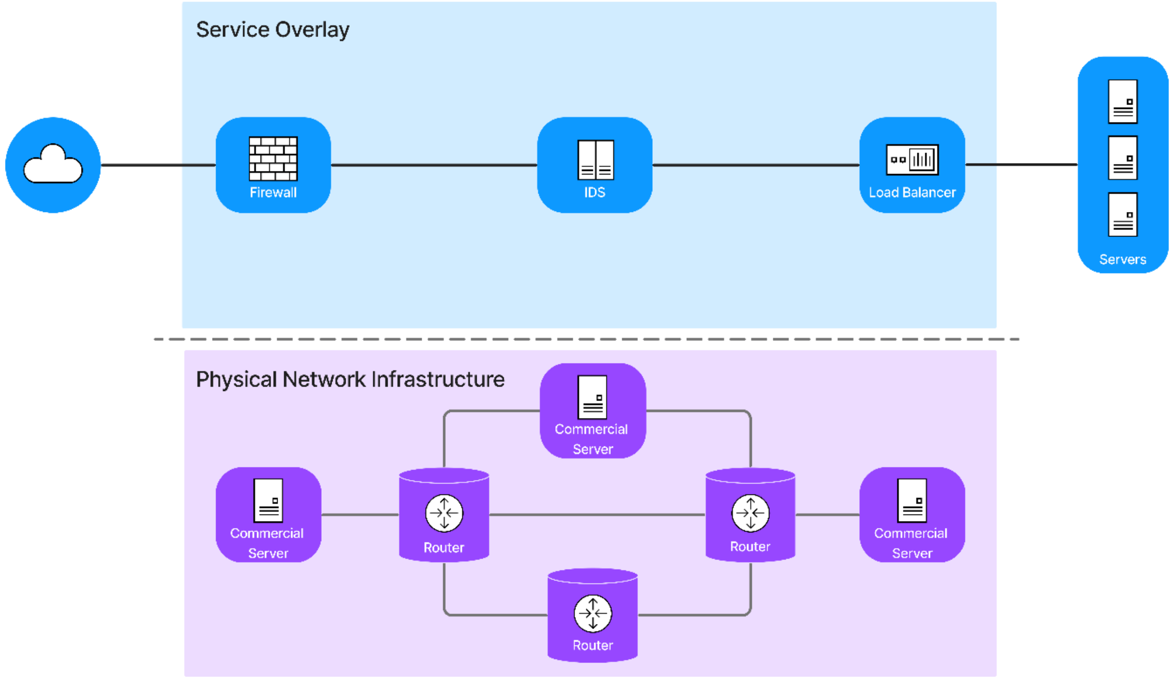
What are Service Function Chains (SFCs)?

- SFCs combine Network Function Virtualisation and Software-Defined Networking to create a service overlay over the physical network.

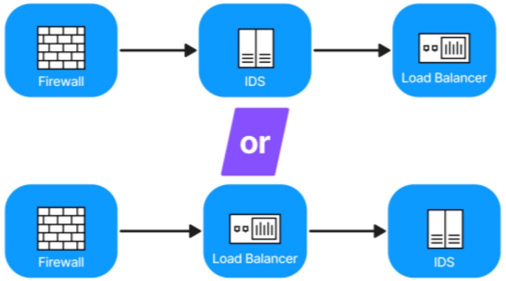
A traditional network:



A Service Function Chain:

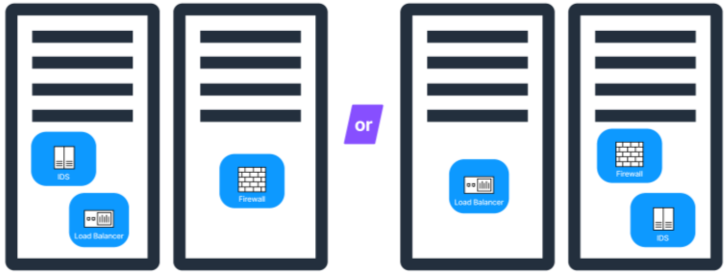


Optimisation Challenges



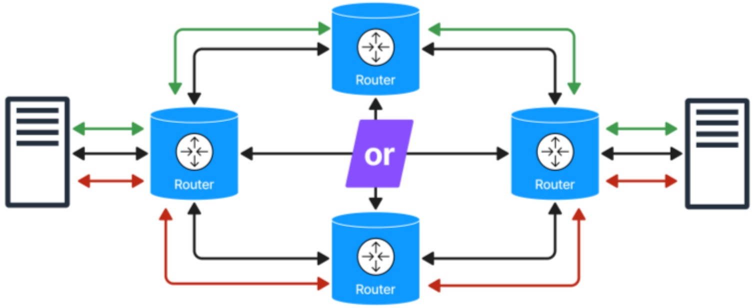
Chain composition

How should the Virtual Network Functions (VNFs) be ordered for optimal performance?



VNF embedding

Where should the VNFs be deployed for optimal performance?



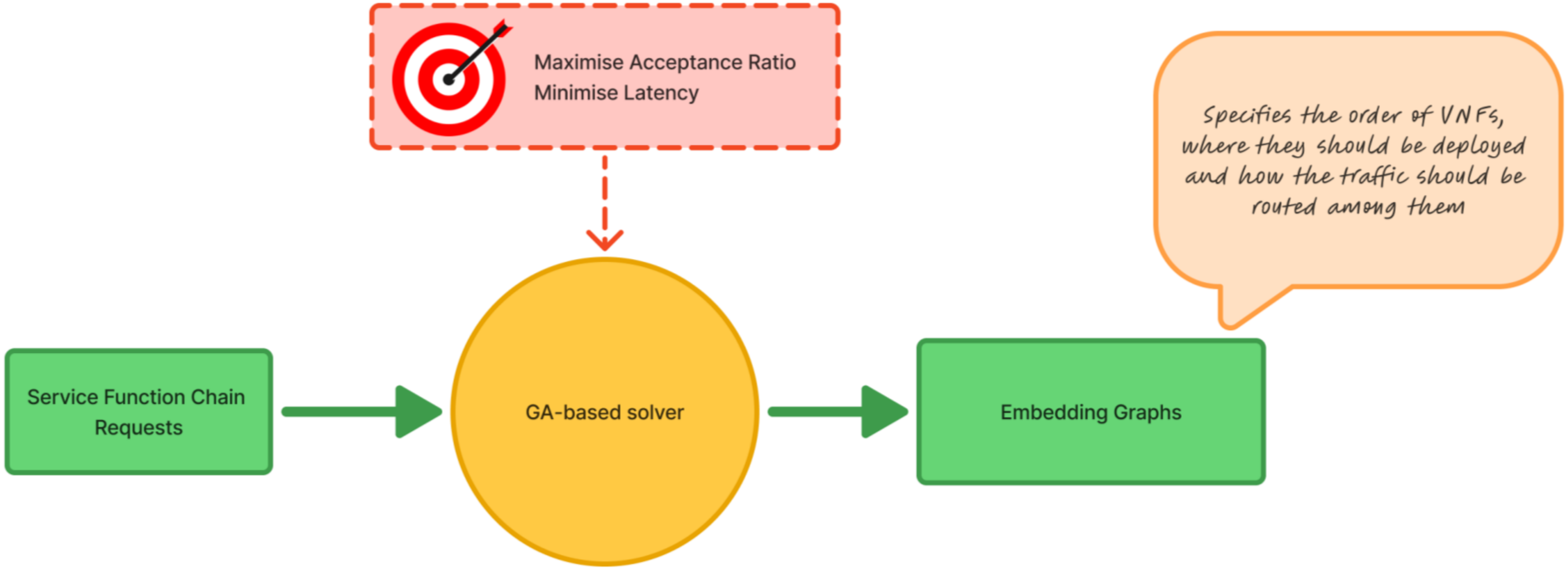
Link embedding

How should the VNFs be linked for optimal performance?

- This has been shown to be an NP-hard optimisation problem.

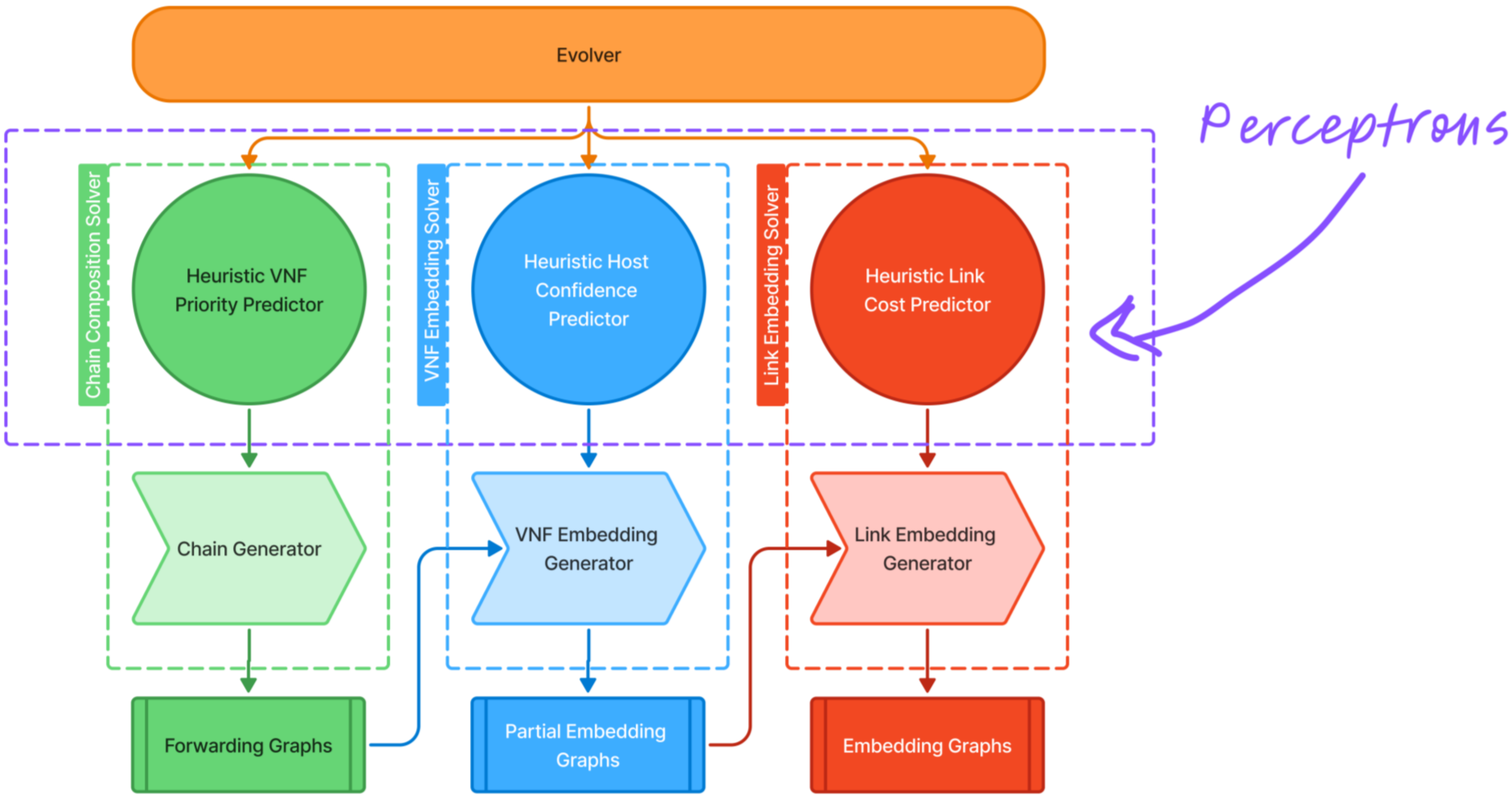
Using GA for optimal SFC embedding

Bird’s Eye View


Maximise Acceptance Ratio
Minimise Latency
Specifies the order of VNFs, where they should be deployed and how the traffic should be routed among them
Service Function Chain Requests
GA-based solver
Embedding Graphs

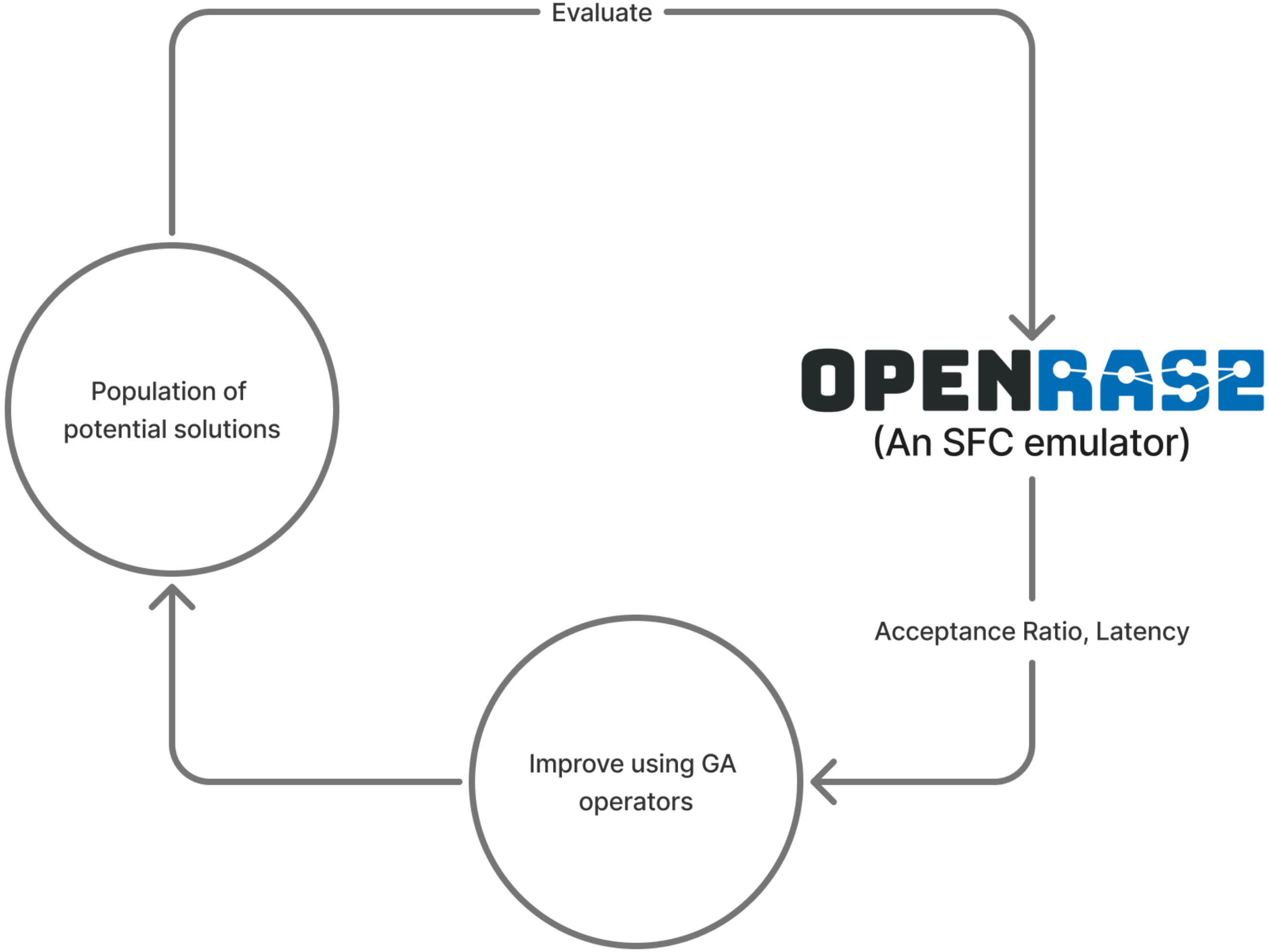
- Acceptance Ratio—the number of SFC Requests that can be accepted over the total number of SFC Requests received.
- Latency—the amount of time taken for traffic to traverse the SFC

GA-based Solver Architecture



Online Evolution

- It involves evaluating potential solutions on a network and evolving them using Genetic Algorithms.
- Simulators and numerical analysis may not capture the complexity of real networks.
- Makes the solution self-adaptive.
- But it is time consuming.

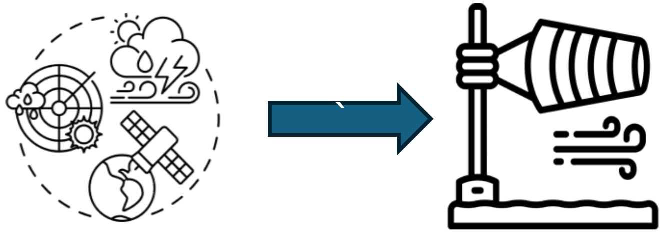

Evaluate
Population of potential solutions
OPENRASE
(An SFC emulator)
Acceptance Ratio, Latency
Improve using GA operators

Why a Genetic Algorithm?

- Back propagation/gradient descent cannot be used as the error function is unknown.
- The ‘error’ is evaluated by online experiments on OpenRASE.
- Back propagation/gradient descent cannot be done concurrently.
- GA can explore the whole search space and is adaptable to a dynamic environment, but it is an underutilised algorithm in the SFC realm. Only 12/163 surveyed studies use GAs.

Surrogate

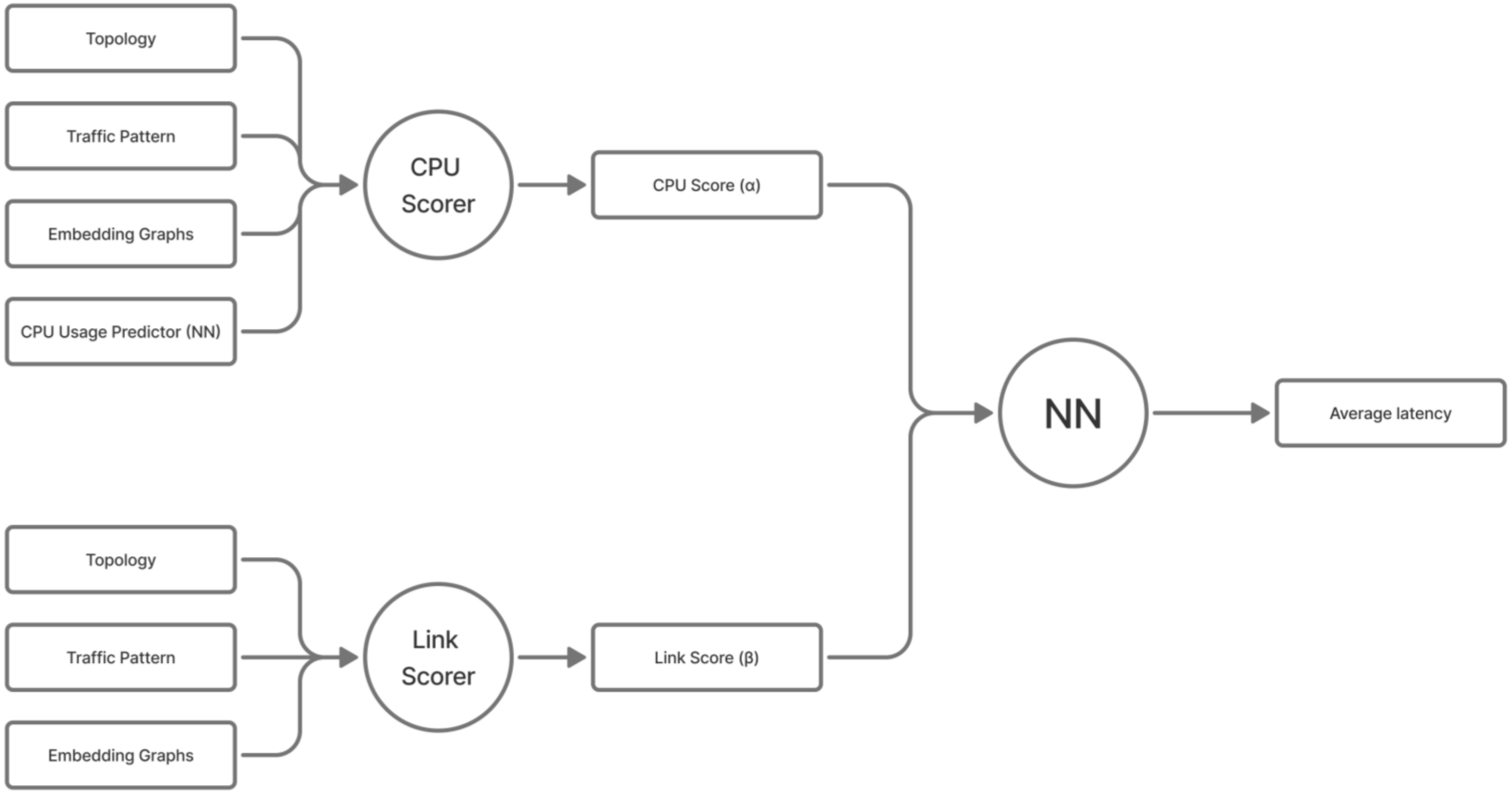
Surrogate



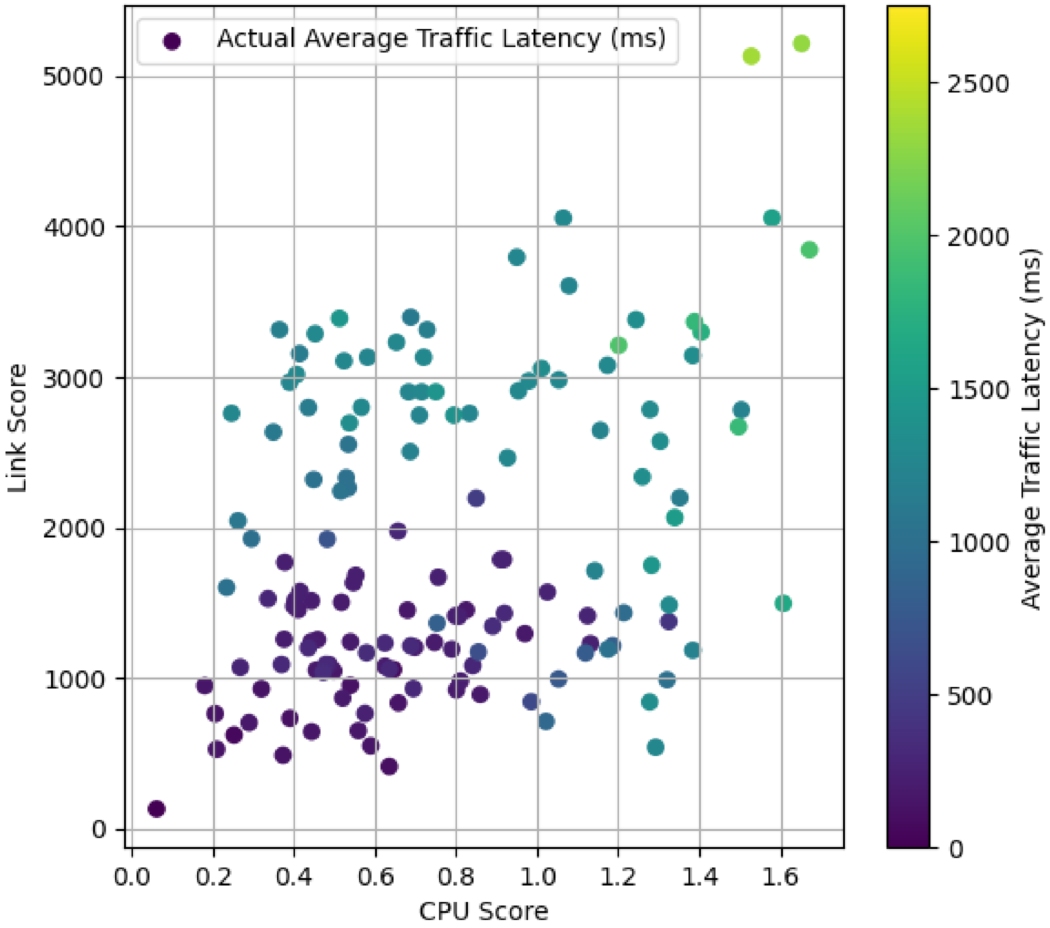
- Online evolution is time-consuming.
- To mitigate this issue, we evaluate candidates with a simplified evaluation environment: surrogate.
- It is an ANN trained on data from OpenRASE. It predicts the latency of a set of embedding graphs, allowing us to perform online evolution quickly.

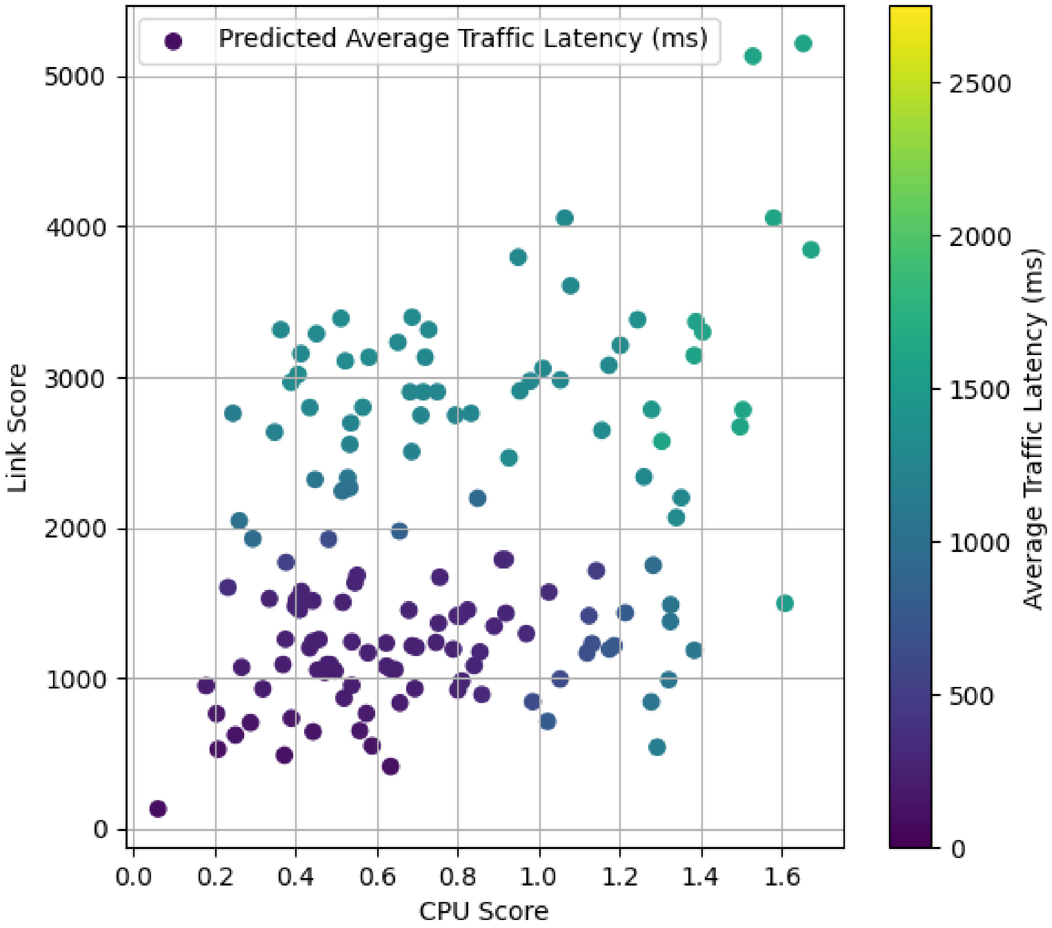
Encoding Embedding Graphs

- The first challenge is to encode the embedding graph into a numerical form.



Performance





Evolution Control

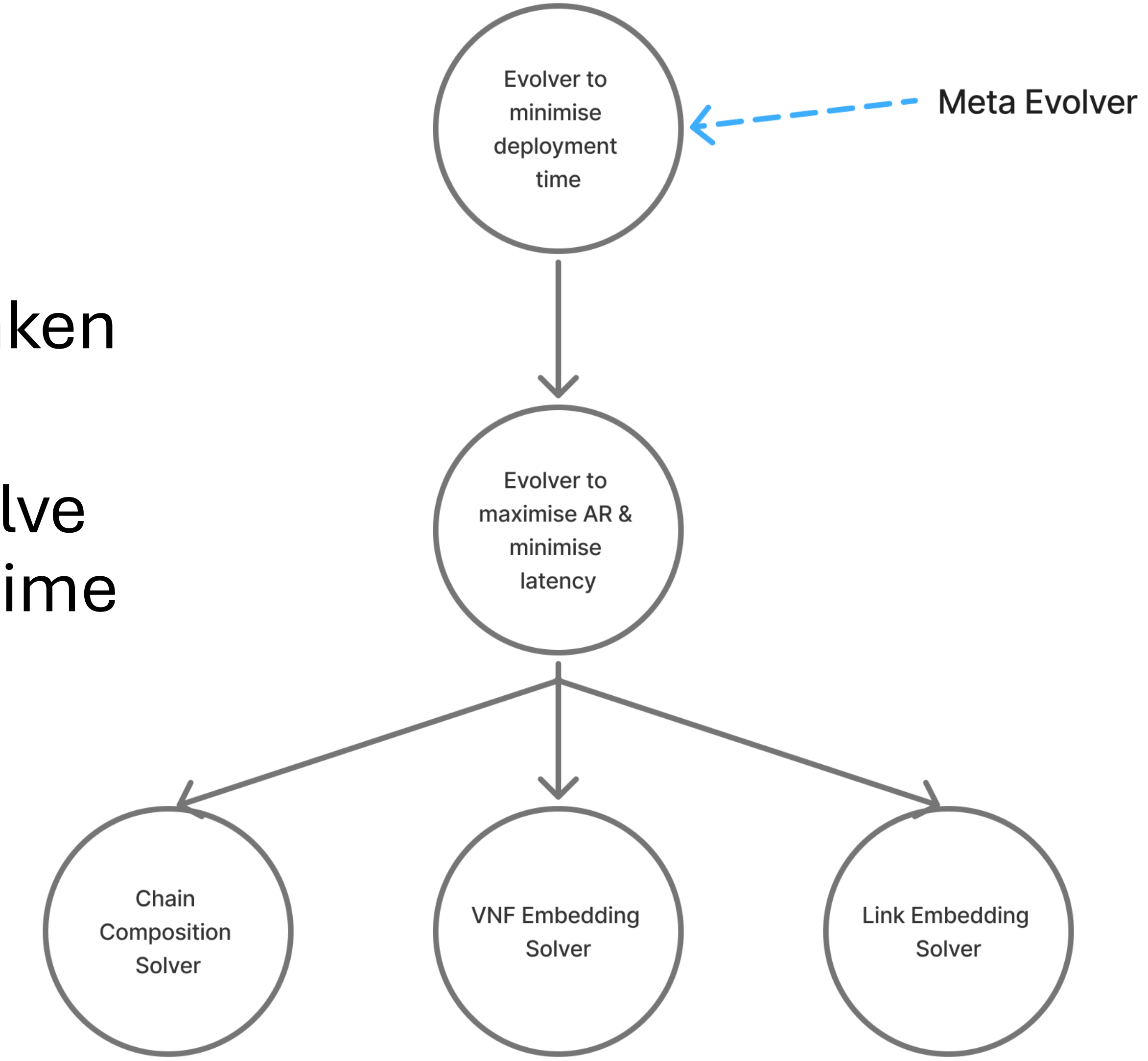
- We use generation-based evolution control.
- So, we evolve using the surrogate until a certain fitness threshold is reached.
- Then, the evolution continues on OpenRASE.
- Ideally, one generation of evolution on OpenRASE should suffice.
- The threshold is decided by an expert initially.

- This saves significant time and allows us to explore the search space more.

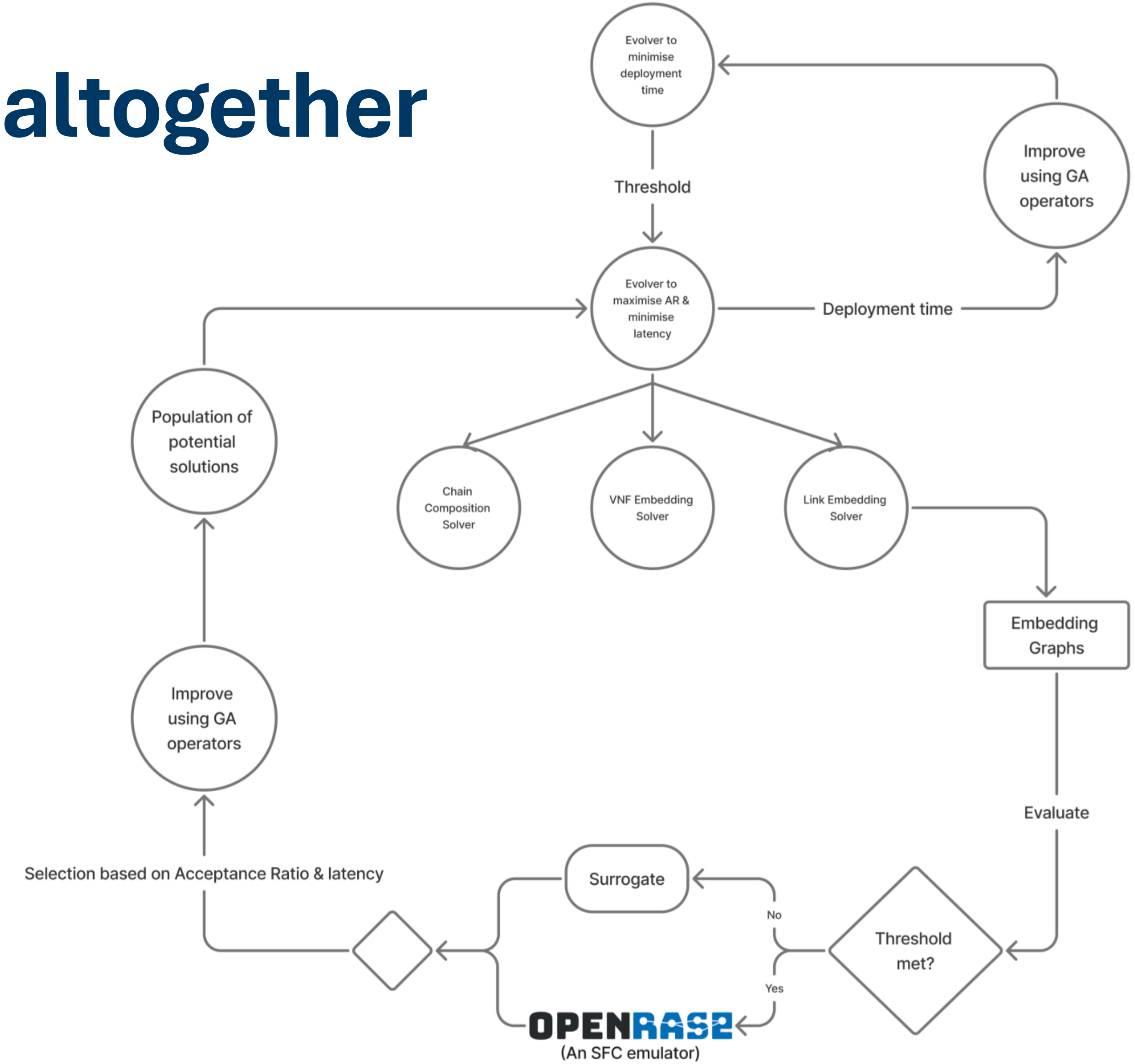
Meta Evolver (WIP)

Meta evolver

- However, as we make the threshold tougher, the time taken can increase.
- A *meta-evolver* is used to evolve the thresholds such that the time taken to produce an optimal embedding graph is reduced.



Putting it altogether


Evolver to minimise deployment time
Threshold
Improve using GA operators
Evolver to maximise AR & minimise latency
Deployment time
Population of potential solutions
Chain Composition Solver
VNF Embedding Solver
Link Embedding Solver
Embedding Graphs
Improve using GA operators
Evaluate
Selection based on Acceptance Ratio & latency
Surrogate
No
Threshold met?
Yes
OPENRASE
(An SFC emulator)

Questions?

Thank You

Appendix

Meta Evolution Control

- I have thought of two strategies:
    - Hill climbing—start with one individual. Evolution contains a watchdog timer, and once it expires, the individual is mutated.
    - Evolve by considering the shortest distance between an individual and the threshold.
        - Start with random thresholds.
        - Evolve to optimise AR and latency for each threshold concurrently.
        - After *n* generations, compute the distance between individuals and the thresholds.
        - The fitness for the threshold evolution is the shortest distance.
        - Continue with the GA operations as usual.

CPU & Link Scorers

- CPU score gives the average maximum CPU usage of a host that hosts an SFC.
- Link score gives the average aggregate link utilisation of all links in an SFC.

Why Perceptrons?

- The chain composition, VNF embedding, and link embedding problems must be solved simultaneously.
- Coming up with an encoding scheme for all three problems is difficult.
- By using three perceptrons and evolving their weights using GA, a floating-point array can be used as the encoding scheme.

Discussion Points

- Evolve hyperparameters
- Predictor architecture in solvers
- Solver algorithms
- VNF CPU predictor model
- The thresholds and how they are used to conclude an evolution experiment