CEGS: Configuration Example Generalizing Synthesizer

Jianmin Liu, Li Chen, Dan Li, Yukai Miao

Tsinghua University

ZGC LAB

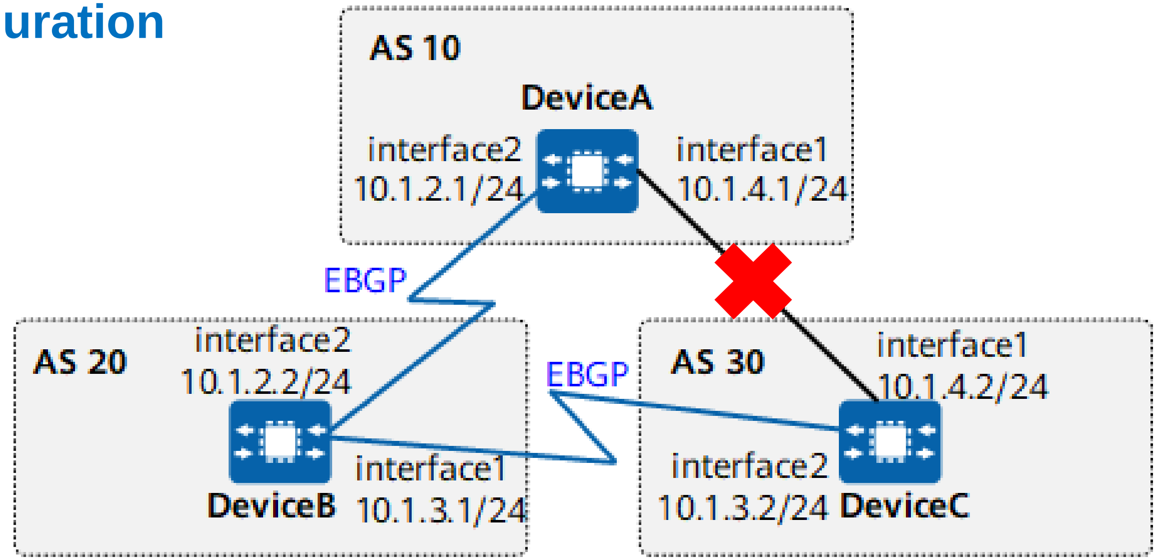
Device Configuration controls network behaviors

Requirement: **AS 10 cannot communicate with AS 30**

operator

configuration

```
! Device A
router bgp 10
 bgp router-id 1.1.1.1
 neighbor 10.1.2.2 remote-as 20
```




```
! Device B
router bgp 20
 bgp router-id 2.2.2.2
 neighbor 10.1.2.1 remote-as 10
 neighbor 10.1.3.2 remote-as 30

 neighbor 10.1.2.1 filter-list 1 out
 neighbor 10.1.3.2 filter-list 2 out
 ip as-path access-list 1 deny _30_
 ip as-path access-list 1 permit .*
 ip as-path access-list 2 deny _10_
 ip as-path access-list 2 permit .*
```

```
! Device C
router bgp 30
 bgp router-id 3.3.3.3
 neighbor 10.1.3.1 remote-as 20
```

Misconfiguration causes severe network downtime

Google Cloud Went Down Because It Was Misconfigured

Posted on **JUNE 7, 2019** Written by Bill Hartzer

A recent major outage of Google Cloud was caused because of a misconfiguration, not only because of network congestion, like Google initially reported. The outage was described by Google as "network congestion issue in eastern USA, affecting Google Cloud, G Suite, and YouTube". It also caused services such as Shopify, Snapchat, and Discord to go down. Some people reported that they could not control the temperature in their home or apartment through Google Nest.

SECURITY POLICY MANAGEMENT

One Simple Misconfiguration. 2.9 Billion Users Down

By FireMon | Oct 09, 2021

A routine maintenance error severs Facebook's data centers from the Internet for over 6 hours

On October 4, Facebook users suffered a complete outage affecting all apps including WhatsApp, Instagram, and Messenger for over 6 hours. Nearly 2.9 billion users were not only inconvenienced, but many also lost a crucial means of communication in regions where WhatsApp is the primary method for text messaging and voice calls.

It was quickly discovered that the culprit was a faulty configuration change on Facebook's backbone routers that manage traffic between their data centers. A simple misconfiguration was propagated across their entire network that affected not only their users, but also impacted their own tools and systems, hindering Facebook's ability to diagnose and solve the problem.

2024: the year misconfigurations exposed digital vulnerabilities

Small configuration errors cascaded into major outages during 2024. Mike Hicks, from Cisco ThousandEyes, propounds techniques to defend digital resilience against tales of the unexpected

Google Cloud Pub/Sub Disruption

On January 8, Google Cloud experienced a misconfiguration-related incident that impacted multiple regions of Pub/Sub, Cloud Logging, and BigQuery Data Transfer Service for close to 75 minutes. Pub/Sub is messaging middleware used for streaming analytics, data integration pipeline, (micro)service integration, and other tasks.

Synthesizers are key tools to combat misconfiguration

- Configuration synthesis

Intent (Define network behaviors)

Topology

⇒ **synthesizer** ⇒ **Configuration**

- **Existing synthesizers still require human involvement**

Require **drafting templates** (NetComplete [NSDI'18])

```
! B Configuration Sketch
! 10G interface to C
interface TenGigabitEthernet1/1/1
  ip address  ?   ?
  ip ospf cost  10 < ? < 100
router ospf 100
   ?
  ...
router bgp 6500
...
  neighbor AS200 import route-map imp-p1
  neighbor AS200 export route-map exp-p1
  ...
ip community-list C1 permit  ?
ip community-list C2 permit  ?
route-map imp-p1 permit 10
  set  ?
```

Require **coding DSL** to describe intents (Aura [NSDI'23])

```
policies{
  RSW_REACHABILITY{
      routing{topology f16}
      origin{location RSW}
      propagation{
          prop-condition (L or W) ← I4
          paths{
              path P1 R1P1 → F2P1 → R3P1 ← I1
              path P2 R1P1 → F2P1 → S1PL2 → F2P2 → R3P2 ← I2
          }
      }
      preference{
          P1 > P2 ← I3
          L > W ← I5
      }
  }
```

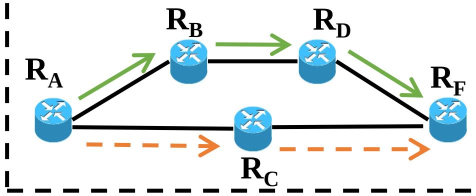
Why do current synthesizers still require human effort?

- **Main reason:** **lack a core capability —— Example Following and Generalization (EFG)**

1. Identify example from documentation

Target scenario

Intent: Prefer path (R_A, R_B, R_D, R_F) over (R_A, R_C, R_F)

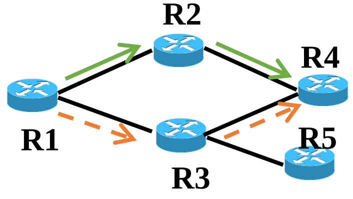


documentation

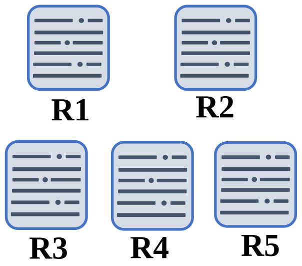
⇩

Configuration example

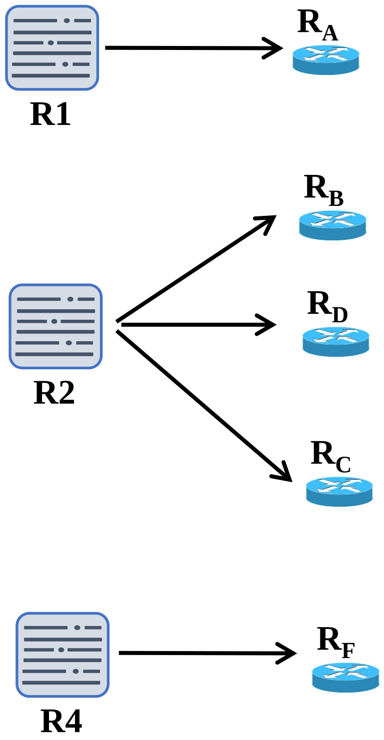
Intent: Prioritize route (R1, R2, R4) over (R1, R3, R4)



Configuration



2. Assign snippets to specific devices



3. Generate configuration

Configuration template for R_A

```
…
router bgp 100
neighbor 192.168.5.1 remote-as 200
neighbor 192.168.5.1 route-map RA_from_ RB in
…
ip community-list 1 permit ?
route-map RA_from_ RB ? 10
  match community 1
  set local-preference 200
  set community ?
…
```

⇧

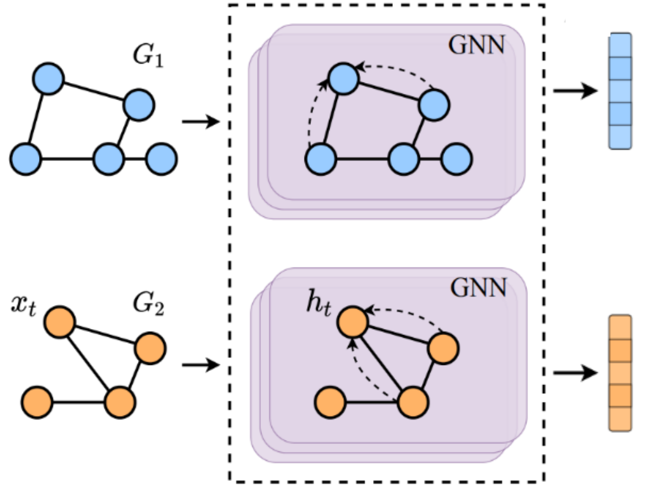
Configuration for R1

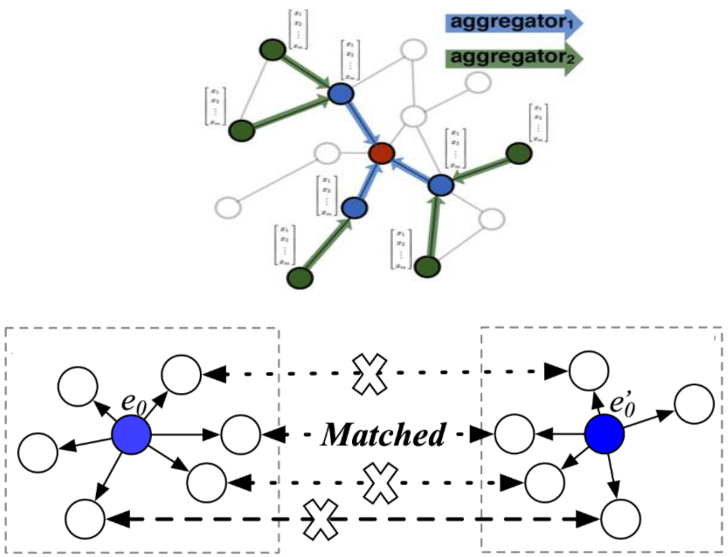
```
…
router bgp 10
neighbor 10.0.10.2 remote-as 20
neighbor 10.0.10.2 route-map R1 _from_ R2 in
…
ip community-list 1 permit 100:1
route-map R1 _from_ R2 permit 10
  match community 1
  set local-preference 200
  set community 100:1
…
```

Can we automate EFG for configuration synthesis?

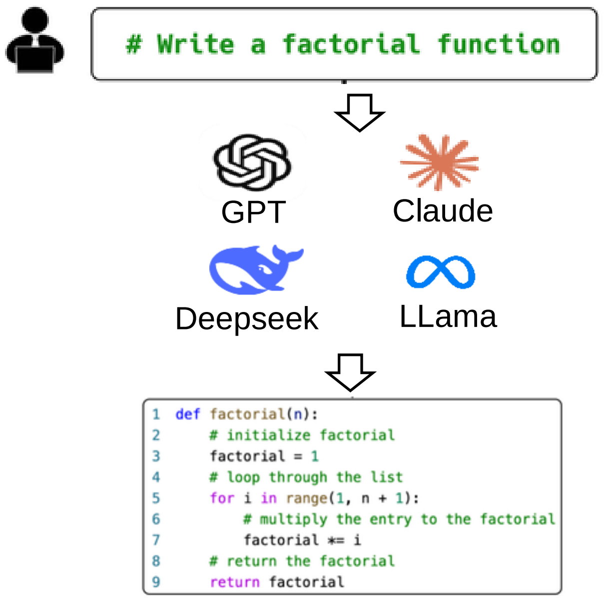
- GNNs and LLMs have potential to automate EFG

1. Identify example from documentation
2. Assign snippets to specific devices
3. Generate configuration





GNNs can model graph and node characteristics



LLMs comprehend human intents and generate code

Challenges C#

1. Identify example from documentation

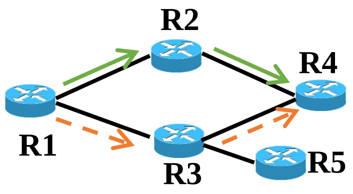
C1: Accurately identify relevant example with:

Similar intent

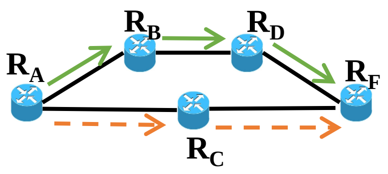
Example Intent:
Prefer path (R_A, R_B, R_D, R_F) over (R_A, R_C, R_F)

User Intent:
Prioritize route (R1, R2, R4) over (R1, R3, R4)

Topological similarity



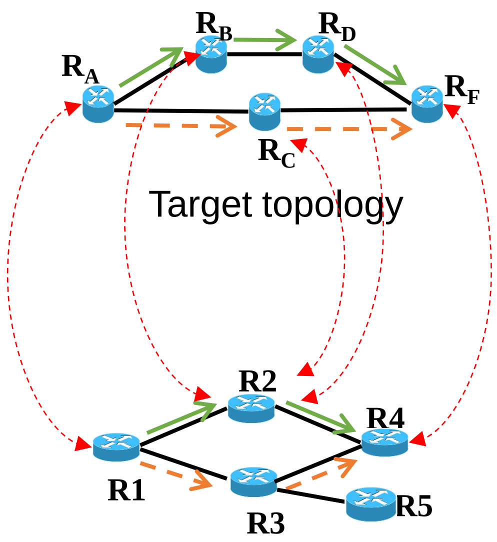
Example topology



Target topology

2. Assign snippets to specific devices

C2: Establish device association relation

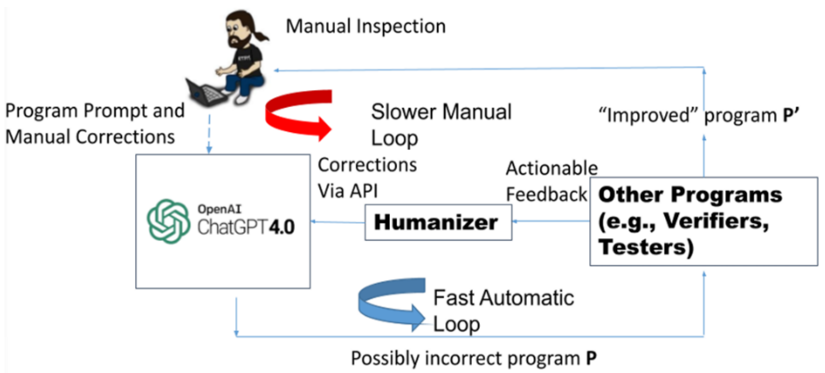


Example topology

3. Generate configuration

C3: Accelerate synthesis without compromising correctness

LLM requires multiple loops for correction

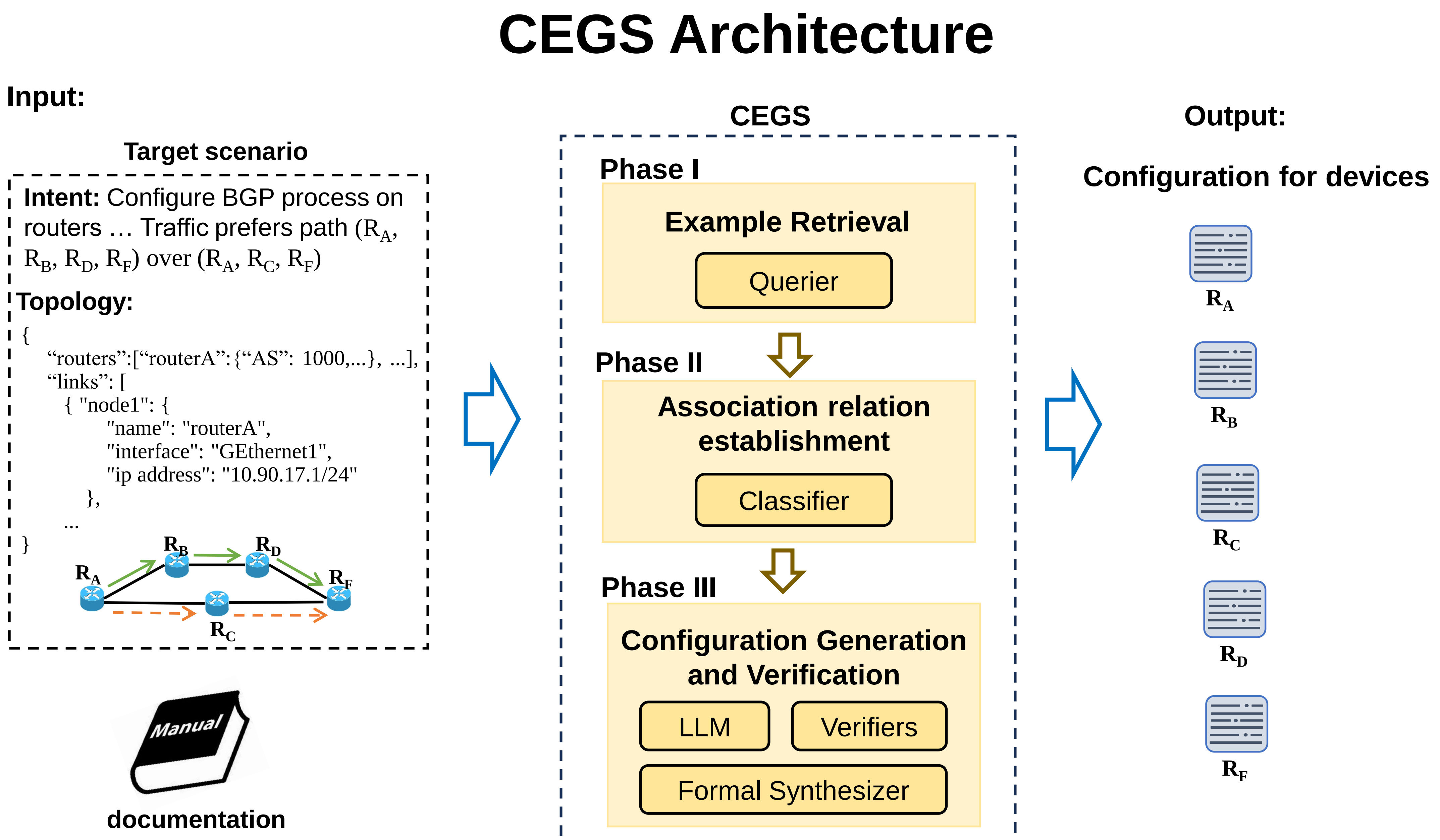


Architecture of COSYNTH [HotNets' 23]

Contributions

We build a tool called CEGS that:

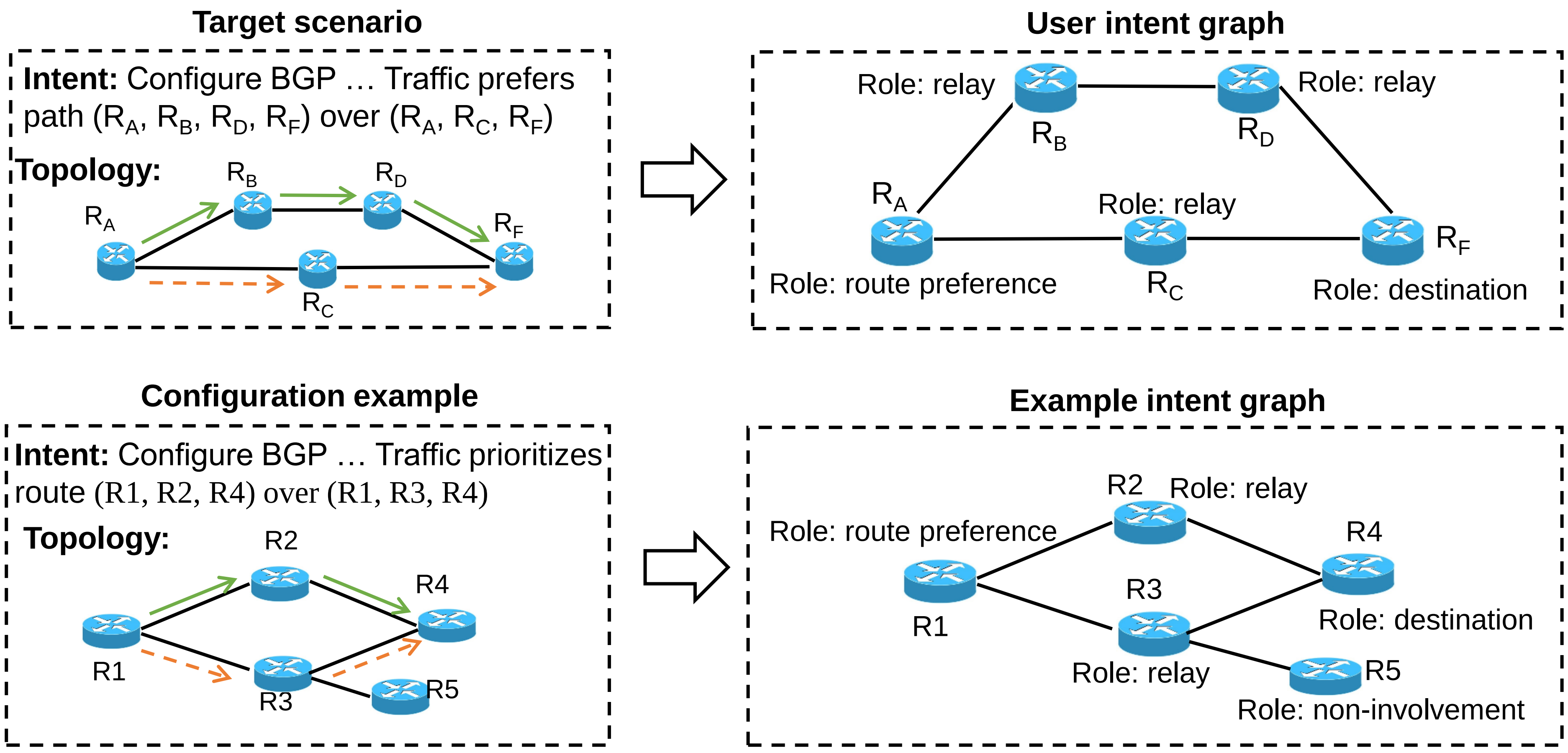
- Can **automate EFG** with GNN and LLM
- Can accurately **identify examples** with a GNN-based Querier
- Can accurately **associate devices** with a GNN-based Classifier
- Can rapidly generate correct configurations using an **efficient LLM-driven synthesis** method

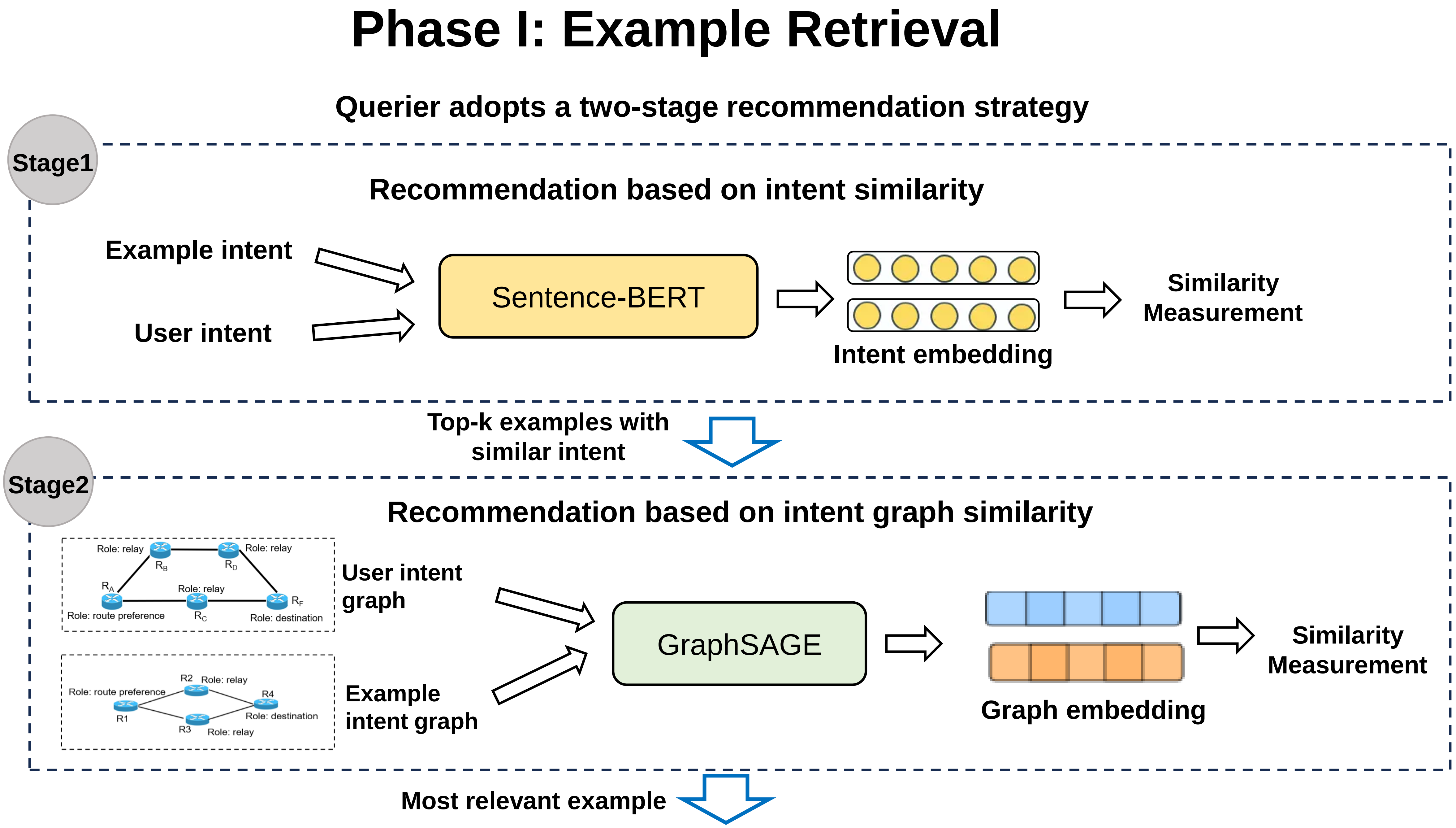

CEGS Architecture
Input:
Target scenario
Intent: Configure BGP process on routers … Traffic prefers path (RA, RB, RD, RF) over (RA, RC, RF)
Topology:
{
"routers":["routerA":{"AS": 1000,...}, ...],
"links": [
{ "node1": {
"name": "routerA",
"interface": "GEthernet1",
"ip address": "10.90.17.1/24"
},
...
}
RA
RB
RD
RF
RC
Manual
documentation
CEGS
Phase I
Example Retrieval
Querier
Phase II
Association relation establishment
Classifier
Phase III
Configuration Generation and Verification
LLM
Verifiers
Formal Synthesizer
Output:
Configuration for devices
RA
RB
RC
RD
RF

Phase I: Example Retrieval

- ☐ **Challenge: identify relevant example with intent and topological similarities**

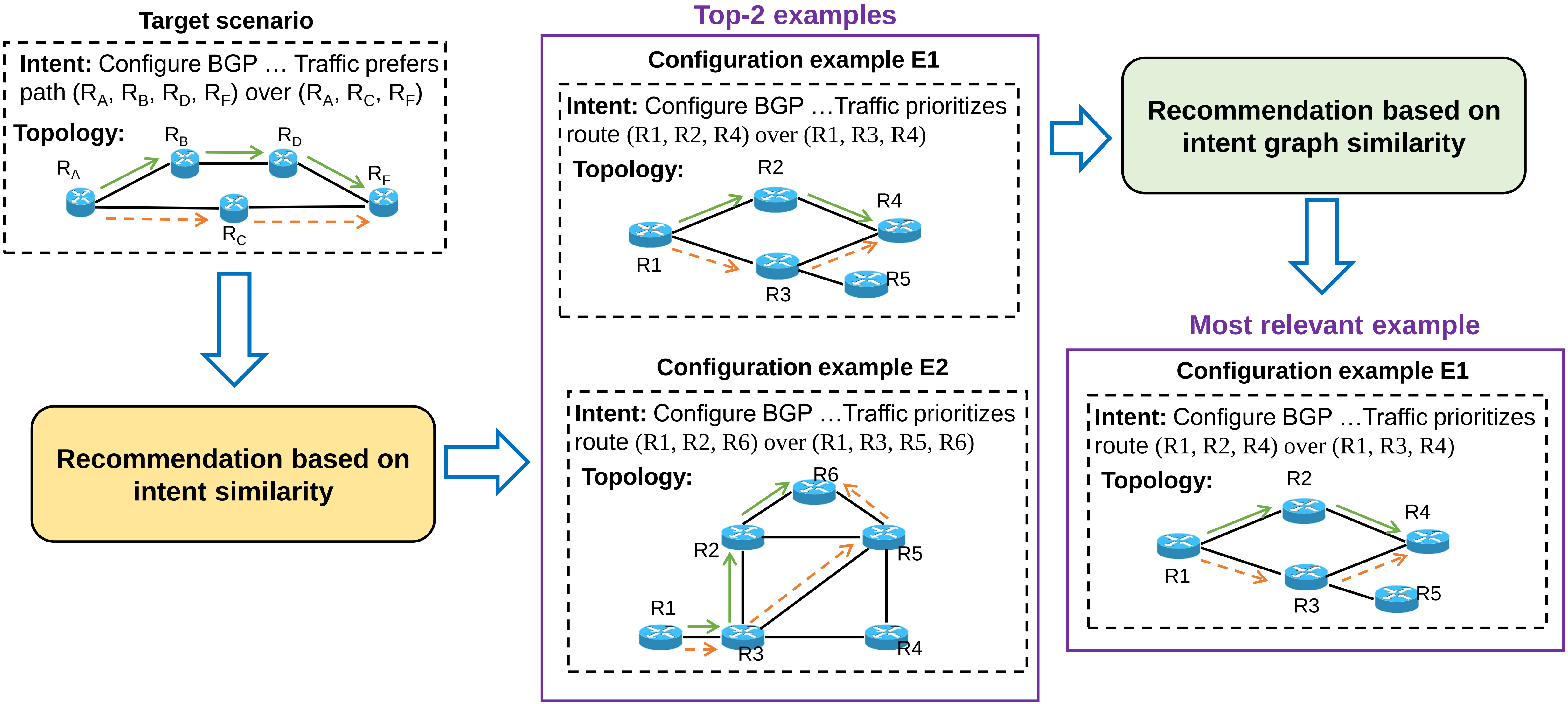
- ☐ **A novel data structure: intent graph**




Phase I: Example Retrieval
Querier adopts a two-stage recommendation strategy
Stage1
Recommendation based on intent similarity
Example intent
User intent
Sentence-BERT
Intent embedding
Similarity Measurement
Top-k examples with similar intent
Stage2
Recommendation based on intent graph similarity
Role: relay
R_B
Role: relay
R_D
R_A
Role: relay
R_F
Role: route preference
R_C
Role: destination
User intent graph
R2
Role: relay
Role: route preference
R4
R1
Role: destination
R3
Role: relay
Example intent graph
GraphSAGE
Graph embedding
Similarity Measurement
Most relevant example

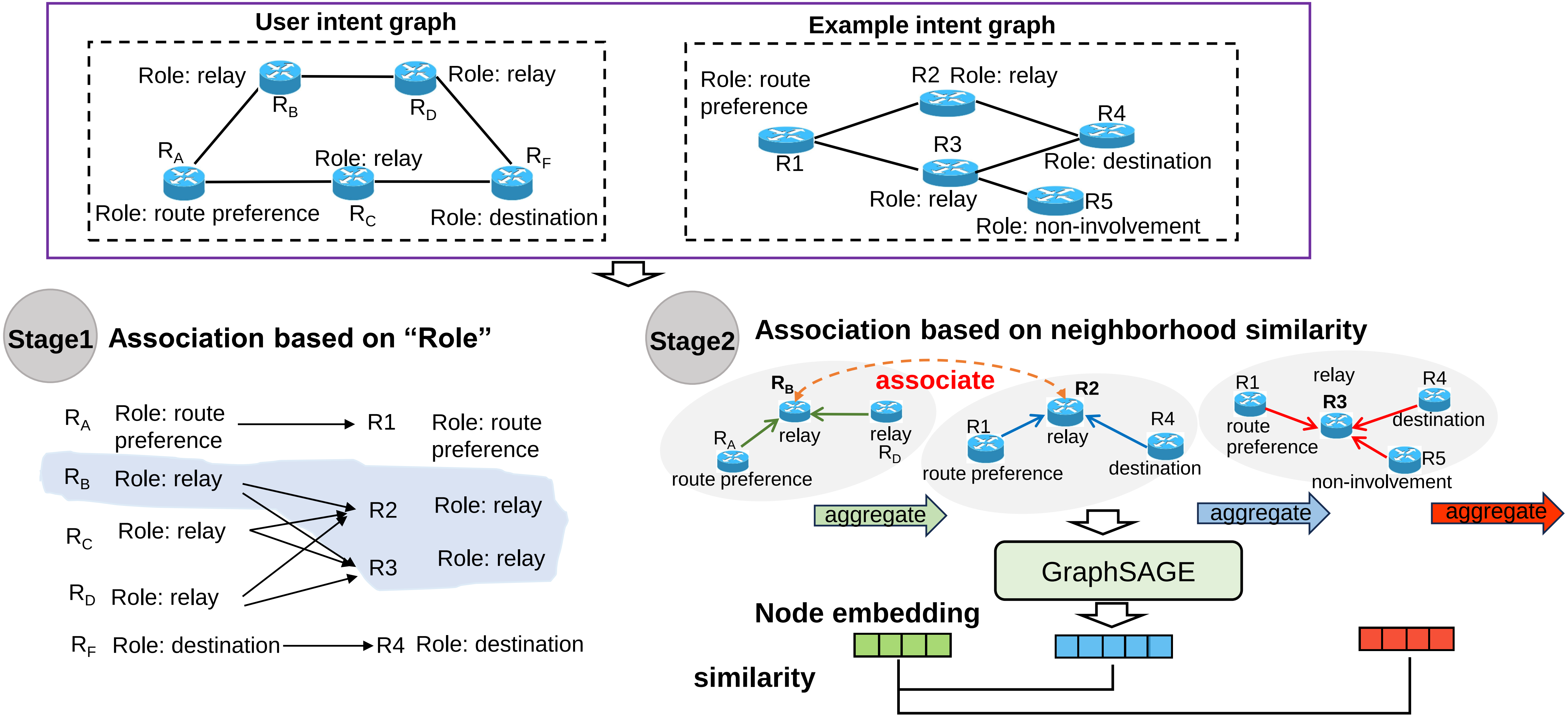
Phase I: Example Retrieval

Querier adopts a two-stage recommendation strategy



Phase II: Association relation establishment

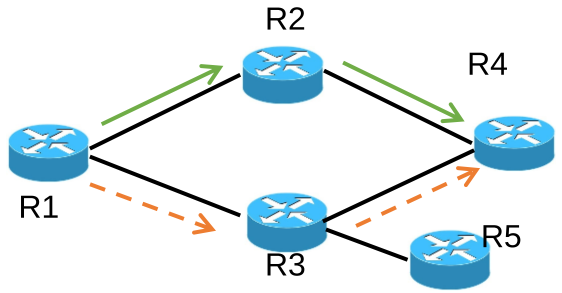
Classifier adopts a two-stage association strategy



Phase III: Configuration Generation and Verification

- **Challenge: Accelerate synthesis without compromising correctness**
- **Insight: LLMs struggle to determine policy parameters that require global reasoning**

Intent: Configure BGP … Traffic prioritizes route (R1, R2, R4) over (R1, R3, R4)




```
interface Ethernet0/1
  ip address 10.0.10.1/24
…
router bgp 10
neighbor 10.0.10.2 remote-as 20
neighbor 10.0.10.2 route-map R1 _from_ R2 in
neighbor 10.0.10.2 route-map R1_to_ R2 out
…
ip community-list 1 permit 100:0
route-map R1 _from_ R3 permit 10
   match community 1
   set local-preference 200
   set community 100:0
route-map R1 _from_ R3 permit 10
   match community 1
   set local-preference 100
   set community 100:0
route-map R1 _to_ R3 deny 10
…
```

```
interface Ethernet0/1
  ip address 10.0.10.2/24
…
router bgp 20
neighbor 10.0.20.1 remote-as 30
neighbor 10.0.20.1 route-map R2 _from_ R4 in
neighbor 10.0.20.1 route-map R2_to_ R4 out
…
ip community-list 1 permit 100:0
route-map R2 _from_ R4 permit 10
   match community 1
   set local-preference 100
   set community 100:0
…
```

Phase III: Configuration Generation and Verification

☐ Design: An efficient LLM-driven synthesis method

Stage1 **Restrict an LLM to generate templates**

prompt

[Task description] You are an expert in...

Target network scenario:

Target topology T1: {" devices": […], " links": …]}

Target Intent I1: …Traffic form router …

Configuration example:

Topology T2: {" devices": […], " links": …]}
Intent I2: ... When traffic originates from ...
Configurations:
……

Association relations … :

$R_A : U_E$, $R_B : U_E$, $R_C : U_F$, …

Format requirements for configuration template:
……
Please directly generate templates for routers ...

LLM

Templates, leaving policy parameters as symbols

Configuration for R_A :

```
interface Ethernet0/1
  ip address 10.2.10.1/24
…
router bgp 10
neighbor 10.2.10.2 remote-as 20
neighbor 10.2.10.2 route-map R_A_from_ R_B in
neighbor 10.2.10.2 route-map R_A_to_ R_B out
…
ip community-list 1 permit  ?
route-map R_A_from_ R_B permit 10
   match community 1
   set local-preference 200
   set community  ?
route-map R_A_to_ R_B ? 10
…
```

……

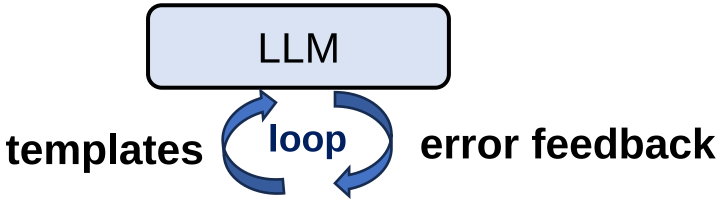
Configuration for R_D:

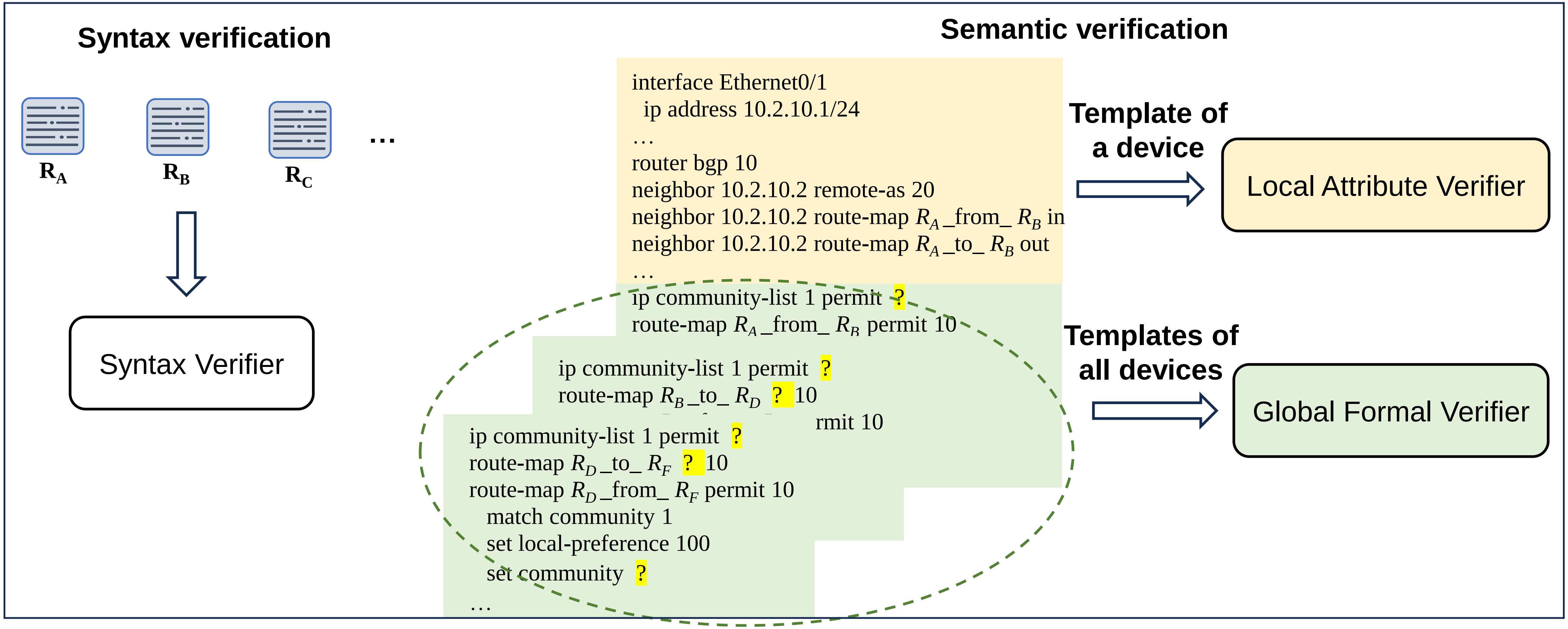
```
interface Ethernet0/2
  ip address 10.2.30.1/24
…
router bgp 50
neighbor 10.2.30.2 remote-as 40
neighbor 10.2.30.2 route-map R_D_from_ R_F in
neighbor 10.2.30.2 route-map R_D_to_ R_F out
…
ip community-list 1 permit  ?
route-map R_D_to_ R_F  ?  10
route-map R_D_from_ R_F permit 10
   match community 1
   set local-preference 100
   set community  ?
…
```

Phase III: Configuration Generation and Verification

☐ **Design: An efficient LLM-driven synthesis method**

Stage2 **Verify templates with verifiers**





Phase III: Configuration Generation and Verification

☐ Design: An efficient LLM-driven synthesis method

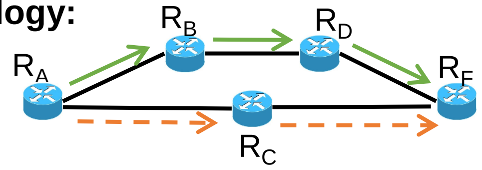
Stage3 **Fill in templates with specific parameter values**

! R_A
interface Ethernet0/1
  ip address 10.2.10.1/24
…
router bgp 10
neighbor 10.2.10.2 remote-as 20
neighbor 10.2.10.2 route-map R_A_from_ R_B in
…
ip community-list 1 permit ?
route-map R_A_from_ R_B permit 10
  match community 1
  set local-preference 200
  set community ?
route-map R_A_to_ R_B ? 10
…

…

! R_D:
interface Ethernet0/2
  ip address 10.2.30.1/24
…
router bgp 50
neighbor 10.2.30.2 remote-as 40
neighbor 10.2.30.2 route-map R_D_from_ R_F in
…
ip community-list 1 permit ?
route-map R_D_to_ R_F ? 10
route-map R_D_from_ R_F permit 10
  match community 1
  set local-preference 100
  set community ?
…

Topology:



Intent: Traffic prefers path (R_A, R_B, R_D, R_F) over (R_A, R_C, R_F)

Formal Synthesizer

! R_A
interface Ethernet0/1
  ip address 10.2.10.1/24
…
ip community-list 1 permit 100:1
route-map R_A_from_ R_B permit 10
  match community 1
  set local-preference 200
  set community 100:1
route-map R_A_to_ R_B deny 10
…

⋮

! R_D:
interface Ethernet0/2
  ip address 10.2.30.1/24
…
neighbor 10.2.30.2 route-map R_D_from_ R_F in
…
ip community-list 1 permit 100:1
route-map R_D_to_ R_F deny 10
route-map R_D_from_ R_F permit 10
  match community 1
  set local-preference 100
  set community 100:1
…

Evaluation

- How does CEGS compare to SOTA LLM-based synthesis system COSYNTH [HotNets' 23] and NETBUDDY [CoNEXT' 24] ?
- How does CEGS perform in achieving various set of intents across different network scales?
- How do the core components of CEGS perform?

Evaluation: setup

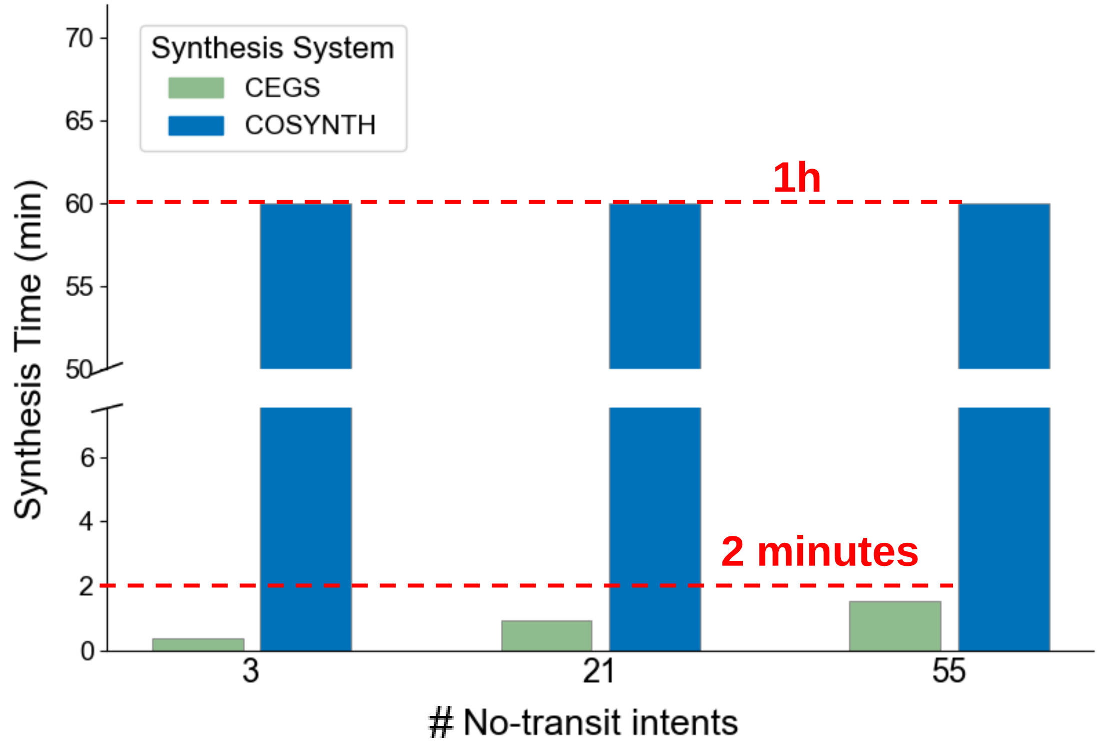
- **Target scenarios:**
  - ✓ **Topologies:** 20 real-world topologies from Internet Topology Zoo dataset, ranging from 20 to 1094 routers
  - ✓ **Intents:** Six types of prevailing routing intents, covering Static, OSPF, and BGP protocols, including static path (Simple), load-balancing (ECMP), Any-path, path-preference (Ordered), and No-transit

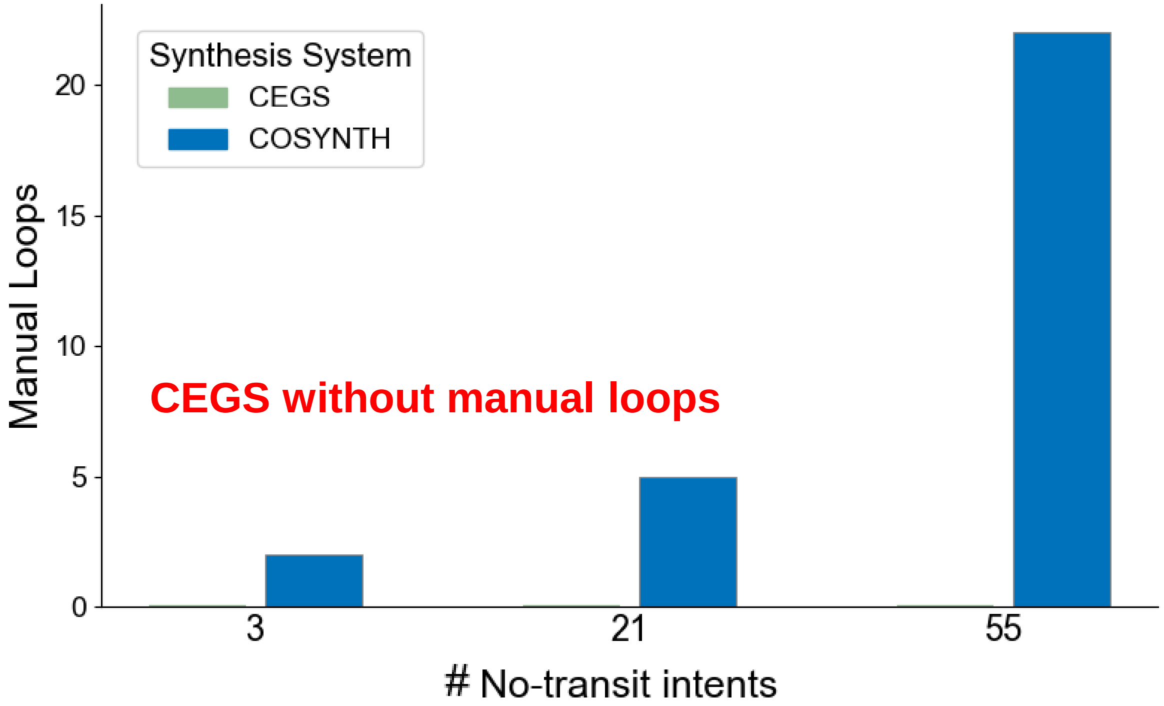
- **Configuration examples corpus:** 300 configuration examples

- **LLM selection:** GPT-4o

- **Metrics:** loops, synthesis time

Evaluation: Comparison to COSYNTH [HotNets' 23]





CEGS successfully synthesize configurations within 2m

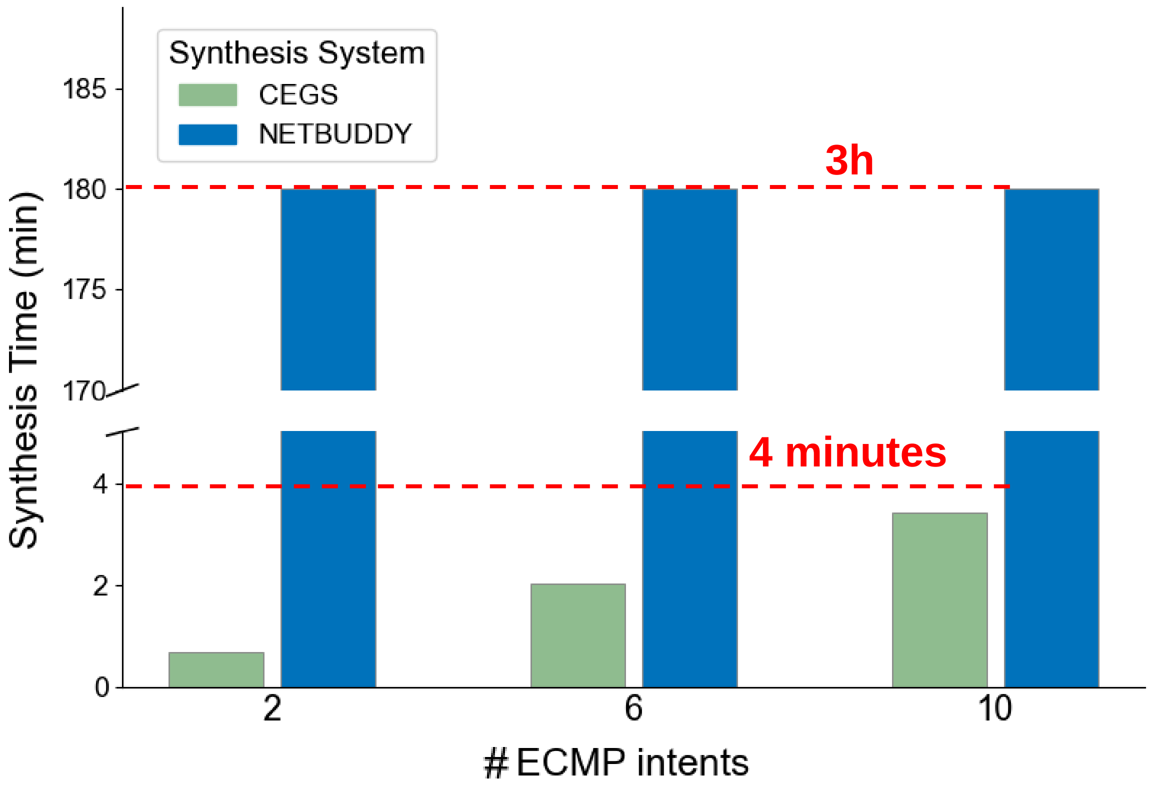
vs.

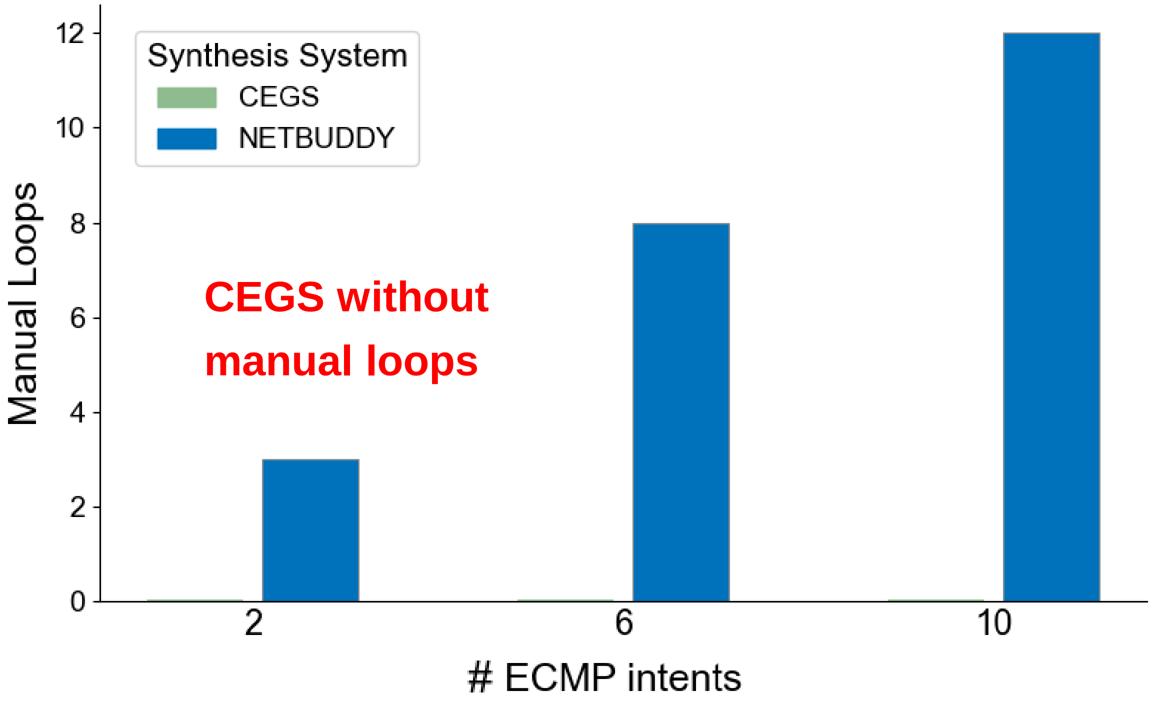
COSYNTH fails to synthesize configurations within 1h

CEGS does not require human involvement

COSYNTH requires human involvement

Evaluation: Comparison to NETBUDDY [CoNEXT' 24]





CEGS successfully synthesize configurations within 4m

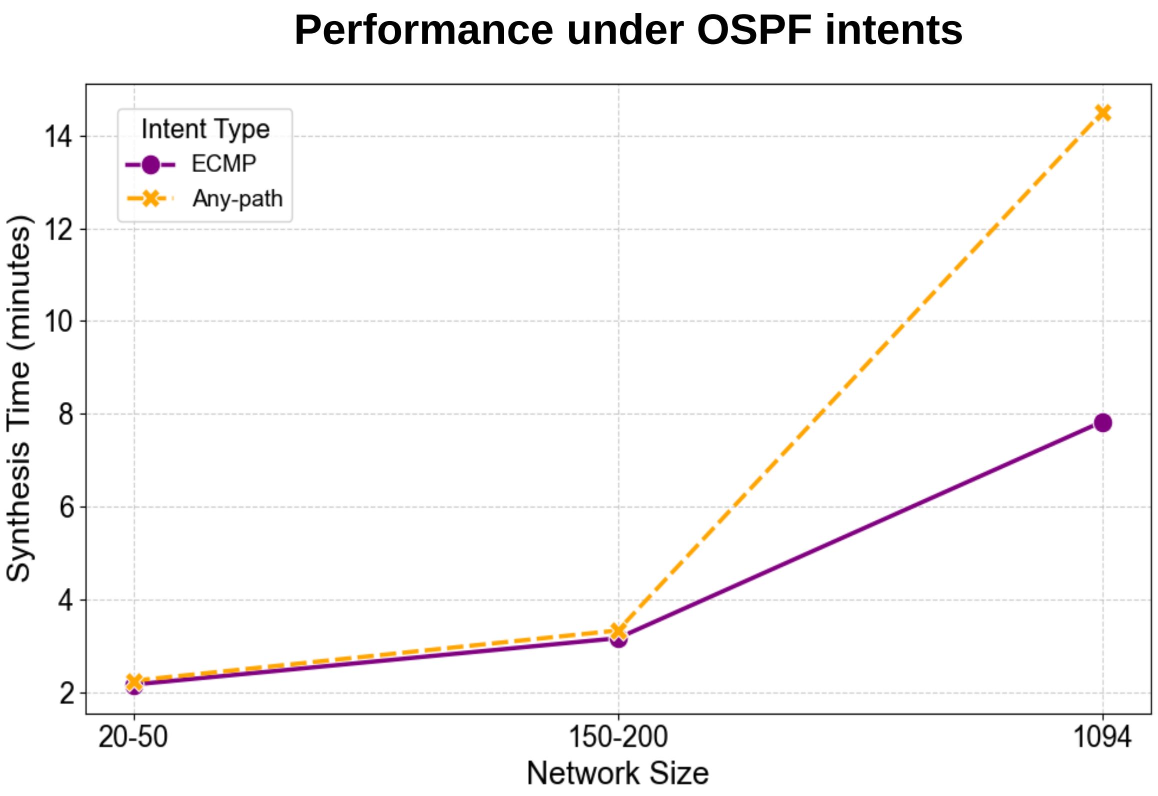
vs.

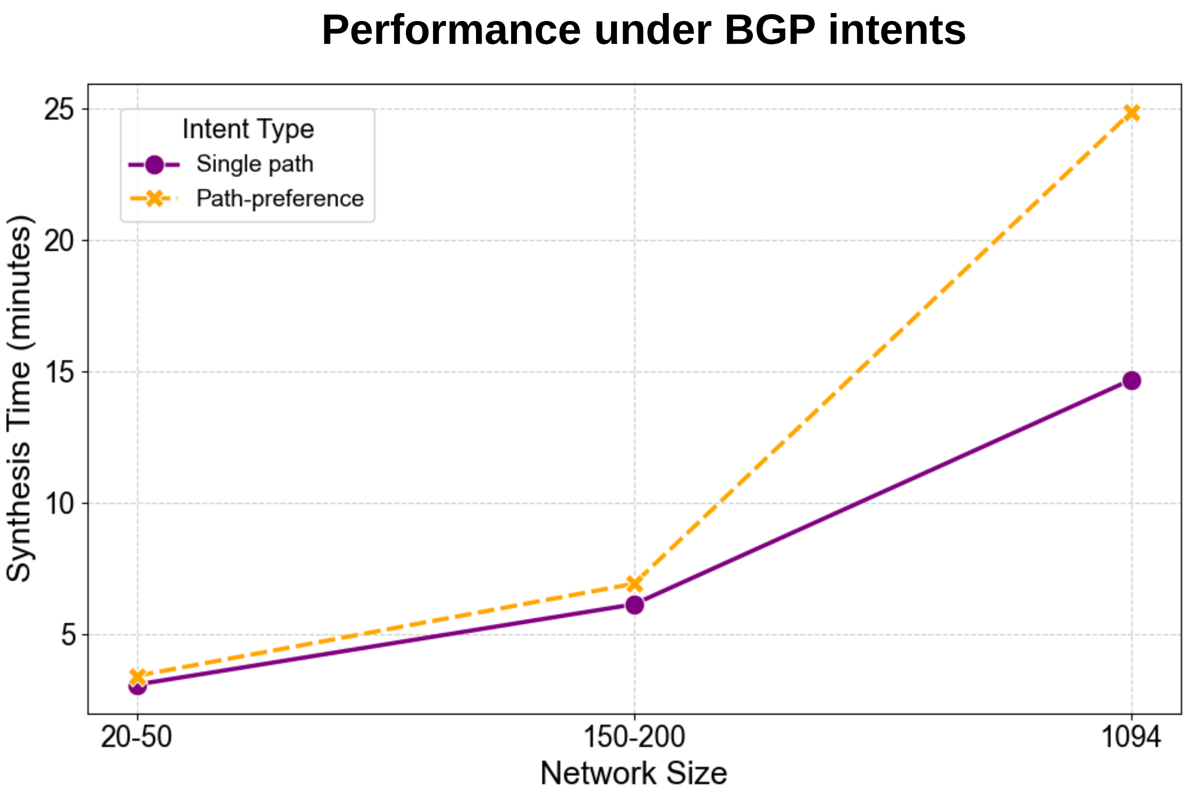
NETBUDDY fails to synthesize configurations within 3h

CEGS does not require human involvement

NETBUDDY requires human involvement

Evaluation: performance across different network scales





Conclusion: synthesis time is proportional to topology size

Evaluation: performance under complex scenarios

Table 5: CEGS performance in synthesizing configurations for various combinations of multiple types of intents.

Scenarios	Protocols	Intent type	#intents	Loops	Synthesis time (SMT time)	Verified
Scen1	BGP	No-transit Ordered	5 5	13	7m30s (45s)	✓
Scen2	Static OSPF	static route ECMP	5 5	3	4m36s (30s)	✓
Scen3	Static OSPF BGP	static route ECMP Ordered	5 5 5	11	10m30s (1m42s)	✓

Conclusion: CEGS can synthesize correct configurations that satisfy multi-type intents simultaneously

Evaluation: Ablation Study on core components of CEGS

Table 6: Ablation study results on six core components.

Querier	Classifier	Syntax Verifier	LAV	GFV	Formal Synthesizer	Loops
√	√	√	√	√	√	10
x	x	√	√	√	√	300(U)
√	x	√	√	√	√	35
√	√	x	√	√	√	300(U)
√	√	√	x	√	√	300(U)
√	√	√	√	x	√	300(U)
√	√	√	√	√	x	50

Conclusion:

- ✓ **Without Querier, CEGS fails to** synthesize configurations
- ✓ **Without Classifier,** CEGS synthesize configurations **at a slower rate**
- ✓ **Without Verifiers, CEGS fails to** synthesize configurations
- ✓ **Without Formal Synthesizer,** CEGS synthesize configurations **at a slower rate**

Conclusions

- **CEGS can generalize examples to arbitrary topologies**

- **CEGS features:**
  - ✓ **A GNN-based Querier** to identify relevant examples
  - ✓ **A GNN-based Classifier** to establish association relations
  - ✓ **An efficient LLM-driven synthesis method**

- **CEGS can synthesize correct configuration without human involvement**. In contrast, SOTA **LLM-based** synthesizers **are more than 30 times slower than CEGS on average, even with human involvement.**