Projekt 1

AHMAD ZIADAT
ALI ALMAZAAL
AMIN BOUALI

Modifizierte Version der Methode __demands_first_waypoints

Kürzeste Pfade-Cache:

- Redundante Berechnungen vermeiden

Mögliches Waypoints-Set:

- Set von potenziellen Waypoints
- Nachbarn und Kapazitäten
- Reduzierter Suchraum

Top-k Nachbarnauswahl:

- Top-k Nachbarn-Heuristik
- Höchste Kapazität
- Anzahl der Wegpunkte reduziert

Modifizierte Version der Methode __demands_first_waypoints

Vergleich der Leistung: Original vs. Modifizierte Methode

- Verbesserte Berechnungseffizienz
- Maximale Auslastung: variierend

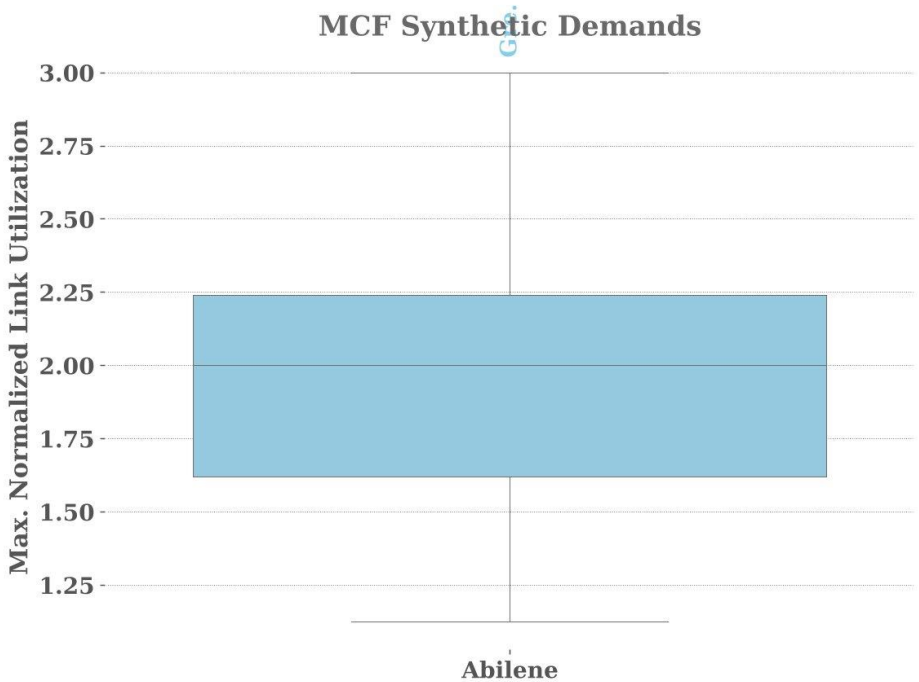
Potenzielle weitere Forschungsbereiche und Verbesserungen

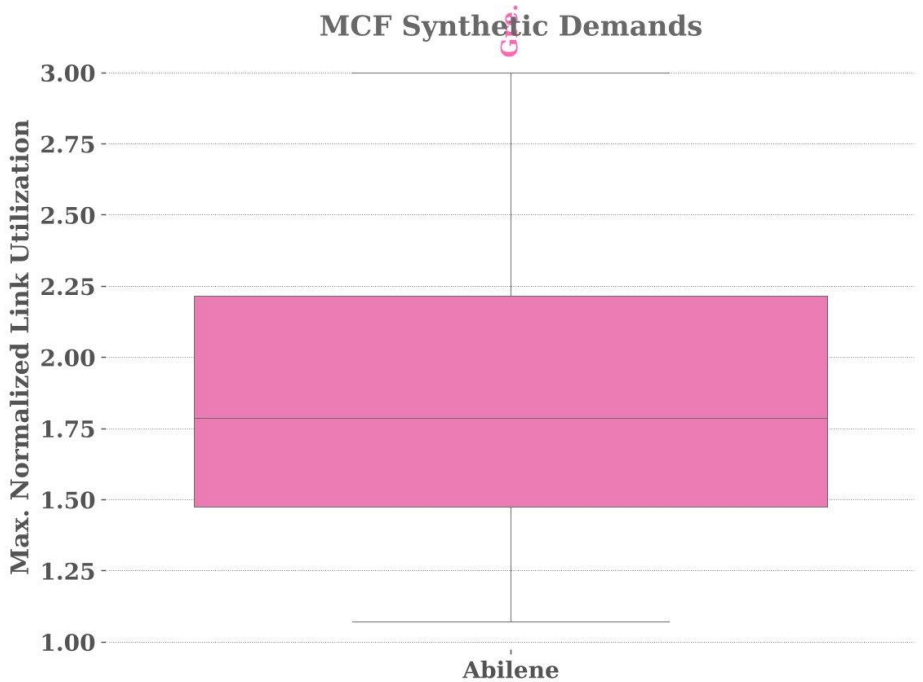
- Berücksichtigung zusätzlicher Faktoren:

1. **Auslastung der Nachbarn**
2. **Kosten der Verbindung**
3. **Netzwerklatenz**

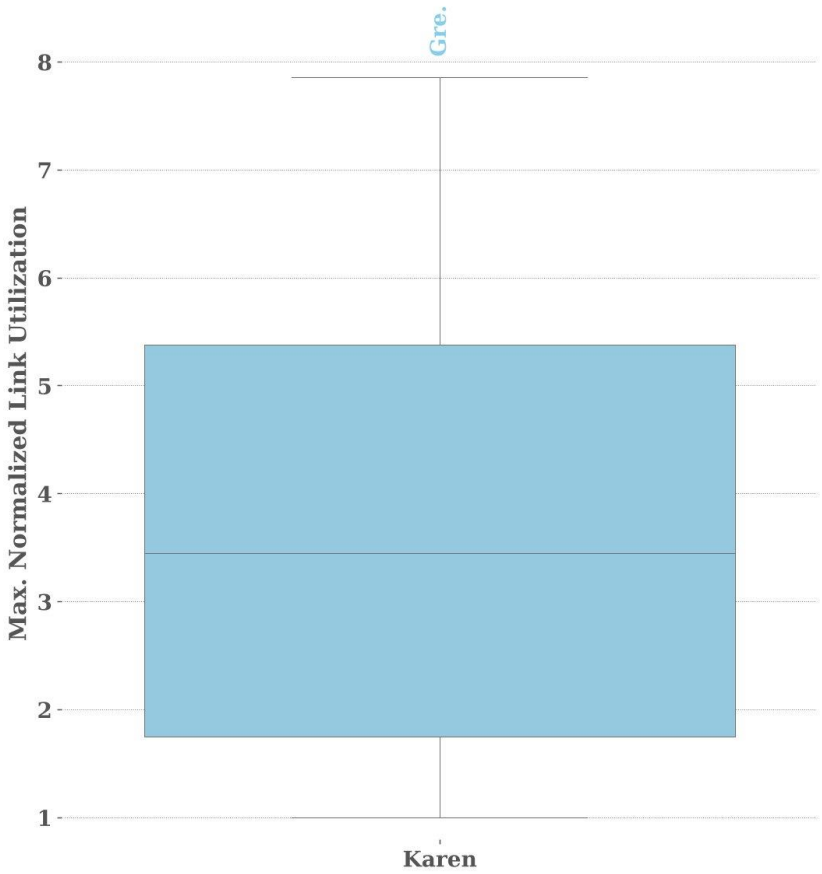
- Einbeziehung von Multi-Path-Routing

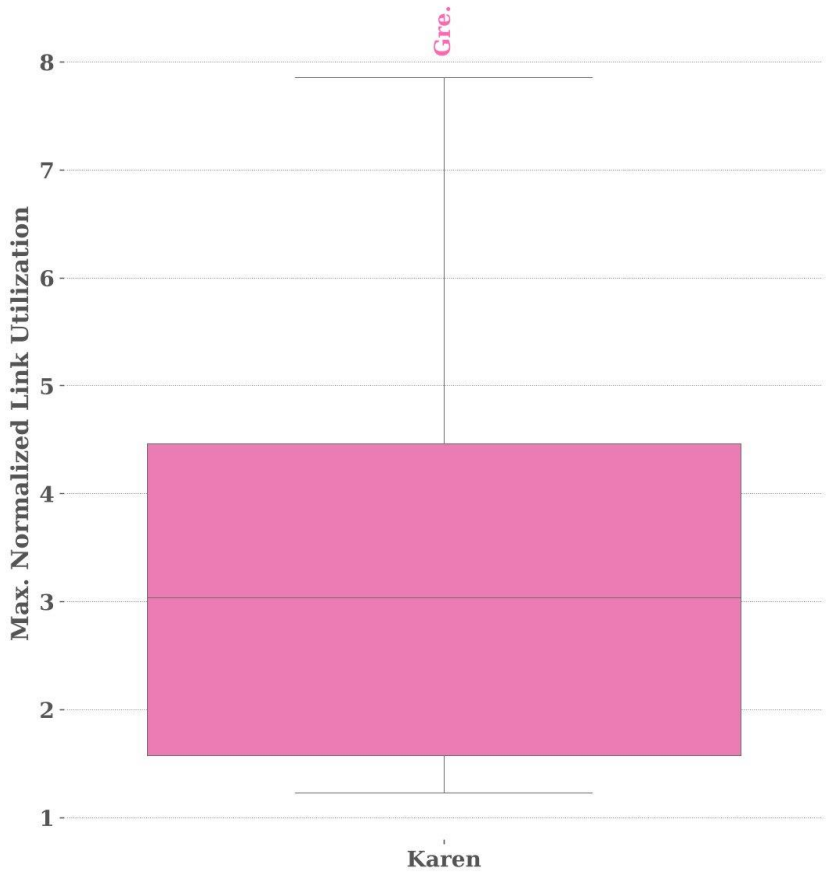
Modifizierte Version der Methode __demands_first_waypoints



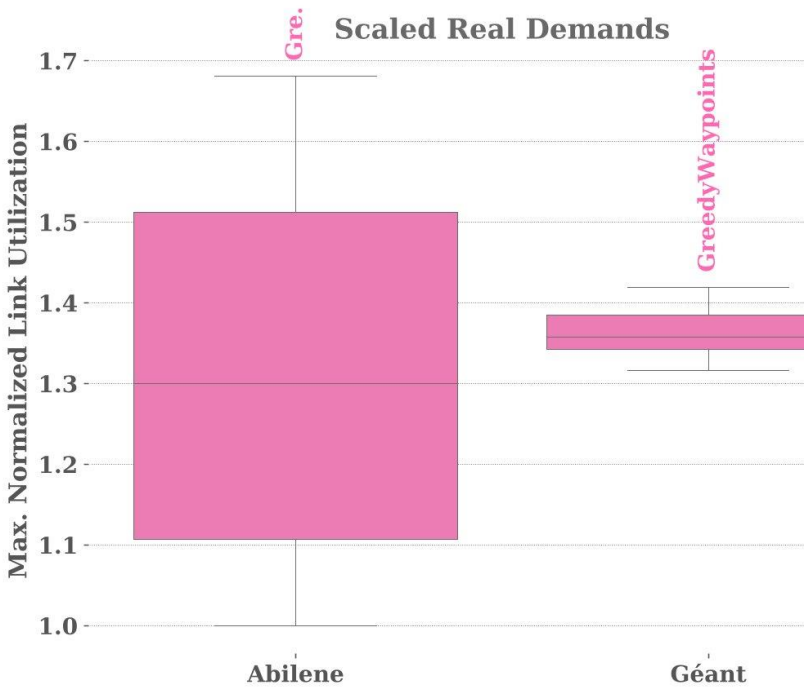


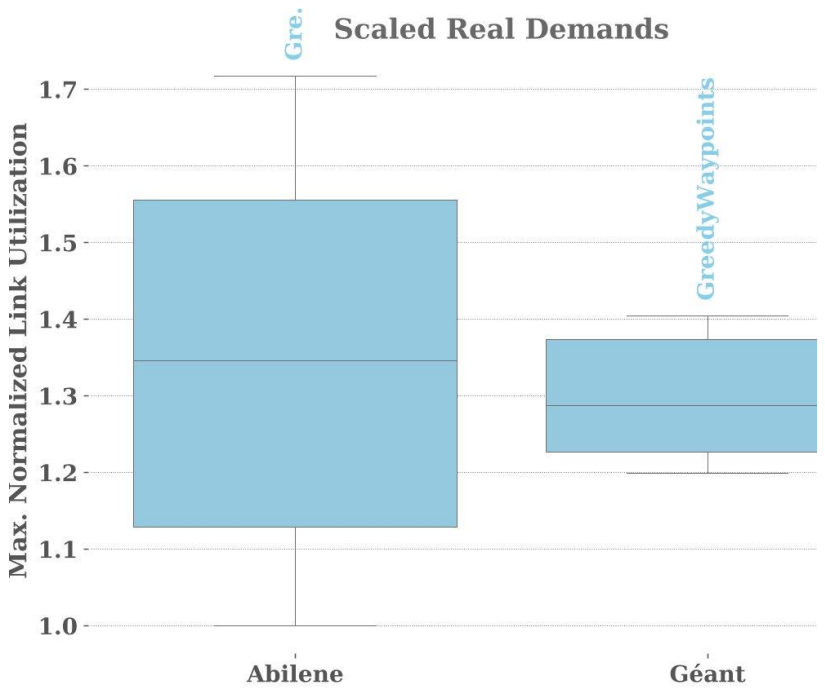
MCF Synthetic Demands





Modifizierte Version der Methode __demands_first_waypoints





Replikation Gruppe 1:Naveed

- Anaconda installiert
- Conda activate
- Dependencies installiert
- SequentialCombination2 aus heur_ospf_weights und uniform_weights
- Verbesserte Berechnungseffizienz: variierend
- Maximale Auslastung: ist schlechter geworden im Vergleich zu SequentialCombination

Erweiterte Version des Inverse-Capacity-Algorithmus

Ziadat Ahmad

25. Mai 2023

Einleitung

Die erweiterte Version baut auf Inverse-Capacity-Algorithmus auf und enthält zusätzliche Funktionen und Logik, um mit bestimmten Szenarien umzugehen

Änderungen und Hinzufügungen im Code (Teil 1)

- Die Variable self.threshold zur Berechnung der Link-Auslastung

Sie wird verwendet. um zu überprüfen. ob die maximale Auslastung eines Links den festgelegten Schwellenwert überschreitet

```
self.threshold= kwargs.get("threshold", 1.0)
```

- Die Methode getlinkutilization() zur Berechnung der Link-Auslastung

eine bessere überwachung und Handhabung von überlasteten Verbindungen

$$\texttt{max}_u tilization = max([self.get_l ink_u tilization(link)$$

Änderungen und Hinzufügungen im Code (Teil 2)

- Notfallmaßnahmen und Logik für alternative Routen

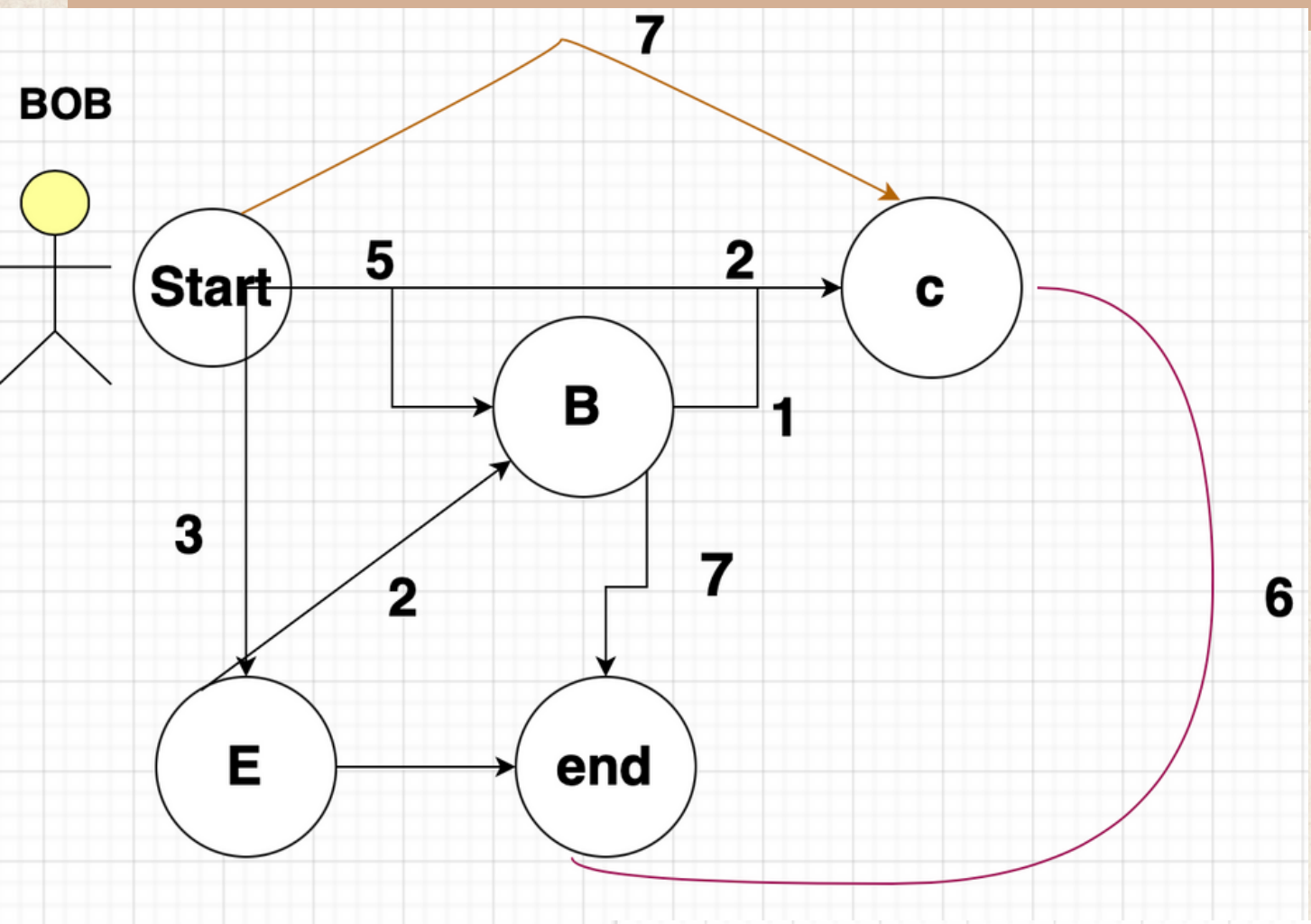
$$\texttt{if } \texttt{max}_u tilization > self.threshold :$$

- Die Methode computealternativeroutes(self) zur Berechnung alternativer Routen

Dadurch können Ausfallsicherheit und Leistungsoptimierung verbessert werden.

```
Beispiel: Füge eine alternative Route für
jede Nachfrage hinzu, indem du den kürzesten
Pfad umkehrst
```

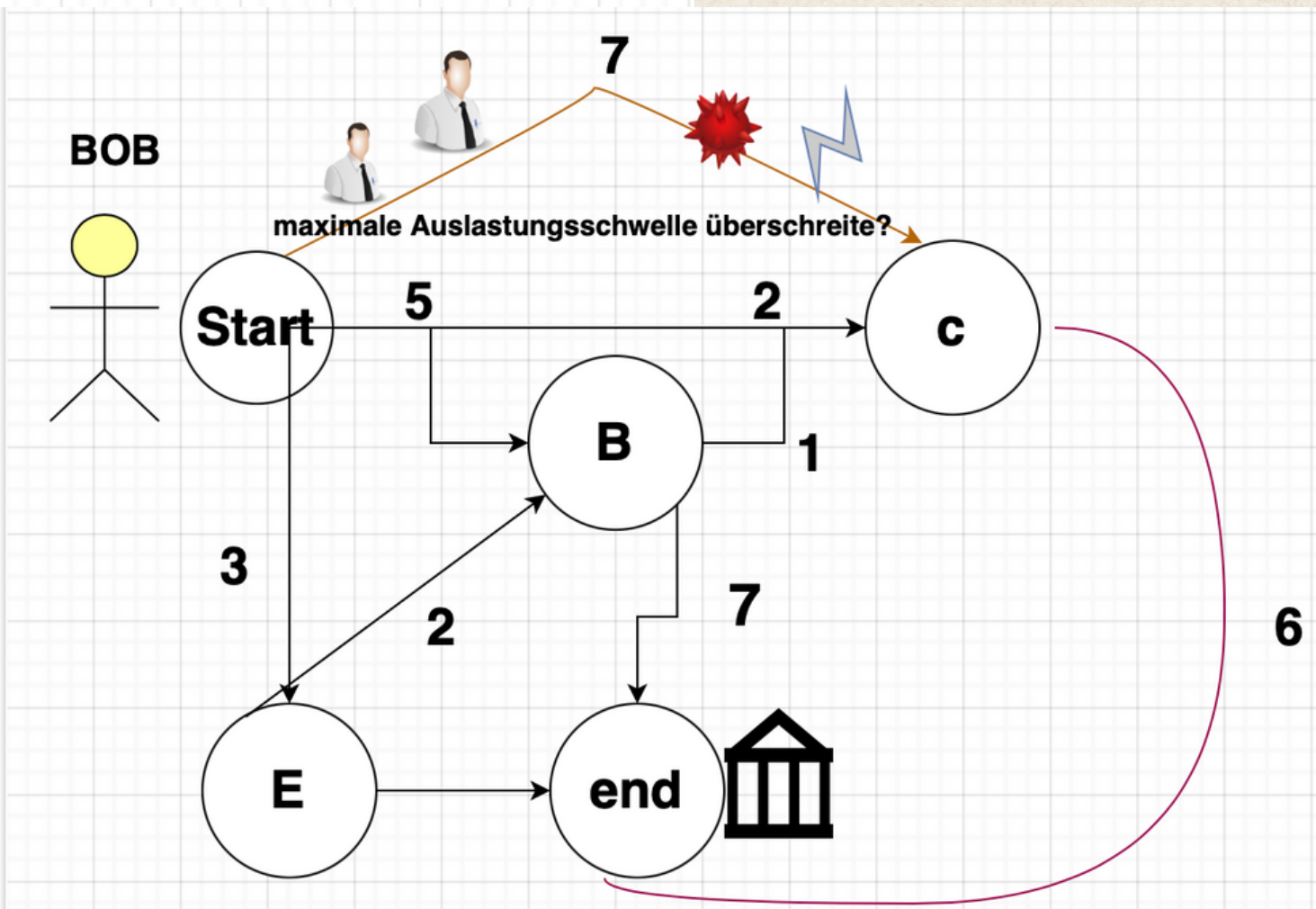
EINFACHE BEISPIELE



INVERSE CAPACITY-ALGORITHMUS

VS

ERWEITEREINVERSECAPACITY-ALGORITHMUS:



Verbesserung der maximalen normalisierten Auslastung

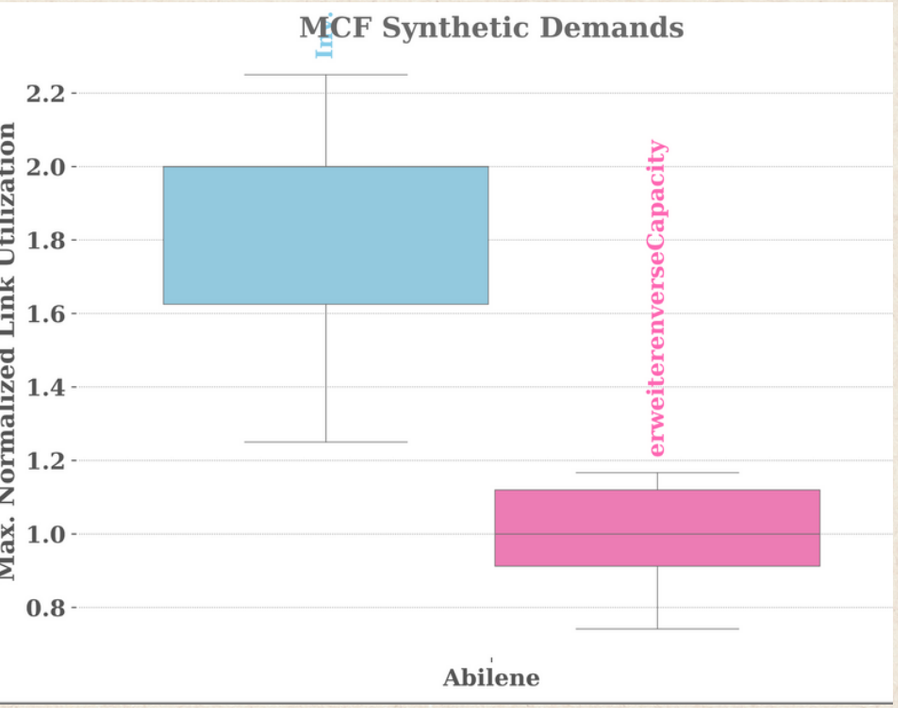


Fig. 4

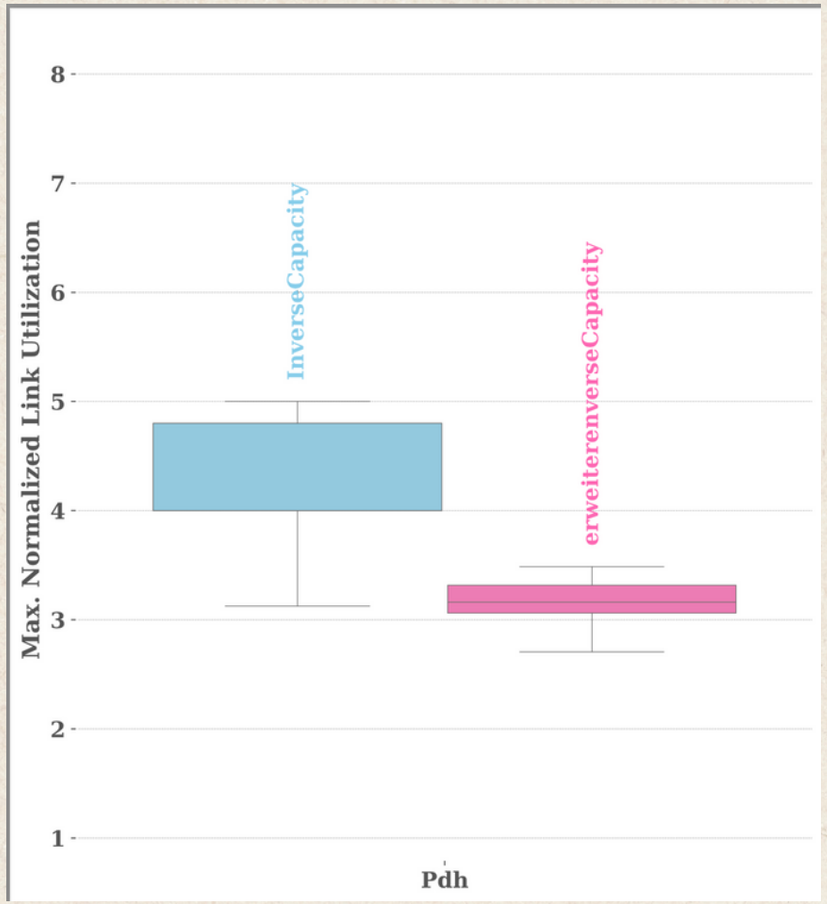


Fig. 3

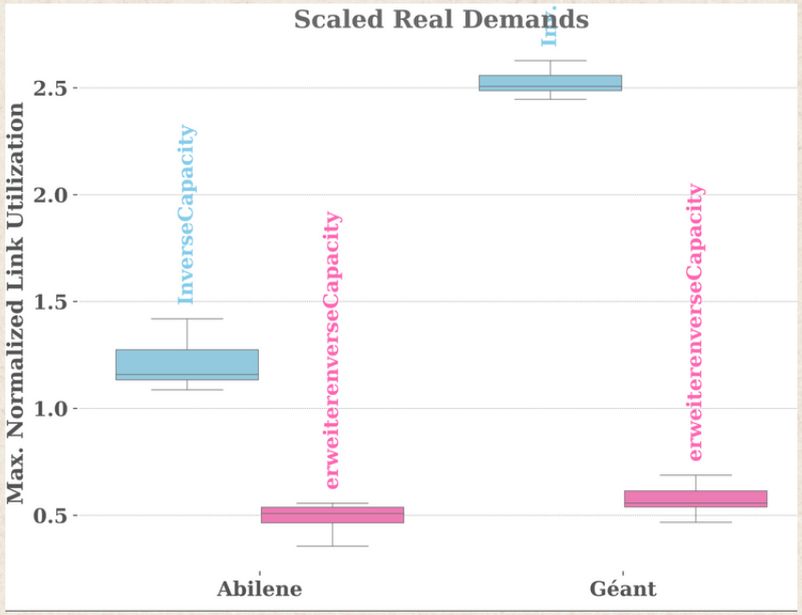


Fig.5

Auswirkungen und Vorteile der erweiterten Version (Teil 1)

- Bessere Netzwerkauslastung: Durch die Einbeziehung der Link-Auslastung und der maximalen Auslastungsschwelle kann die erweiterte Version des Algorithmus die Netzwerkauslastung besser überwachen und optimieren.
- Robustheit gegenüber Ausfällen: Die Berechnung alternativer Routen und die Implementierung von Notfallmaßnahmen verbessern die Robustheit des Algorithmus gegenüber Verbindungs- oder Knotenausfällen.

Auswirkungen und Vorteile der erweiterten Version (Teil 2)

- Leistungsverbesserung: eine verbesserte Leistung des Algorithmus, indem Engpässe reduziert und die Ressourcennutzung optimiert werden.
- Skalierbarkeit: größeren Netzwerken effizient umzugehen und die Skalierbarkeit zu verbessern.

Fazit und zukünftige Entwicklungen

- Zusammenfassung der erweiterten Version: Die erweiterte Version des Inverse-Capacity-Algorithmus bietet eine verbesserte Netzwerkauslastung, Robustheit gegenüber Ausfällen und Leistungsverbesserungen.
- Erfüllung der gesteckten Ziele: Die erweiterte Version erfüllt die gesteckten Ziele, indem sie zusätzliche Funktionen und Logik hinzufügt, um spezifische Szenarien im Netzwerkmanagement zu adressieren.