



MASTERARBEIT | MASTER'S THESIS

Titel | Title

Algorithm Engineering and Empirical Evaluation for Fast
Rerouting with Multiple Failures

verfasst von | submitted by

Philipp Zabka BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree
programme code as it appears on the
student record sheet:

UA 066 921

Studienrichtung lt. Studienblatt | Degree
programme as it appears on the student
record sheet:

Masterstudium Informatik

Betreut von | Supervisor:

Univ.-Prof. Dipl.-Math. Dr. Peter Reichl Privatdoz.

Acknowledgements

I want to express my sincere gratitude towards Univ.-Prof. Dipl.-Math. Dr. Peter Reichl, Privatdoz., Univ.-Prof. Dr. Stefan Schmid and Univ.-Prof. Dr. Dr. Klaus-Tycho Förster for their expertise, constant support, ideas and feedback that made this thesis possible.

I also want to kindly thank Wenkai Dai, B.Sc. M.Sc., for supporting me throughout the journey of algorithm implementation.

Finally, I want to thank my family and friends, for their unwavering encouragement and support.

Abstract

Due to the continuously increasing traffic and reliance on data in our evolving digital society, modern communication networks not only need to scale in size, but also need to ensure a high routing availability and respond swiftly to unforeseen link failures, to circumvent information flow and maintain network resilience. This stands in contrast to many contemporary networks which only feature routing protocols such as OSPF or IS-IS. These algorithms are far too slow for modern demands in the case link failures occur as recomputation of routing tables happens at a global scale. To mitigate the slow response time of such traditional protocols, most modern routing devices additionally feature fast failover routing algorithms for a swifter response. Fast failover algorithms offer local rerouting to preinstalled alternatives paths, making recomputation much quicker.

In this work, we conduct an extensive empirical study of the most recent advances in *state-of-the-art* fast failover algorithms, namely *SquareOne*, *Feigenbaum* and *TwoResilient*, which all provide guarantees for k simultaneous link failures. In particular, *SquareOne*, discussed by Foerster et al. [FPST19], provides a resilience against $k - 1$ failures in a k -connected network. *Feigenbaum*, presented by Feigenbaum et al. [FGP⁺12], promises a resilience against $k = 1$ link failures and *TwoResilient* against $k = 2$ link failures in general networks. For the latter algorithm, *TwoResilient*, so far, only an abstract mathematical description was available. We analyze the algorithmic details and design a new implementation that efficiently solves various subproblems in practice inherent to the global routing scheme discussed in Dai et al. [DFS23]. Moreover, we design and implement extensions to the *TwoResilient* algorithm to heuristically increase its routing performance and discuss details as well as challenges we have faced during this process. Our simulations, conducted on various synthetic and real world networks with two link failure models RANDOM and CLUSTER, reveal the strengths and weaknesses of these algorithms, which provide multiple interesting insights to increase their effectiveness on specific network topologies. In our evaluation, the DAG based *Feigenbaum* algorithm proves to achieve the best performance in routing success, stretch and precomputation efficiency. In terms of routing success *TwoResilient* is on par with *Feigenbaum* and on some network topologies such as Erdős–Rényi even shows a slight advantage in performance. The *SquareOne* algorithm shows the least effective metrics in all our tests.

Keywords

Fast Failover, Routing Schemes, Network Resilience, Fault Tolerance, Data Plane

Kurzfassung

Aufgrund des kontinuierlich steigenden Informationsverkehrs und der zunehmenden Abhängigkeit von Daten in unserer sich entwickelnden digitalen Gesellschaft, müssen moderne Kommunikationsnetzwerke nicht nur in der Größe skalieren, sondern auch eine hohe Verfügbarkeit gewährleisten und somit schnell auf unvorhergesehene Verbindungsfehler oder Ausfälle reagieren, um den Informationsfluss aufrechtzuerhalten und die Netzwerkresilienz zu sichern.

Dies steht im Gegensatz zu vielen gegenwärtigen Netzwerken, die nur Routingprotokolle wie OSPF oder IS-IS verwenden. Diese Algorithmen sind für moderne Anforderungen im Falle von Verbindungsfehlern viel zu langsam, da die Neuberechnung von Routingtabellen global erfolgt. Um die langsame Reaktionszeit solcher traditionellen Protokolle zu mildern, verfügen die meisten modernen Router zusätzlich über schnelle Failover Routingalgorithmen für eine schnellere Reaktionszeit. Failover-Algorithmen bieten eine lokale Umleitung zu vorinstallierten alternativen Pfaden und daher auch eine schnellere Neuberechnung von Routingtabellen.

In dieser Arbeit führen wir eine umfangreiche empirische Studie über die neuesten Entwicklungen bei hochmodernen Fast-Failover-Algorithmen *SquareOne*, *Feigenbaum* und *TwoResilient* durch. Das besondere an diesen Algorithmen ist, dass alle Garantien für k -simultane Verbindungsfehler bieten. Insbesondere hat *SquareOne*, diskutiert in Foerster et al. [FPST19], eine Widerstandsfähigkeit gegen $k - 1$ Fehler in einem k -verbundenen Netzwerk, während *Feigenbaum*, präsentiert in Feigenbaum et al. [FGP⁺12], gegen $k = 1$ Verbindungsfehler und *TwoResilient* gegen $k = 2$ Verbindungsfehler in allgemeinen Netzwerken absichert. Für den *TwoResilient*-Algorithmus lag bisher nur eine abstrakte mathematische Beschreibung vor, deswegen analysieren wir die algorithmischen Details und entwickeln eine neue Implementierung, die verschiedene in der Praxis inhärente Teilprobleme des globalen Routingschemas effizient löst, wie in Dai et al. [DFS23], diskutiert. Darüber hinaus entwerfen und implementieren wir Erweiterungen für den *TwoResilient*-Algorithmus, um seine Routingleistung heuristisch zu verbessern, und diskutieren Details sowie Herausforderungen, denen wir während dieses Prozesses begegnet sind.

Unsere Simulationen, wurden an verschiedenen sowohl synthetischen als auch realen Netzwerken mit den Kantensausfallmodellen RANDOM und CLUSTER durchgeführt. Unsere Ergebnisse zeigen die Stärken und Schwächen dieser Routingalgorithmen auf und liefern interessante Einblicke zur Steigerung ihrer Wirksamkeit auf spezifischen Netzwerktopologien. In unserer Studie erweist sich der DAG-basierte *Feigenbaum*-Algorithmus als leistungsfähigster Algorithmus in Bezug auf Routerfolg, Stretch und Vorberechnungsseffizienz. *TwoResilient* steht in Bezug auf den Routerfolg gleichauf mit *Feigenbaum* und verzeichnet sogar einen leichten Vorteil auf Netzwerken wie unter anderem Erdős–Rényi. Das Schlusslicht in allen unseren Tests bildet *SquareOne*.

Contents

Acknowledgements	i
Abstract	iii
Kurzfassung	v
List of Tables	ix
List of Figures	xi
List of Algorithms	xix
Listings	xxi
1. Introduction	1
1.1. Contributions	2
1.2. Organization	3
2. Related Work	5
2.1. Fast Failover Advances and Limits	5
2.2. Arborescence Routing Advances	6
2.3. Routing Challenges with Multiple Failures	6
2.4. Evolving Fast Failover Solutions	7
3. Preliminaries	9
3.1. Planes in Networks	9
3.1.1. Data Plane	9
3.1.2. Control Plane	9
3.2. Fast Failover Algorithms	9
3.2.1. Network Structure	11
3.2.2. SquareOne	11
3.2.3. Feigenbaum	12
4. The TwoResilient Algorithm	15
4.1. 2-Resilient Routing Scheme	15
4.2. Implementation Details and Extensions	16
5. Methodology	21
5.1. Network Topologies	21

5.2. Experimental Setup	23
6. Experimental Evaluation	25
6.1. Hardware and Software Specification	25
6.2. Experimental Evaluation in the Topology Zoo	26
6.2.1. Randomized Failures	26
6.2.2. Clustered Failures	29
6.2.3. Precomputation Performance	32
6.2.4. Discussion	33
6.3. Experimental Evaluation of Rings of Cliques	35
6.3.1. Randomized Failures	35
6.3.2. Clustered Failures	37
6.3.3. Precomputation Performance	38
6.3.4. Discussion	38
6.4. Experimental Evaluation of Erdős–Rényi Networks	41
6.4.1. Random Failures	41
6.4.2. Clustered Failures	42
6.4.3. Precomputation Performance	43
6.4.4. Discussion	44
6.5. Experimental Evaluation of Regular Networks	46
6.5.1. Random Failures	46
6.5.2. Clustered Failures	47
6.5.3. Precomputation Performance	48
6.5.4. Discussion	48
6.6. Experimental Evaluation of Rocketfuel Networks	51
6.6.1. Random Failures	51
6.6.2. Clustered Failures	52
6.6.3. Precomputation Performance	53
6.6.4. Discussion	53
6.7. Recapitulation and Discussion of Evaluation Results	55
7. Conclusion and Future Work	57
7.1. Conclusion	57
7.2. Future Work	58
Bibliography	59
A. Appendix	69

List of Tables

1.1. Average success rate of algorithms for the first 30% of the total failure ranges across different topologies for RANDOM and CLUSTER link failure models described in Section 5.2. The highlighted table cells show the highest success rates on each topology and link failure model.	3
2.1. Summary of the resiliency of routing schemes. Highlighted schemes are discussed in this work.	7

List of Figures

3.1.	Directed Acyclic Graph (DAG) with root node 1 and target node t	12
6.1.	Similar to [FKP ⁺ 21], this figure depicts the average pathfinding effectiveness on the Topology Zoo networks for a range of random disrupted links, where <i>Feigenbaum</i> and <i>TwoResilient</i> algorithms show nearly identical performance . The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 50 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.	26
6.2.	Similar to [FKP ⁺ 21], this chart illustrates the average pathfinding effectiveness on the Topology Zoo networks, based on the subset of nodes that remain connected after 10 random disrupted links, across 50 individual trials. Both <i>Feigenbaum</i> and <i>TwoResilient</i> algorithms achieve success rates close to the ideal value, denoted as p . The threshold signifies the average fraction of nodes that remain connected to the destination sub-graph after failures, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y	27
6.3.	Similar to [FKP ⁺ 21], this chart illustrates the comparison of the relative average pathfinding effectiveness of all algorithms against the <i>TwoResilient</i> algorithm on the Topology Zoo networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. For randomized link failures, in the majority of test cases, <i>Feigenbaum's</i> performance lies near to <i>TwoResilient's</i> performance	28
6.4.	This figure depicts the average normalized stretch on the Topology Zoo networks for a range of random disrupted links, where <i>Feigenbaum</i> outperforms the other two algorithms by a large margin . The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 50 individual trials carried out for each failure mark.	29

List of Figures

- 6.5. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Topology Zoo networks for a range of clustered disrupted links, **where *Feigenbaum* and *TwoResilient* algorithms show nearly identical performance**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 50 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. 30

- 6.6. Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the Zoo topology networks, based on the subset of nodes that remain connected after 10 clustered disrupted links, across 50 individual trials. **Both *Feigenbaum* and *TwoResilient* algorithms achieve success rates close to the ideal value, denoted as p** . The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y 31

- 6.7. Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *TwoResilient* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. **For clustered link failures, in the majority of test cases, *Feigenbaum*'s performance lies near to *TwoResilient*'s performance**. 32

- 6.8. This figure depicts the average normalized stretch on the Topology Zoo networks for a range of clustered disrupted links, **where *Feigenbaum* significantly surpasses the other two algorithms**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 50 individual trials carried out for each failure mark. 33

- 6.9. This figure depicts the precomputation time in seconds on the Topology Zoo networks for all algorithms, carried out across 50 individual trials. The width of the violin plots is proportional to the number of observations, **showing a skewed distribution of the violin plots, especially for *Feigenbaum* and *SquareOne***. 34

- 6.10. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the rings of cliques networks for a range of random disrupted links, **where *SquareOne* experiences major routing issues for the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. . . . 35
- 6.11. Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the rings of cliques networks, based on the subset of nodes that remain connected after 10 random disrupted links, across 50 individual trials. **Both *Feigenbaum* and *TwoResilient* show a higher dispersion around the threshold**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y 36
- 6.12. This figure depicts the average normalized stretch on the rings of cliques networks for a range of random disrupted links, **where *Feigenbaum* outperforms the other two algorithms by a large margin and *TwoResilient*, decreases in stretch as failures increase**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. 37
- 6.13. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the rings of cliques networks for a range of clustered disrupted links, **where *SquareOne* experiences major routing issues for the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. . . . 38

List of Figures

- 6.14. Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the rings of cliques networks, based on the subset of nodes that remain connected after 10 clustered disrupted links, across 50 individual trials. **Both, *Feigenbaum* and *TwoResilient* show a higher dispersion around the threshold.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y 39
- 6.15. This figure depicts the average normalized stretch on the rings of cliques networks for a range of clustered disrupted links, **where *Feigenbaum* significantly surpasses the other two algorithms and *TwoResilient* shows the largest stretch overall.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. 40
- 6.16. This figure depicts the precomputation time in seconds on the rings of cliques networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, **showing a skewed distribution of the violin plots, especially for *Feigenbaum* and *SquareOne*.** 40
- 6.17. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Erdős–Rényi networks for a range of random disrupted links, **where *TwoResilient* performs the best out of all three algorithms.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. . . . 41
- 6.18. This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, **where *SquareOne* is generating the longest stretch.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. 42
- 6.19. This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, only for runs where both *Feigenbaum* and *TwoResilient* have been successful and had a success rate of ≥ 0.5 . ***Feigenbaum* produces less stretch for the same source-destination pairs compared to *TwoResilient*.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.. . . . 43

- 6.20. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Erdős–Rényi networks for a range of clustered disrupted links, **where *Feigenbaum* and *TwoResilient* performs almost identically**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. . . . 44
- 6.21. This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of clustered disrupted links, **where *TwoResilient* displays the largest stretch out of all algorithms**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. 45
- 6.22. This figure depicts the precomputation time in seconds on the Erdős–Rényis networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, **where *Feigenbaum* experiences an unusually wide distribution**. . 45
- 6.23. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the 3-regular networks for a range of random disrupted links, **where *Feigenbaum* and *TwoResilient* experience routing issues from the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. 47
- 6.24. This figure depicts the average normalized stretch on the 3-regular networks for a range of random disrupted links, **where *TwoResilient* produces the highest stretch**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. . . 48
- 6.25. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the 3-regular networks for a range of clustered disrupted links, **where *Feigenbaum* performs slightly better than *TwoResilient***. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. 49

List of Figures

- 6.26. This figure depicts the average normalized stretch on the 3-regular networks for a range of clustered disrupted links, **where *TwoResilient* has the highest stretch out of all algorithms**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark. 50
- 6.27. This figure depicts the precomputation time in seconds on the 3-regular networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, **where *TwoResilient* has a wide distribution range**. 50
- 6.28. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Rocketfuel networks for a range of random disrupted links, **where *Feigenbaum* is slightly ahead of *TwoResilient***. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 10 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. 51
- 6.29. This figure depicts the average normalized stretch on the Rocketfuel networks for a range of random disrupted links, **where *Feigenbaum* displays the lowest and *SquareOne* the highest stretch**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark. 52
- 6.30. Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Rocketfuel networks for a range of clustered disrupted links, **where *TwoResilient* is the best performing algorithms**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 10 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation. 53
- 6.31. This figure depicts the average normalized stretch on the Rocketfuel networks for a range of clustered disrupted links, **where *SquareOne* has the highest stretch, due to long detours**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark. 54
- 6.32. This figure depicts the precomputation time in seconds on the Rocketfuel networks for all algorithms, carried out across 10 individual trials. The width of the violin plots is proportional to the number of observations, **where all algorithms have a wider distribution**. 54

A.1. Similar to [FKP ⁺ 21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the <i>Feigenbaum</i> algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. <i>TwoResilient</i> experiences a dispersion around <i>Feigenbaum</i>, however still lies in the vicinity of the line, whereas <i>SquareOne</i> lies clearly below.	69
A.2. Similar to [FKP ⁺ 21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the <i>Feigenbaum</i> algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 clustered disrupted links, with the analysis conducted across 50 individual trials. The routing success for <i>SquareOne</i>, in some cases, is better than <i>Feigenbaum</i>, for the majority however stays below.	70
A.3. Similar to [FKP ⁺ 21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the <i>SquareOne</i> algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. In almost all tests both <i>Feigenbaum</i> and <i>TwoResilient</i> perform better.	71
A.4. Similar to [FKP ⁺ 21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the <i>SquareOne</i> algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. In the majority of tests both <i>Feigenbaum</i> and <i>TwoResilient</i> perform better.	72
A.5. This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, only for runs where all algorithms have been successful and had a success rate of ≥ 0.5 . Although, <i>Feigenbaum</i> is still taking the shortest detours, <i>TwoResilient</i> produces less stretch, when looking at runs with a higher success rate. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark.	73

List of Algorithms

1.	<i>SquareOne</i> : Route package from s to t in graph G	11
2.	<i>Feigenbaum</i> : Construct 1-resilient forwarding pattern from an arbitrary node n in G to target node t in G	13
3.	<i>TwoResilient</i> : Construct a 2-resilient forwarding pattern for a (s-t)-pair in graph G	15

Listings

4.1. Main functions	16
4.2. Compute biconnected components	16
4.3. Determine connectivity	17
4.4. Edge-disjoint routing	17
4.5. Compute cuts	17
4.6. Compute connected components	18
4.7. Add dummy nodes for augmented component	18
4.8. Gadgets 2 and 6 computation	19
4.9. Routing logic	19

1. Introduction

In our modern digitally interconnected era, the robustness of communication networks against unexpected link failures is crucial, as data traffic increases constantly with the rise of new technologies such as generative artificial intelligence. As the network’s scale expands and traffic rises, so does the frequency of link failures [GJN11], calling for swift and reliable failover mechanisms to uphold service quality and ensure uninterrupted connectivity as even short network disruptions can cause a serious decline in service quality [AGM⁺10] [VHV12] [ZDM⁺12].

Traditional routing protocols such as OSPF [Moy98] and IS-IS [ISO02] often fall short in speed, propelling the adoption of local fast failover strategies within the data plane. These strategies enable immediate rerouting to alternative paths, ensuring network resilience without the latency of global recomputation of routing tables [FGP⁺12] [LPS⁺13].

In the data plane, rapid rerouting is pivotal for ensuring network reliability, markedly surpassing control plane speeds. Networks now commonly employ pre-set conditional failover paths, such as IP Fast Reroute [Jar13] [Kam18] [PSA05] or MPLS Fast Reroute [PSM⁺18], facilitating immediate traffic redirection upon detecting disruptions. This necessitates a complex setup where routing protocols are pre-established, demanding foresight to anticipate and counteract any subsequent failures downstream, thereby ensuring uninterrupted network resilience and robustness against a spectrum of potential failures.

Fast failover algorithms are widely adopted in contemporary communication networks as they offer a swift solution for redirecting traffic when links fail, hence their name. Operating within the data plane, it leverages local data for making quick decisions, bypassing the typical control plane’s latency due to path reconvergence [VCR15], link reversal [GB81], or alerts from a centralized controller [FGP⁺12], thus enhancing network responsiveness and reliability.

However, designing an effective fast reroute algorithm can be intricate, as the decision making such as to which port to forward data, is based on local information. Additionally, challenges arise such as sustaining network connectivity [CGM⁺16] [CNP⁺14] and the management of factors such as network stretch or congestion [FPST19]. The conception of effective failover routing strategies has been a key focus in recent research [CKR⁺21]. While employing randomization [CGM⁺16] [CSA⁺21], such as random walks, can address network failures if the graph remains connected, most standard network devices such as routers and switches lack the capability for such randomized routing efficiently. Further-

1. Introduction

more, strategies that involve replicating packets, such as flooding, are not favored due to the significant extra burden they introduce to the network infrastructure. Utilizing edge-disjoint spanning trees and paths is a prevalent method, enabling local rerouting to alternate pathways in the event of a failure, thus ensuring the packet’s progress to its destination continues uninterrupted. While spanning trees conserve routing table space, their utility is constrained by overall network connectivity, unlike edge-disjoint paths, which offer superior routes and enhanced failure resilience, such as the SquareOne algorithm presented by Foerster et al. [FPST19], ensuring robustness against $k - 1$ link failures in networks with k -connectivity.

Concerning resilience guarantee, Feigenbaum et al. [FGP⁺12] presents a directed acyclic graph (DAG) based method ensuring 1-resilient failover routing, though complete resilience against multiple link failures is unattainable in networks with as few as eight nodes. What’s more Chiesa et al. [CNM⁺17] shows that achieving 2-resilience is unfeasible without considering source data, despite standard protocols including such information. Existing methods mostly offer heuristic based resilience, necessitate changes to packet headers, or depend on high network connectivity [CGM⁺16] [CSA⁺19] [FHP⁺21] [YLS⁺14]. Thus, the effectiveness of 2-resilient or 3-resilient failover routing remains uncertain, especially in real world networks with varied connectivity levels, highlighting the need for robust solutions adaptable to general network topologies. In regards to 2-resilient fast failover routing, Dai et al. [DFS23] presents a novel routing scheme and shows that 2-resilience is always possible on every network topology and can be calculated in polynomial runtime and that $k = 3$ resilience or better is impossible in general.

1.1. Contributions

Driven by the growing demand for fast acting and reliable failover routing schemes, we present a comprehensive empirical analysis of the most recent advances in *state-of-the-art* fast failover algorithms, which particularly guarantee resilience against k failures such as discussed in [FPST19], [FGP⁺12] and [DFS23]

In our work, we implement the *TwoResilient* algorithm, for which so far, only an abstract mathematical description was available. We analyze the algorithmic details and design a new implementation that efficiently solves various subproblems in practice inherent to the global routing scheme discussed in Dai et al. [DFS23]. Moreover, we design and implement extensions to the *TwoResilient* algorithm to heuristically increase its routing performance and discuss details as well as challenges we have faced during this process.

Furthermore, we conduct a comprehensive evaluation of these three k -resilient fast rerouting algorithms and measure their performance in terms of overall routing success, stretch and precomputation time of the routing tables on both synthetic and real world networks with two link failure models, random and clustered, orienting on previous work conducted in [SFS21], [FPST19], [FKP⁺21], [FKP⁺19].

Our simulation results reveal the strengths and weaknesses of these algorithms applied on divergent networks, which provide beneficial insights to increase their effectiveness on specific network topologies. In our evaluation, the DAG based algorithm presented by Feigenbaum et al. [FGP⁺12] proves to achieve the overall best performance in routing success, stretch and precomputation efficiency. On par with the *Feigenbaum* algorithm in terms of routing success is also the *TwoResilient* algorithm introduced by Dai et al. [DFS23]. As we can see in Table 1.1, *TwoResilient* even seems to have a slight advantage in routing performance compared to *Feigenbaum* on the majority of network topologies for the first 30% of the total failure ranges. The *SquareOne* algorithm discussed by Foerster et al. [FPST19] shows the least effective metrics in all our tests.

To contribute to the research community and promote reproducibility, as well as to aid future studies, we are releasing all our code and experimental resources on our Github repository [Zab24].

Topology	<i>SquareOne</i>		<i>Feigenbaum</i>		<i>TwoResilient</i>	
	RANDOM	CLUSTER	RANDOM	CLUSTER	RANDOM	CLUSTER
Zoo	0.36%	0.40%	0.46%	0.50%	0.47%	0.51%
Ring of Cliques	0.70%	0.68%	0.96%	0.96%	0.93%	0.95%
Erdős–Rényi	0.80%	0.70%	0.97%	0.88%	0.98%	0.88%
Regular	0.51%	0.69%	0.79%	0.82%	0.81%	0.82%
Rocketfuel	0.75%	0.80%	0.89%	0.96%	0.90%	0.97%

Table 1.1.: Average success rate of algorithms for the first 30% of the total failure ranges across different topologies for RANDOM and CLUSTER link failure models described in Section 5.2. The highlighted table cells show the highest success rates on each topology and link failure model.

1.2. Organization

The structure of this paper is outlined as follows. In Chapter 2 we discuss preceding research conducted in the area of fast failover routing algorithms and related fields. In Chapter 3 we introduce general information on networks and fast failover algorithms relevant for understanding this work as well as the *SquareOne* and *Feigenbaum* algorithms studied in this evaluation. We provide a more in depth explanation of the *TwoResilient* algorithm in Chapter 4 before moving on to Chapter 5 where we explain our methodology, including an introduction into the used topologies as well as our experimental testing setup. Followed by Chapter 6, where we discuss the results of our empirical evaluation. After discussing the main findings of our work, we conclude in Chapter 7 and provide an outlook on future research.

2. Related Work

In this chapter, we introduce and study preceding research conducted in the area of fast failover algorithms and related topics. We start this chapter off by introducing research done in the area of fast failover algorithms in general in Section 2.1, highlighting their advances and limits. In Section 2.2 we take look into studies using arborescences for constructing routing tables and in Section 2.3 work is discussed, which reports about the challenges of routing algorithms when faced with multiple link failures. Lastly, we take a peak into ongoing research concerning load as well as stretch in Section 2.4.

2.1. Fast Failover Advances and Limits

Several studies have already performed an analysis of fast failover algorithms that do not pose the need for packet header rewriting. Concerning algorithms which do not pose the need for a packet headers modification, Feigenbaum et al. [FGP⁺12] highlight the impossibility of achieving perfect resilience in this context, where local rerouting cannot always ensure packet delivery even if the network remains intact following failures. Related findings by Borokhovich and Schmid [BS13] also highlight limitations in load balancing under similar scenarios. Subsequently, Foerster et al. [FHP⁺21] expanded these impossibility results, creating a connection between graph minors and network resilience, demonstrating that perfect resilience is unattainable in non planar graphs but feasible in outer planar graphs. What's more, Dai et al. [DFS23] shows that 2-resilience is possible by proposing a 2-resilient forwarding scheme, but proves that resilience against already three link failures does not exist by showing the impossibility of 3-resiliency.

In the domain of computer network resilience, particularly concerning ISP networks [MIB⁺08], [TLSS10], cloud provider WANs [GMK⁺16], and data centers [GJN11], [PJ13], link failures are not uncommon. Such failures have accelerated the development of efficient fast rerouting algorithms and hence have been intensively explored in the literature. The main goal of these studies is to devise strategies that can promptly respond to link failures, especially in the control plane [JRT20], [LKM⁺14], using various techniques such as fast reconvergence [BST03], link reversal [GB81], [CE95], or approaches rooted in software defined networks [VCR15], [MS15].

Also within the data plane, strategies been extensively studied are commonly employed as primary defense against link failures. Various relevant protocols have implemented these mechanisms, including IP [AZ08], [RTEC11], as well as MPLS [PSA05], [JKM⁺18], BGP [FMB⁺11], and in the realms of Software Defined Networks and modifiable data planes [ONF12], [CSA⁺19], among others.

2.2. Arborescence Routing Advances

The conventional method for destination oriented routing involves the use arborescences, [CNP⁺14], [CNM⁺17], [EGR16]. These structures offer robust worst case resilience based on the count of such trees implementable. Recent advancements have refined their integration into diverse network types through heuristic methods, such as incorporating local non spanning arborescences [FKP⁺21]. This work from Foerster et al. studies and compares the improved resilience of fast failover algorithms in heterogeneous networks with varying connectivity by modifying them with mechanisms based on spanning trees and arborescences, which are arc disjoint spanning trees. Yang et al.'s [YLS⁺14] work on the partial structural network model marks a departure from traditional spanning trees and acyclic graphs, demonstrating several practical advantages. However, their model encounters challenges in worst case failure scenarios, where a lone arc failure can lead their algorithm to break, even in networks with high remaining connectivity.

2.3. Routing Challenges with Multiple Failures

The inherent limitation of destination focused routing is, its longer path lengths compared to approaches considering both source and destination as reported in [FPST19]. In scenarios involving routing decisions based on both the source and destination, the CASA framework demonstrated that employing edge disjoint paths (EDPs) enhances the resilience in worst case scenarios while also optimizing the number of hops, albeit at the cost of expanded routing table requirements.

Our research closely aligns with the framework established by Foerster et al. [?] and extends their methodology for performing an analysis on fast failover algorithms. Further exploration of fast failover mechanisms reveals a considerable difference between single and multiple failure scenarios. While many existing robust routing solutions focus on single failures [ERC07], [NLY⁺07], they often fall short in more complex scenarios, such as failures of link clusters or node failures [SYR05], [GJN11] and [SIG⁺02].

Alternative methods for managing multiple link failures involve modifying packet headers [EGR16], [CKLS15], such as enabling packets to transmit failure data [LCR⁺07], [FPCS18], or duplicating packets [HCCR09]. These techniques, while effective, tend to either use up header capacity or add extra network load. Another strategy involves precalculating various flow paths, enabling ingress switches to adjust traffic distribution effectively and efficiently in response to numerous failures, all without necessitating extra computational resources [LKM⁺14]. A major part of research on fast failover routing algorithms with multiple link failures has been centered on maintaining network connectivity. However, modern communication networks need to fulfill availability and performance standards, from which connectivity is only one attribute out of many. The issue of evaluating the effectiveness of failover routing paths remains largely unresolved, as highlighted in the literature [LPS⁺13].

2.4. Evolving Fast Failover Solutions

Although some preliminary studies have explored the impact on failover paths in highly dense networks [PST17], the broader topic still requires further exploration. Existing arborescence based solutions [CNM⁺17], [CNP⁺14], [CGM⁺16], [CNM⁺16], without additional enhancements, often lead to extended failover routes. This issue is also prevalent in numerous other algorithms, such as those based on graph search techniques that depend on modifying packet headers [BSS14]. We give an overview of failover routing schemes in Table 2.1. To date, Foerster et al.’s work [FPST18] stands out as it develops arborescences with minimal stretch, though this is limited to specific network structures like torus, hypercube, grid graphs, and trees within complete bipartite graphs. In contrast, our approach extends to more general graph types and includes comprehensive practical assessments and analyses of complexity.

Resilience for static & dynamic routing tables	Packet header modification	K-edge connected	Per-source	Per-destination	Per-incoming port
($k-1$)-resilience [CGM ⁺ 16]		x		x	x
($k-1$)-resilience for $k \leq 5$ [CNM ⁺ 17]		x	x	x	x
($k-1$)-resilience for $k \leq 5$ [CNP ⁺ 14] [CNM ⁺ 17]	x	x		x	x
1-resilience [FGP ⁺ 12]				x	x
2-resilience [DFS23]			x	x	x
($k-1$)-resilience [FPST19]		x	x	x	x
($\lfloor k/2 \rfloor$)-resilience for $k \geq 6$ [CNM ⁺ 17]		x		x	x
arbitrary resilience as long as network stays connected [BSS14]	x			x	x

Table 2.1.: Summary of the resiliency of routing schemes. Highlighted schemes are discussed in this work.

3. Preliminaries

We now present some foundational concepts and essentials of network routing, where we briefly talk about network planes in Section 3.1. Afterwards, we move on to a more in detail explanation of fast failover algorithms in Section 3.2 and the two out of three algorithms we are discussing in this work, namely *SquareOne* in Section 3.2.2, *Feigenbaum* in Section 3.2.3. We discuss *TwoResilient* in the following Chapter 4.

3.1. Planes in Networks

In communication networks, the network layer of the OSI stack is subdivided in two sub-layers, namely the data plane (also commonly referred to as forwarding plane) and the control plane [Bon24]. There is also a third plane known as the management plane, which is concerned with network maintenance and administration.

3.1.1. Data Plane

As the name already may suggest, the data plane's main concern is the actual transmission of data across the network, which involves different operations such as the handling, forwarding and processing of incoming data packets as they are transferred from their source to their destination. To create processes which enable the exchange of data, the data plane contains all algorithms and protocols. The data plane is often implemented in hardware on high end routers, to enable a higher throughput and processing of packets and for a given network technology there is only one data plane.

3.1.2. Control Plane

Where the data plane is concerning with the actual transfer and right format of data across the network, the control plane layer is where the network's routing decisions are made. This plane uses routing tables and protocols to find the most viable paths for the packets from their sources to their destinations. To do so, it contains all necessary algorithms and protocols, which are installed on the routing devices in the network. Also, while there is only one data plane for a given network technology, as we mentioned above, there can be different control planes for the same network technology.

3.2. Fast Failover Algorithms

Fast failover algorithms in network routing are designed to ensure high availability and reliability within complex communication networks by rapidly addressing link failures,

3. Preliminaries

which is a critical aspect given the increasing scale and interconnectivity of modern telecommunication networks. Unlike conventional routing protocols, which often suffer from considerable delays during reconvergence following a link failure [CKR⁺21], fast failover mechanisms function at the data plane level making them more reactive and flexible in nature as their traditional counterparts. Fast failover algorithms leverage local, readily available information, enabling fast decision making [FGP⁺12] [LPS⁺13], thus bypassing the delays and complexities associated with control plane processes such as path reconvergence [VCR15], link reversal [GB81], or coordination with a centralized controller [FGP⁺12].

One of the fundamental challenges when designing fast failover algorithms lies in the inherent limitations of local solutions. These solutions must make forwarding decisions with limited knowledge of the network state due to local constraints. These constraints can make it difficult to maintain not only connectivity but also to optimize for other network performance metrics like path stretch, which is the increase in path length due to rerouting or congestion levels. Most recent fast failover approaches utilize precomputed, edge-disjoint spanning trees, connected components and paths [FKP⁺21]. In these algorithms packets are immediately redirected to an alternate precomputed path or tree, upon encountering and routing error. This redirection is based on locally stored information and occurs without the need for packet header modifications or dynamic changes in the routing tables, which ensures minimal delays.

Edge-disjoint paths offer a robust solution, providing direct routes between source and destination and enhancing resilience to failures. The diversity of paths ensures that even if one path becomes inoperable due to a failure, alternate paths remain viable for rerouting traffic. Meanwhile, spanning trees, though they occupy less space in routing tables, are limited by their reliance on the global connectivity of the network. They may not fully utilize the available path diversity in the network, potentially leading to suboptimal rerouting decisions.

In the context of fast failover, maintaining the balance between quick, local decision making and the need for comprehensive network information becomes a critical design consideration. The effectiveness of a fast failover algorithm is often measured by its ability to provide rapid recovery from failures, minimize the stretch of rerouted paths, and efficiently use the network’s resources without causing congestion. These algorithms are not only pivotal in maintaining the uninterrupted flow of data in the face of network disruptions but also play a crucial role in optimizing the overall performance and resilience of the network infrastructure.

In the following sections, we introduce three different fast failover algorithms, namely *SquareOne*, *Feigenbaum* and *TwoResilient*, which have all in common, that they provide guarantees in network routing resilience against link failures. In particular, *SquareOne* guarantees a resilience against $k-1$ failures in a k -connected network, *Feigenbaum* promises

to compute a 1-resilient routing scheme in general and lastly *TwoResilient* even a 2-resilient routing scheme if specific conditions are met. Other fast failover algorithms such as *CASA*, *Keep Forwarding* or *DAG-FRR* (Feature Fast Reroute) discussed in [FPST19] and [FKP⁺21] do not provide such guarantees, respectively are based on heuristic extensions. For an overview of related studies in the field of fast failover routing, please refer to Chapter 2.

3.2.1. Network Structure

We present a given network as a bidirected graph $G = (V, E)$ where each node in the set V represents a router in the network and each edge (u, v) in the set E represents a bidirected link between two routers u and v . Furthermore, multi-edges between two nodes u and v are distinguishable. We model failed edges, that are link which can not be used for packet transfer, in the graph G as $F \subseteq E$.

3.2.2. SquareOne

The *SquareOne* algorithm described in [FPST19] uses a simple and straight forward routing technique. It computes edge-disjoint paths in a given graph G to route a package from a source s to a target t and ensures robustness to $k - 1$ link failures in k -connected graphs. To be more precise, the algorithm sorts the computed routes based on their respective length and begins with the shortest route from s to t . If the algorithm encounters a failed link, the algorithm detours the taken path to the beginning and then switches to the next available path in the set of computed paths until the destination is reached or no more paths are available.

Algorithm 1: *SquareOne*: Route package from s to t in graph G

Input: A graph $G = (V, E)$ with a source-destination pair $(s, t) \in V$ and a set of failed links $F \subseteq E$

- 1 Let $P = \{p_1, p_2, \dots, p_n\}$ be a set where each element p_i represents an edge-disjoint path from s to t . The length of each path p_i is denoted as $\ell(p_i)$;
 - 2 $\forall p_i, p_j \in P$ with $i < j$, $\ell(p_i) \leq \ell(p_j)$;
 - 3 For each $p_i \in P$ traverse p_i from s to t . Upon encountering a link $l \in p_i$ such that $l \in F$, then detour p_i back to s and switch to p_{i+1} ;
 - 4 Repeat until target t is reached or no more paths are available in P ;
-

3. Preliminaries

3.2.3. Feigenbaum

The *Feigenbaum* [FGP⁺12] algorithm utilizes directed acyclic graphs (DAGs) for constructing routing schemes and guarantees 1-resilience provided the network stays connected. Due to its directed nature DAGs already induce simple forwarding rules, where a node s can forward a packet from its incoming edges to its outgoing edges. In Feigenbaum, each node's incoming edge in DAG D is directed to its initial outgoing edge in D . The order on which is determined which one of the outgoing edges is first can be arbitrary. Figure 3.1 illustrates a simple DAG where node 2 directs traffic from node 1 to node 3 if the incoming edge to 3 is active otherwise it will send the traffic to node 4. The algorithm uses a gradual approach where it identifies biconnected nodes to a target t which are not yet included in the partial routing pattern so far. In Figure 3.1 nodes 2, 3 and 4 are such nodes. By also adding the next routing rules to node 2, the node is added to the forwarding pattern, thus making the pattern more resilient:

1. If both outgoing edges are inactive (i.e. have failed) in DAG D , then the traffic reaching node 2 from node 1 is sent back to 1.
2. If node 1's direct edge to target node t is active, then all traffic reaching node 1 from 2 is sent along this edge.

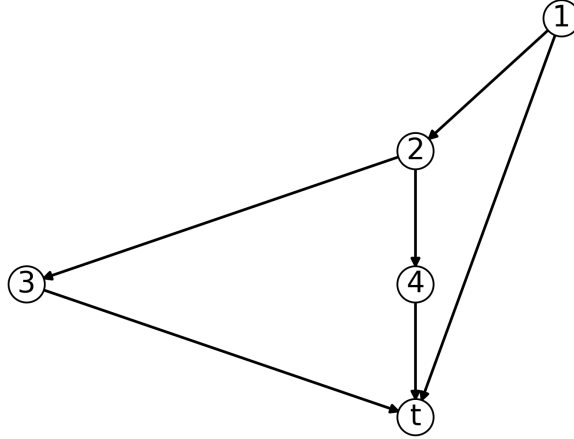


Figure 3.1.: Directed Acyclic Graph (DAG) with root node 1 and target node t

Algorithm 2: Feigenbaum: Construct 1-resilient forwarding pattern from an arbitrary node n in G to target node t in G

Input: A graph $G = (V, E)$ with a target node $t \in V$ and a set of failed links $T \subseteq E$

Output: A 1-resilient forwarding pattern f^t in G

- 1 **Initialization:** $\forall e = (i, j)$ in E and $\forall T \subseteq E$, set $f_j^t(e, T) := \emptyset$;
 - 2 **Construct DAG:** Construct a DAG $D = (V, E_D)$ that is rooted in t and covers all edges in E . Nodes V have the following partial order: $\forall (i, j) \in V, i <_D j$, only if there is a route from j to i in D ;
 - 3 **Add DAG-based Forwarding Rules:** $\forall i \in V$, define E_D^i as i 's outgoing edges in D . For each incoming edge e and subset T where $T \cap E_D^i \neq \emptyset$, set $f_i(e, T)$ to the highest edge in E_D^i that is also in T ;
 - 4 **Additional forwarding rules:**
 - while** $\exists q$, a biconnected to t in G but not in $f^t = (f_1^t, \dots, f_n^t)$ **do**
 - 5 Choose i as a minimal node under $<_D$ that is biconnected to t in G but not in f^t ;
 - 6 Choose j as a minimal node under $<_D$ such that $i <_D j$ and $\exists x \in V$ such that (j, x) in D with $i \not\leq_D x$;
 - 7 Choose a simple route $R = (j = v_1, \dots, v_k = i)$ from j to i in D ;
 - 8 Set $c := k - 1$;
 - 9 **while** $(c > 1)$ and $(f_{v_c}^t(v_{c+1}, v_c) = \emptyset)$ **do**
 - 10 $f_{v_c}^t(v_{c+1}, v_c) := (v_c, v_{c-1})$;
 - 11 $c := c - 1$;
 - 12 **end**
 - 13 **if** $c = 1$ **then**
 - 14 $f_j^t(v_2, v_1) := (j, x)$;
 - 15 **end**
 - 16 **end**
-

4. The TwoResilient Algorithm

In this chapter, we conduct an in-depth examination of the novel *TwoResilient* algorithm. We describe the 2-resilient routing scheme in Section 4.1 and follow up on the implementation details and first extensions in Section 4.2.

4.1. 2-Resilient Routing Scheme

Opposed to the *Feigenbaum* algorithm we have discussed in Section 3.2.3, the *TwoResilient* algorithm, introduced by Dai et al. in [DFS23], guarantees 2-resilience in general network typologies and also demonstrates why 3-resilience is impossible in already simple networks. The algorithm computes a 2-resilient routing scheme by performing a subdivision and kernelization of the initial network into smaller and more sparse subgraphs, with the premise that each of these subgraphs has to be exactly 2 edge-connected and at least 2 node-connected between its respective source and destination pair. These subgraphs can be further broken down into even smaller components with specific characteristics that maintain the same level of connectivity between source and target. By integrating these components into a single kernelized subgraph of the original subgraph the algorithm preserves the required connectivity level in the face of two link failures between source and destination. To be able to evaluate *TwoResilient* together with *SquareOne* and *Feigenbaum*, we implemented the algorithm in the course of this work and will discuss the implementation details in the following section.

Algorithm 3: *TwoResilient*: Construct a 2-resilient forwarding pattern for a (s-t)-pair in graph G

Input: A graph $G = (V, E)$ with a source node s and a target node $t \in V$

Output: A 2-resilient forwarding pattern $\Pi^{(s,t)}$ for G

- 1 **Perform reductions:** Reductions on graph G stated in Claims 1 and 2 can be applied (see §4.2 in [DFS23]);
 - 2 **Compute 2-(s,t)-cuts:** Let $E^c := \{\{e, e'\} : \{e, e'\} \text{ is a } s - t \text{ cut in } G\}$;
 - 3 **Compute Connected Components:** Let $C = \{C_l : |V(C_l) \cap V(P_1 \cup P_2)| \geq 2\}$ (see Def. 4.3 §4 in [DFS23]);
 - 4 **Determine Types of Connected Components:** $\forall C_l \in C$ determine *Type* 1 to 6;
 - 5 **Determine Ports of Connected Component:** $\forall C_l \in C$ and $\forall$ elementary paths $P \in \{P_1, P_2\}$ the *left* - P port respectively the *right* - P port is defined as the first traversed node in C_l when traversing P from s to t or vice versa;
 - 6 **Compute Gadgets of Connected Components:** $\forall C_l \text{ in } C$ according to its type defined (see §5 in [DFS23]);
 - 7 **Compute Kernel Graph:** Combine all gadgets to get kernel graph $\mathcal{G} \subseteq G$;
 - 8 **Determine Forwarding Pattern:** Based on kernel graph $\mathcal{G}$ determine a 2-resilient forwarding pattern $\Pi^{(s,t)}$ to route on (see Def. 6.2 §6 in [DFS23]);
-

4.2. Implementation Details and Extensions

In the following section, we will discuss our implementation approach of the 2-resilient routing scheme. Please refer to our Github repository [Zab24] for the full implementation. The programming language of our choice was Python 3.9.0 [Pyt24] [VRD09] as it elevates the abstraction level and allows us to purely concentrate on the logic of the algorithm. Moreover, using Python allows us to utilize NetworkX [dev24b] [HSSC08], a powerful package for network analysis. The algorithm is divided into two steps 4.1, namely the preprocessing of the provided graph G and the routing from source to destination on the preprocessed network $\mathcal{G}$.

```
1 def PreComputeTwoResilient(graph, source, target)
2 def RouteTwoResilient(source, target, failed_links, kernel_graphs)
```

Listing 4.1: Main functions

1. We start off by computing all biconnected components in the graph, that are maximal subgraphs such that the removal of a node will not disconnect the graph. We then calculate the shortest path between the source and destination and filter out all biconnected components, which do not lie on the path. NetworkX proves already helpful as it provides all the functions needed as you can see in Listing 4.2 above.

```
1 biconnected_components = list(nx.biconnected_components(graph))
2 try:
3     shortest_path = list(nx.shortest_path(graph, source, target))
4 )
5 except nx.NetworkXNoPath as e:
6     print(e)
7     return -1
8
9 filtered_biconnected_components = list()
10 for component in biconnected_components:
11     nodes_on_path = [node for node in shortest_path if node in
12                     component]
13     if len(nodes_on_path) >= 2:
14         filtered_biconnected_components.append([component, [
15             nodes_on_path[0], nodes_on_path[-1]]])
16 # Implementation logic continues...
```

Listing 4.2: Compute biconnected components

2. To apply the algorithm and in the following compute 2-(s, t)-cuts 4.3, a subgraphs needs to have exactly 2 edge-disjoint paths and at least 2 node-disjoint paths from source to destination, respectively subsource and subdestination, named P_1 and P_2 . The subsource and subdestination for each subgraph are the nodes that the shortest path crosses through. To determine the number of edge-disjoint and node-disjoint paths NetworkX provides a function. If a subgraph is not 2 edge-connected or at least 2 node-connected, we skip the preprocessing, as we can't apply further steps.

4.2. Implementation Details and Extensions

```
1     if nx.edge_connectivity(subgraph, sub_source, sub_target) == 2
2         and nx.node_connectivity(subgraph, sub_source, sub_target)
3         >= 2:
4             cuts = get_two_s_t_cuts(subgraph, sub_source, sub_target
5             )
6             connected_components = get_connected_components(subgraph
, cuts)
# Implementation logic continues...
```

Listing 4.3: Determine connectivity

In the case we cannot continue with the 2-resilient routing scheme, we still can try to route the package, by finding all edge-disjoint paths in the subgraph and sequentially try them out. If a link failure occurs along the path, we need to route back to the source node and try another path in the list. This method can have a substantially negative impact on the stretch, as we always need to route back to the start, we have enhanced the algorithm to start with the shortest path first and in the case of failure taking the next shortest path in the list such as described in Listing 4.4. To even further extend the algorithm and possibly shorten the produced stretch, spanning trees connecting the paths can be utilized, so that in case of failure, the algorithm could switch between the paths without routing back to the source node.

```
1     paths = sorted(nx.edge_disjoint_paths(graph, sub_source,
2     sub_target), key=len)
3     # Try all shortest paths between s & t as long as
4     routing is unsuccessful
5     for path in paths:
6         success = simple_routing(path, failed_links)
7         if success:
8             break
9     # Implementation logic continues...
```

Listing 4.4: Edge-disjoint routing

3. The next step involves computing 2-(s, t)-cuts 4.5 and extracting all connected components 4.6 in the graph as described in 3.

```
1     def get_two_s_t_cuts(graph, source, target):
2         node_disjoint_paths = get_node_disjoint_paths(graph, source,
3         target)
4         p1 = get_edge_path(node_disjoint_paths[0])
5         p2 = get_edge_path(node_disjoint_paths[1])
6         two_s_t_cuts = []
7         for link_p1 in p1:
8             graph.remove_edges_from([link_p1])
9             for link_p2 in p2:
10                 graph.remove_edges_from([link_p2])
11                 if not nx.has_path(graph, source, target):
12                     two_s_t_cuts.append([link_p1, link_p2])
```

4. The TwoResilient Algorithm

```

12         graph.add_edges_from([link_p2])
13         graph.add_edges_from([link_p1])
14         return two_s_t_cuts
15

```

Listing 4.5: Compute cuts

```

1     def get_connected_components(graph, two_s_t_cuts):
2         cuts = set()
3         [(cuts.add(x[0]), cuts.add(x[1])) for x in two_s_t_cuts]
4         [graph.remove_edge(edge[0], edge[1]) for edge in cuts]
5         connected_components = list(nx.biconnected_components(graph)
6         )
7         connected_components = join_components(graph,
8         connected_components)
9         connected_components = [graph.subgraph(component) for
10        component in connected_components]
11        return connected_components

```

Listing 4.6: Compute connected components

4. After determining the types of connected components and their key nodes we can compute the gadgets of connected components. Gadgets of Types 1, 3, 4 and 5 can be computed right away, but, the computation of gadgets of Type-2 and Type-6 involves constructing three variants of so called augmented components 4.7 and two of these augmented components require parallel edges or paths between two nodes. NetworkX supports Multigraphs, however doesn't support flow functions, which we need to further processing. We overcome this obstacle by introducing dummy nodes, between parallel edges.

```

1     def compute_augmented_component(component, source, target):
2         augmented_component = nx.Graph()
3         augmented_component.add_edges_from(component.edges)
4         dummy_nodes = ["dummy-1", "dummy-2", "dummy-3", "dummy-4"]
5         # Augmented component logic continues...
6

```

Listing 4.7: Add dummy nodes for augmented component

5. To extract gadgets of Type-2 or Type-6 out of the augmented component we need to apply a *maxflow* function such as the *Ford-Fulkerson* method [CLRS09]. NetworkX provides the *Edmonds-Karp* algorithm [dev24a], which is an implementation of the *Ford-Fulkerson* method for computing the maximum flow in a network 4.8.
6. The last step of the preprocessing is to put the gadgets and 2-(s, t)-cuts together to create a (reduced) kernel graph $\mathcal{G}$. For joining graphs together NetworkX provides a useful function. Furthermore, $\mathcal{G}$ should be planar and we can check the planarity of the kernel graph with NetworkX, which returns a *PlanarEmbedding* class [dev24c]. This class has amongst other useful functions a method that traverses the face of the planar graph.

```

1     nx.edge_disjoint_paths(augmented_component, source, target,
2     flow_func=edmonds_karp), key=len)

```

Listing 4.8: Gadgets 2 and 6 computation

7. We can now proceed routing on $\mathcal{G}$ 4.9. Since the paths $P1$ and $P2$ may not be in $\mathcal{G}$ anymore, we introduce the primary path $Q1$ and secondary path $Q2$. We start the routing process from the source to the destination along the primary path $Q1$, which is also simultaneously the outer face of the planar graph. If a failed link is encountered along the path, we calculate a failover path by traversing the outer face from the in-port to the out-port of the failed link and resume routing on $Q1$ until the destination is reached. The failover routing is done recursively with reaching the out-port as termination condition.

```

1     def two_resilient_routing(planar_kernel_graph, source, target,
2     failed_links):
3         q1 = get_q1(planar_kernel_graph, source, target)
4         for link in q1:
5             if link in failed_links:
6                 face_hops = planar_kernel_graph.traverse_face(link
7                 [0], link[1])
8                 result = recursive_routing(planar_kernel_graph,
9                 face_hops, failed_links, target,
10                 link[1], link[1])
11             # Routing logic continues...
12
13     def recursive_routing(planar_kernel_graph, face_hops,
14     failed_links, target):
15         for face_link in face_hops:
16             if face_link in failed_links:
17                 face_hops = planar_kernel_graph.traverse_face(
18                 face_link[0], face_link[1])
19                 result = recursive_routing(planar_kernel_graph,
20                 edge_face_hops_new, failed_links)
21             # Recurrasive routing logic continues...

```

Listing 4.9: Routing logic

5. Methodology

So far we have introduced and described the algorithms we examine in our study. In this Chapter we introduce the network topologies in Section 5.1 as well as our experimental setup in Section 5.2, which we use to perform our evaluation.

5.1. Network Topologies

Network Topologies build the foundation of our research and therefore we have decided to perform our experiments on a vast number of different heterogeneous as well as homogeneous topologies, which vary in attributes such as size and connectivity.

- **Topology Zoo:** The Internet Topology Zoo [KNF⁺11] is an ongoing project for collecting network topologies from around the world. To conduct our research we have used all of currently 260 available networks in this collection.
- **Ring of Cliques:** A ring of cliques is a network that consists of densely connected cliques which are connected by bridges, that are links that connect adjacent cliques. Similar to [FKP⁺21], we examine a network structure composed of L highly interlinked cliques., where each clique has a internal link connectivity of k_{clique} . Neighboring cliques are linked through m_{bridge} edges connected to randomly selected nodes in the adjacent cliques, creating a network where the total connectivity is $k_{graph} \geq \min(k_{clique}, 2m_{bridge})$. Furthermore, we assume that $k_{clique} > k_{graph}$, meaning the internal connectivity within the cliques surpasses that of the overall network.
- **Erdős-Rényi Networks:** The prior two collections provide many different networks concerning their connectivity, number of nodes and edges. However, to get a better understanding of how the individual algorithms perform under more uniform conditions we also make use of our own randomly generated graphs. For graph generation we use the Erdős-Renyi model [ER60], which provides two closely related models for generating random graphs:
 - The $G_{n,m}$ model chooses a graph uniformly from a collection of graphs which have n nodes and m edges.
 - The $G_{n,p}$ model constructs a graph by connecting n nodes randomly and each edge in the graph is included with the probability p on a per edge individual basis.

5. Methodology

For our experiments we decided to proceed with the probabilistic method of network generation, due to its characteristics concerning degree distribution and edge formation. Whereas the $G_{n,m}$ method does not control the degree distribution, the $G_{n,p}$ method often results in a binomial distribution.

$$P(\deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

or a Poisson distribution for a large number of nodes n and a small probability p .

$$P(\deg(v) = k) \rightarrow \frac{(np)^k e^{-np}}{k!}$$

Furthermore, for our analysis we want to ensure that our network creation will produce connected graphs, which can only be disconnected by link failures.

In their work [ER60] Erdős and Renyi describe how graph generation acts for various probabilities p , namely if the condition

$$p > \frac{(1+\epsilon) \ln n}{n}$$

is met then a graph $G_{n,p}$ will be most certainly connected. Thus $\frac{\ln n}{n}$ acts as a threshold for the graph being connected.

- **Regular graphs:** The key feature of regular graphs is their homogeneity as each vertex has the same number of neighbors, meaning each vertex has the exactly same degree. This common degree of all vertices is often referred to as k and leads to some interesting properties such as symmetry.

For our analysis, we have used regular graphs with $k = 3$, that are graphs with vertices of degree three, also referred to as cubic graphs. Cubic graphs are of particular interest as they offer enough connectivity to be robust against multiple link failures, however still maintain structural simplicity.

- **Rocketfuel:** These networks were created by using an ISP topology mapping engine by researchers at the University of Washington [oW24]. The project involves trace routes sourced from over 800 vantage points hosted by approximately 300 trace route web servers. The project claims to have the most detailed ISP mappings besides the mappings of the ISP providers themselves.

These datasets have already been utilized by several studies. Chiesa et al. [CGM⁺16] and Foerster et al. [FPST19] evaluate fast failover algorithms on datasets of regular networks and respectively on ring of cliques networks. Schweiger et al. performs an analysis of routing algorithms on the Zoo and Erdős-Rényi datasets and an evaluation from Foerster et al. [FKP⁺19] utilizes the Rocketfuel dataset.

5.2. Experimental Setup

Before moving on to the evaluation and discussion of our experimental results concerning the three fast failover algorithms *SquareOne*, *Feigenbaum* and *TwoResilient*, we first want to describe our testing setup.

We conducted all of our experiments for an algorithm on a given graph $G = (V, E)$ based on these steps:

1. **Link failures:** In [FPST19] and [FKP⁺21] Foerster et al. as well as in Schweiger et al. [SFS21] describe two link failure models, namely RANDOM and CLUSTER. These two models profoundly differentiate in the way how failed links are calculated. We also define a parameter f_{num} to control the number of failed links.

- **RANDOM:** As the name may already suggest, in this method we uniformly choose f_{num} failed links $f \in F \subseteq E$ at random from all edges across the given graph G . All links in the network are exposed to an equal failure rate, which helps us to execute these algorithms under more non specific circumstances, where links aren't targeted by an attack or affected by a breakdown of central routers or router clusters.
- **CLUSTER:** Speaking of clusters, in this method we try to simulate the state of the network, as it would be after an attack. Meaning it tries to disengage well connected areas and therefore making the network more vulnerable to routing failures. In the cluster link failure model, failed links are selected through a two-step process:

First, compute the clustering coefficient for every node in the network. Each directed edge (arc) having a connection to a node with a clustering coefficient > 0 is included to the candidate set F_{cand} of potential disabled directed edges, except the candidate set already contains the adverse edge. Next, given that the amount of disrupted bidirectional links is constrained by the parameter f_{num} , two scenarios are taken into account for further selection. If the set of candidate directed edges F_{cand} has more elements than the parameter f_{num} allows, f_{num} directed edges are randomly selected from F_{cand} , and the corresponding bidirectional links are disabled. If F_{cand} contains f_{num} elements or fewer, all bidirectional links corresponding to the directed edges F_{cand} in are disabled.

2. **Target:** For each network, we uniformly select a target (destination) node t at random from all nodes in the set V of the given graph $G = (V, E)$.
3. **Precomputation:** For each algorithm the precomputation is performed to construct their routing patterns or routing tables and the time for their computation is measured separately.
4. **Source nodes:** Next we choose s_{num} source nodes $s_i \in V$ where $\forall s_i \neq t$.

5. Methodology

5. **Routing:** Similar to the precomputation step we now perform the routing of each algorithm separately based on their routing scheme for the respective graph and source s and target t . Note that after removing f_{num} failed links from the graph, the source and target must not be necessarily connected anymore.
6. **Success rate:** The performance of the evaluated algorithms was quantified by calculating their pathfinding effectiveness, defined as the proportion of the absolute amount of routing attempts that successfully reached their destination to the overall number of routing attempts initiated. For an ideal resilient algorithm the success rate is determined by the formula $p = \frac{(n_{cc}-1)}{(n-1)}$, where n_{cc} represents the count of nodes within the destination's connected component post failure occurrences.
7. **Repetitions:** Lastly, the parameter r_{num} denotes the number of independent repetitions we want to execute on the given graph to provide better statistical accuracy and increase the confidence and robustness of our results by observing variability and reducing the impact of outliers that may cause skews in our data.

6. Experimental Evaluation

We have conducted an extensive analysis on the three failover algorithms *SquareOne*, *Feigenbaum* and *TwoResilient* introduced at the beginning of our work in Chapter 3 and Chapter 4. In the following chapter, we briefly introduce our hardware and software specifications in Section 6.1 before moving to the detailed evaluation of the three algorithms on each network topology, which we have discussed in Chapter 5. We start off by evaluating the algorithms on the Zoo topology in Section 6.2. Next, we move on to the Ring of Cliques topology in Section 6.3 and Erdős–Rényi topology in Section 6.4. We then proceed our evaluation of the algorithms on Regular graphs in Section 6.5 and the Rocketfuel dataset in Section 6.6. We conclude this chapter with an discussion and summary of our findings in Section 6.7.

6.1. Hardware and Software Specification

The following experiments were conducted on a Macbook Air 2015 with the following specifications:

- **CPU:** 1.6 GHz Dual-Core Intel Core i5
- **RAM:** 8GB 1600 MHz DDR3
- **OS:** macOS Monterey Version 12.6.3

Python 3.9.0 [Pyt24] [VRD09] was used as the programming language of choice with the following packages/dependencies:

- Matplotlib v3.6.2 [Hun07]
- NetworkX v3.0 [dev24b] [HSSC08]
- Numpy v1.24.2 [HMvdW⁺20]
- Pandas v1.5.3 [pdt20]
- Pip v21.3.1 [pip]
- Scipy v1.11.3 [VGO⁺20]
- Seaborn v0.13.2 [Was21]

6.2. Experimental Evaluation in the Topology Zoo

Currently there are 260 graphs included in the Topology Zoo collection, which greatly differ in node size and connectivity and therefore pose an ideal study ground for analyzing the performance of the algorithms under real world circumstances. These highly heterogeneous networks vary between 20 to 200 nodes and 40 to 500 links.

6.2.1. Randomized Failures

Figure 6.1 depicts the routing success of *Feigenbaum*, *TwoResilient* and *SquareOne* based on our randomized link failure model from 10 up to 70 failed links. At first glance we can already observe that both *Feigenbaum* and *TwoResilient* perform almost identical in this scenario. When encountering up to 10 link failures both algorithms achieve an almost perfect success rate p denoted by the threshold. On the contrary, *SquareOne* seems to already exhibit routing problems when exposed to 10 link failures in the network. From 10 to 30 link failures the distance from the threshold increases for all three algorithms and decreases constantly the closer we get to the 70 failures mark. At the mark of 50 failures, *SquareOne* seems to catch up to the other two algorithms. This is most likely due to the fact that at this mark most of the smaller networks are not connected anymore and only a handful of larger networks remain which still experience a higher connectivity level even after 50 to 70 link failures.

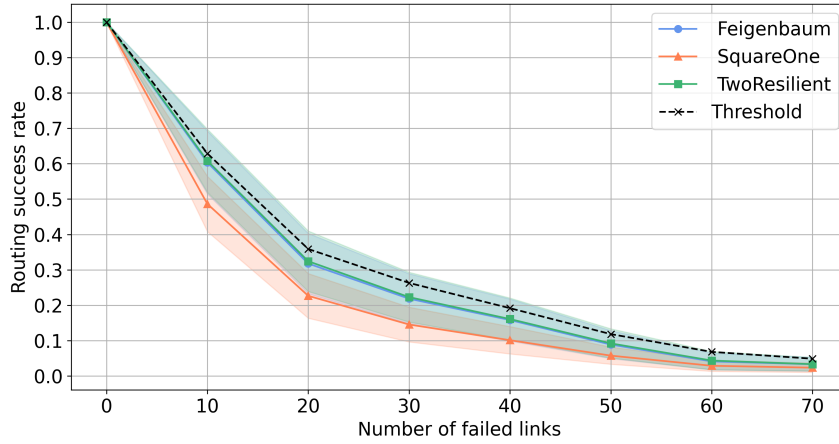


Figure 6.1.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Topology Zoo networks for a range of random disrupted links, **where *Feigenbaum* and *TwoResilient* algorithms show nearly identical performance.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 50 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6.2. Experimental Evaluation in the Topology Zoo

We delve deeper into the performance of the failover routing algorithms by analyzing the pathfinding effectiveness relative to the proportion of nodes that remain connected within the network components containing the destination after a certain amount of failures, marked by the threshold. By examining Figure 6.2, which depicts the success rate after 10 link failures, we see that the success for both *Feigenbaum* and *TwoResilient* is again similar and also lies near the perfect success rate p , whereas *SquareOne* experiences difficulties in routing in the still fairly connected component as we can see on between the 0.5 and 1.0 connected nodes range. Some markings in Figure 6.2 are depicted above the threshold. This phenomenon is due to limited amount of repetitions we performed for generating each dataset. In the case of Topology Zoo we performed 50 repetitions per graph for each algorithm. Increasing the amount of repetitions would align the markings on the threshold.

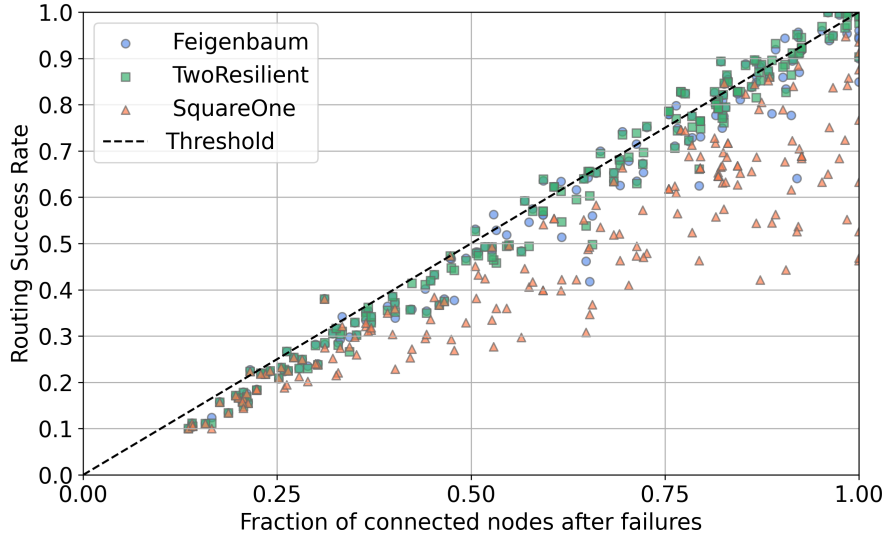


Figure 6.2.: Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the Topology Zoo networks, based on the subset of nodes that remain connected after 10 random disrupted links, across 50 individual trials. **Both *Feigenbaum* and *TwoResilient* algorithms achieve success rates close to the ideal value, denoted as p .** The threshold signifies the average fraction of nodes that remain connected to the destination sub-graph after failures, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y .

6. Experimental Evaluation

Observing Figure 6.3, we can study the performance of the algorithms relative to each other. In this case we are interested in the performance of *Feigenbaum* and *SquareOne* relative to *TwoResilient*. This allows for an even more detailed comparison between these algorithms. The routing success of each algorithm for every graph is normalized against the performance of *TwoResilient*, depicted as the threshold. Markings above the relative line translate to better performance and marking below translate to a worse performance than *TwoResilient*. For randomized link failures in the majority of test cases *Feigenbaum* performs similar to *TwoResilient*, as most of the blue dots are near or on the relative line, with some few cases where *Feigenbaum* performed better. *TwoResilient* outperformed *SquareOne*, since almost all test cases experience a lower success rate.

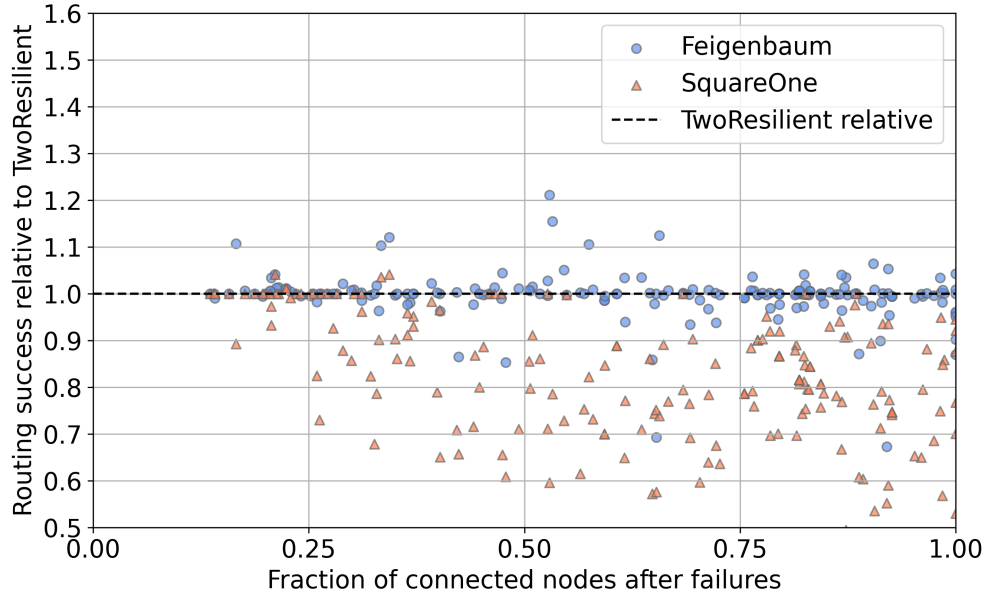


Figure 6.3.: Similar to [FKP⁺21], this chart illustrates the comparison of the relative average pathfinding effectiveness of all algorithms against the *TwoResilient* algorithm on the Topology Zoo networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. **For randomized link failures, in the majority of test cases, *Feigenbaum*'s performance lies near to *TwoResilient*'s performance.**

Another key indicator for the performance of a routing algorithm is stretch. Stretch denotes the difference of the amount of hops the routing algorithm needed to route a packet successfully from source to destination to the amount of hops of the shortest path between this pair of nodes. Since the stretch can highly vary in real world networks, we

6.2. Experimental Evaluation in the Topology Zoo

take the relative stretch to the shortest path. Figure 6.4 shows a significant difference in stretch for the three algorithms. In these metrics, *Feigenbaum* significantly surpasses the other two algorithms, having constantly a stretch near 1.0, meaning the taken routing path is almost optimal every time. *SquareOne* experiences a higher stretch at 10 failures than *Feigenbaum*. However, as the number of failed links increases the stretch decreases, coming nearer towards the level of *Feigenbaum*. The by far largest stretch is produced by *TwoResilient*. Interestingly, whereas *Feigenbaum* remains relatively constant throughout all failures and *SquareOne* experiences a downward trend when faced with an increased number of failures, *TwoResilient*'s stretch increases between 20 and 50 link failures.

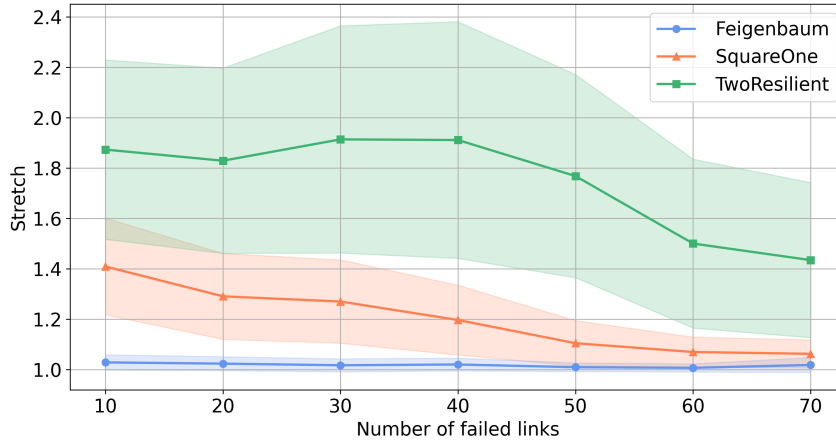


Figure 6.4.: This figure depicts the average normalized stretch on the Topology Zoo networks for a range of random disrupted links, **where *Feigenbaum* outperforms the other two algorithms by a large margin**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 50 individual trials carried out for each failure mark.

6.2.2. Clustered Failures

Random link failures occur in networks from time to time, but seldom in such a quantity. We now examine how the algorithms perform if a certain area of the network experiences multiple link failures, as it would be the case in a scenario, where attackers try to disrupt the network.

In Figure 6.5 depicting the routing effectiveness of each algorithm in regards to the magnitude of clustered link failures, we observe a similar picture as before in Figure 6.1, however with some noticeable differences. Until 30 failed links *Feigenbaum* and *TwoResilient* perform identical in terms of their success rate, but from 30 to 50 failed links *TwoResilient* faces a steady drop in success rate reaching the tipping point at 50 failures, where it continues to perform identical to *SquareOne*.

6. Experimental Evaluation

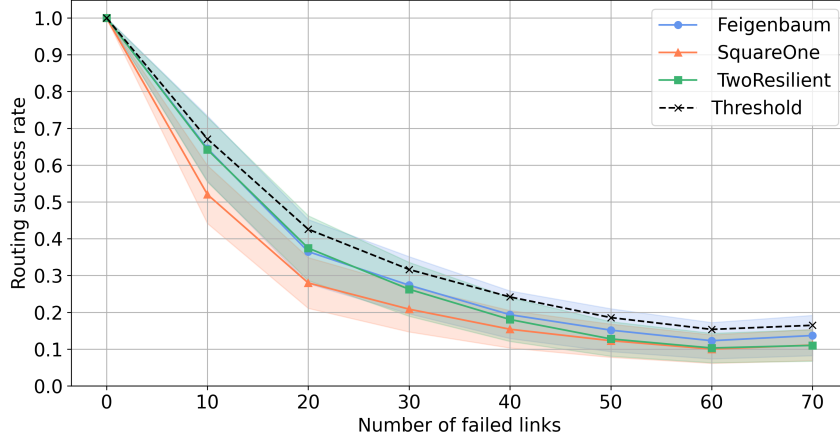


Figure 6.5.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Topology Zoo networks for a range of clustered disrupted links, **where *Feigenbaum* and *TwoResilient* algorithms show nearly identical performance.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 50 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

In the case of the success rate with respect to the percentage of connected nodes, we can detect nuances that have changed compared to the results presented in Figure 6.6 with 10 randomized link failures. The performance of all three algorithms seems to have decreased when confronted with 10 clustered link failures as *Feigenbaum* and *TwoResilient* are still relatively near the threshold, though this time more broadly scattered. The same goes for *SquareOne*, experiencing a new low in success rate of approximately 35% between the 75% and 100% node connectivity mark.

Comparing the relative success of *Feigenbaum* and *SquareOne* to *TwoResilient* shows us also a wider dispersion. *Feigenbaum*, previously keeping near *TwoResilient*, now spreads more apart in both directions when studying the section between 75% and 100% node connectivity in Figure 6.7. In regions below $y = 0.75$ *Feigenbaum*'s relative success rate tends to be in line with *TwoResilient*. Interestingly, *SquareOne*, which still widely underperforms compared to the other two algorithms, now shows in some cases a better performance than *TwoResilient* and also *Feigenbaum*, particularly between 25% and 50% node connectivity.

Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *TwoResilient* algorithm on the Zoo topology

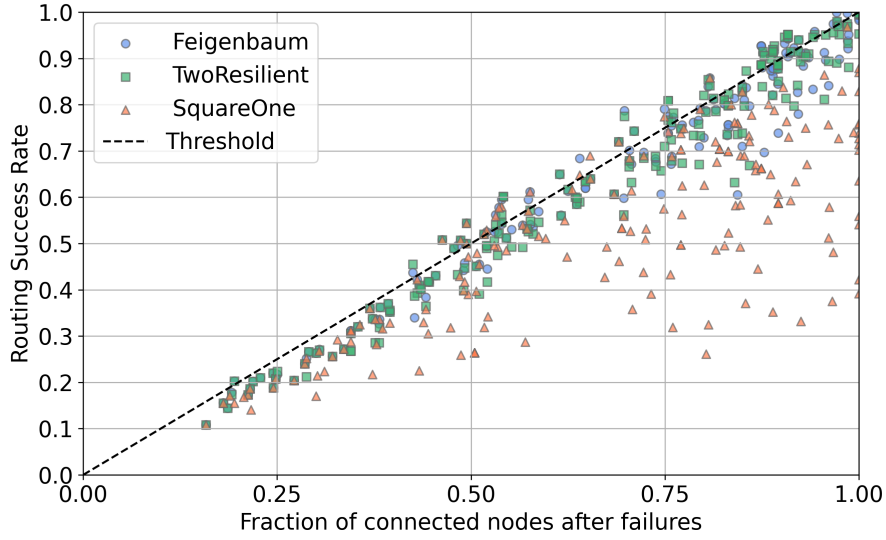


Figure 6.6.: Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the Zoo topology networks, based on the subset of nodes that remain connected after 10 clustered disrupted links, across 50 individual trials. **Both *Feigenbaum* and *TwoResilient* algorithms achieve success rates close to the ideal value, denoted as p .** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y .

dataset. This comparison is based on the subset of nodes that remain connected to the destination after 10 random link failures, with the analysis conducted over 50 individual trials. **For clustered link failures, in the majority of test cases, *Feigenbaum's* performance lies near to *TwoResilient's* performance.**

6. Experimental Evaluation

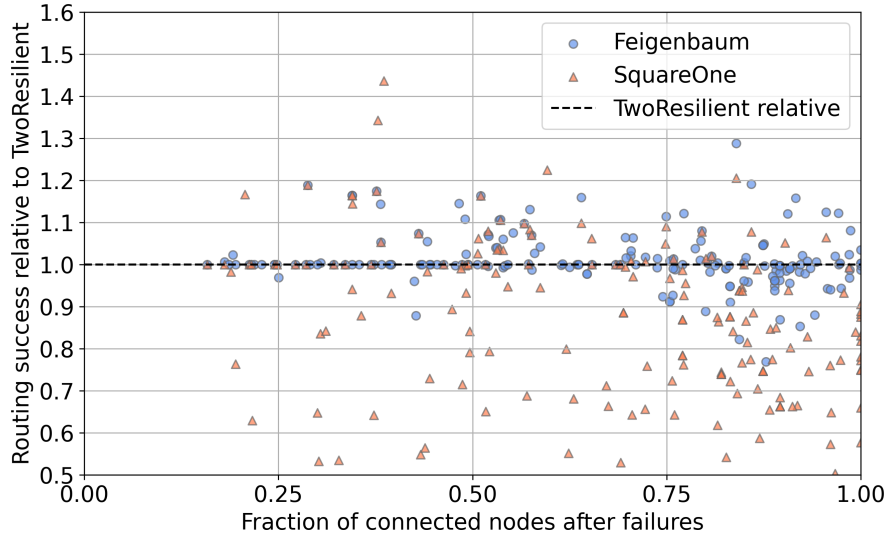


Figure 6.7.: Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *TwoResilient* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. **For clustered link failures, in the majority of test cases, *Feigenbaum*'s performance lies near to *TwoResilient*'s performance.**

As far as stretch is concerned, *Feigenbaum* still outperforms both *TwoResilient* and *SquareOne* by having a constant stretch of almost 1.0 throughout all link failure sizes, as shown in Figure 6.8. *SquareOne* now performs slightly worse and *TwoResilient* although still producing the widest stretch, is slightly better. Also this time, there is no increase in stretch with a growing number of failures as it has been the case in Figure 6.4.

6.2.3. Precomputation Performance

We lastly study the precomputation time of the algorithms for the routing tables. The most striking characteristic in Figure 6.9 is the skewed distribution of the violin plots, especially for *Feigenbaum* and *SquareOne*. *TwoResilient*'s precomputation time is the highest of all three algorithms, with the average time taken for preprocessing the network being approximately 0.07 seconds. We also see that *TwoResilient* has the highest distribution compared to *Feigenbaum* or *SquareOne*. Latter almost has no distribution width as the 25% and the 75% quartile are identical with 0.004 seconds, meaning that the time needed for preprocessing is almost the same on all topologies. Both *Feigenbaum* and *SquareOne* perform almost indistinguishable with an average time of 0.004 seconds, however *Feigenbaum* achieving far better results in routing success as we have seen in the analysis above.

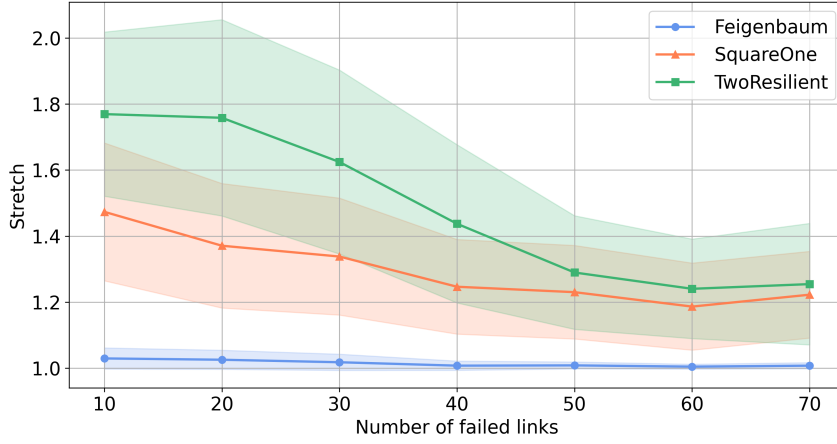


Figure 6.8.: This figure depicts the average normalized stretch on the Topology Zoo networks for a range of clustered disrupted links, **where *Feigenbaum* significantly surpasses the other two algorithms**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 50 individual trials carried out for each failure mark.

6.2.4. Discussion

These results showcase the performance of all three algorithms on realistic network topologies. Both, *Feigenbaum* and *TwoResilient* performed quite well in terms of routing success, outclassing *SquareOne* in all scenarios. In terms of routing path length, *Feigenbaum* outperforms the other two algorithms in both randomized and clustered link failures. Interestingly, *TwoResilient*'s stretch decreased when tested with clustered link failures. This can be explained by the nature of *TwoResilient*'s routing process, which traverses faces of graphs or their components. In certain scenarios where random link fails block the algorithm to proceed routing on a face it has to switch to another face, in some cases taking a major detour. The higher precomputation time of *TwoResilient* can also be explained by the complex preprocessing it has to do for each source and destination pair. Similar to *TwoResilient*, *SquareOne* also needs to perform preprocessing for each source and destination pair, however the calculation are far simpler, as the algorithm only calculates edge disjoint paths. In comparison, *Feigenbaum* only constructs a DAG for a chosen destination, keeping its precomputation time relatively low, since all our tests start from the same destination.

6. Experimental Evaluation

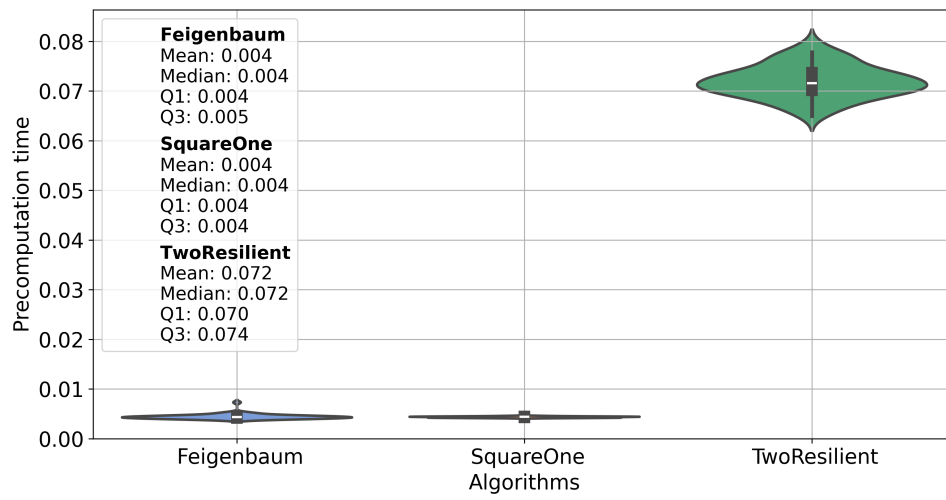


Figure 6.9.: This figure depicts the precomputation time in seconds on the Topology Zoo networks for all algorithms, carried out across 50 individual trials. The width of the violin plots is proportional to the number of observations, **showing a skewed distribution of the violin plots, especially for *Feigenbaum* and *SquareOne*.**

6.3. Experimental Evaluation of Rings of Cliques

After the Zoo topology dataset, which gave us first insights on the performance of the algorithms applied on real world networks, we now shed light at the effectiveness of these algorithms on the synthetic ring of cliques network topology. We randomly generate ring of clique networks having 10 cliques with $k_{clique} = 9$ and $m_{bridge} = 2$. Furthermore the number of nodes $n = 100$ and the number of edges $m = 470$.

6.3.1. Randomized Failures

We start off by evaluating the routing success rate in terms of increasing random link failures. Both *TwoResilient* and *Feigenbaum* can handle approximately 50 link failures with minimal pathfinding issues. Though, both drop in routing performance after 50 link failures, but not significantly, as shown in 6.10. Overall, *Feigenbaum* performs better until around 225 link failures. After this mark both algorithms share the same routing to around 260 link failures, where *TwoResilient* outperforms *Feigenbaum*. In contrast, *SquareOne* already experiences major routing issues when confronted with 50 link failures and keeps getting worse as the number of failed link increases.

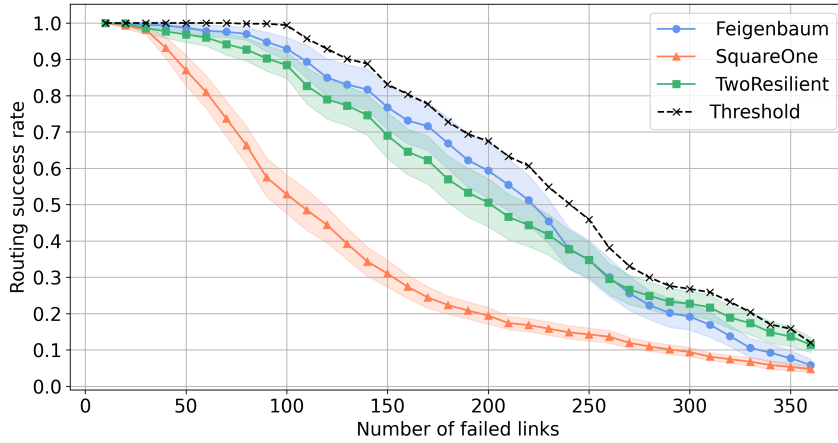


Figure 6.10.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the rings of cliques networks for a range of random disrupted links, **where *SquareOne* experiences major routing issues for the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6. Experimental Evaluation

As before, 6.11 depicts the pathfinding effectiveness of the failover routing schemes with regards to the subset of the nodes that are still connected in the networks components containing the destination after 10 random link disruptions, marked by the threshold. In comparison to the Zoo topology in Figure 6.2, where the dots for *Feigenbaum* and *TwoResilient* were located more in the vicinity of the threshold, now the dots more are scattered beneath the threshold, indicating a higher dispersion in the success rate of each experiment executed on each graph. Interestingly, in almost none of the experiment iterations carried out, *SquareOne* came close to the success rate needed to match the threshold.

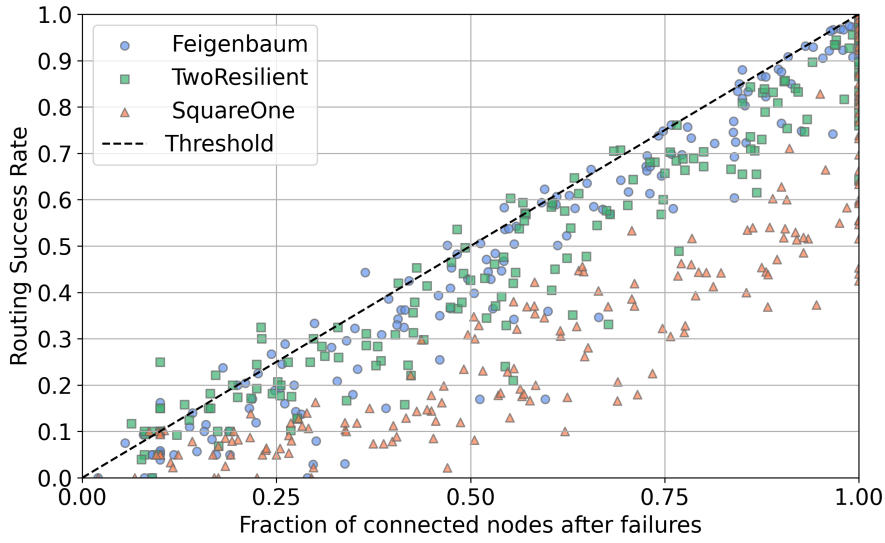


Figure 6.11.: Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the rings of cliques networks, based on the subset of nodes that remain connected after 10 random disrupted links, across 50 individual trials. **Both *Feigenbaum* and *TwoResilient* show a higher dispersion around the threshold.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y .

As far as stretch is concerned, *Feigenbaum* still is the most effective algorithm, by keeping the routing paths optimal despite increasing link failures, as shown in Figure 6.12. *SquareOne*'s stretch ranges between 1.5 and 2 hops and *TwoResilient* again exhibits the largest stretch, with varying between 2 and 3 hops until approximately 200 link failures. However, one interesting observation is, that whereas both *Feigenbaum* and *SquareOne* remain relatively constant in their stretch, *Feigenbaum*'s stretch decreases with increasingly higher link failures.

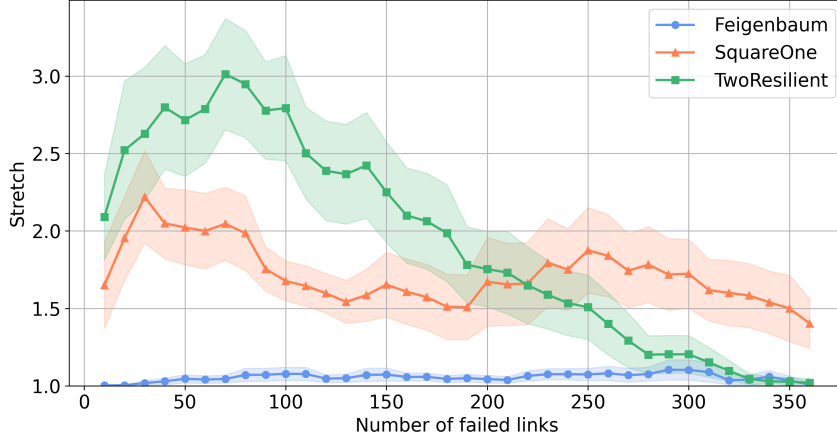


Figure 6.12.: This figure depicts the average normalized stretch on the rings of cliques networks for a range of random disrupted links, **where *Feigenbaum* outperforms the other two algorithms by a large margin and *TwoResilient*, decreases in stretch as failures increase.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

6.3.2. Clustered Failures

In Figure 6.13 we present the outcome of the success rate of the algorithms when confronted with clustered link failures. As we have assumed, *SquareOne* experiences a notable downfall in success from the beginning on. *Feigenbaum* and *TwoResilient* on the other hand perform again almost identical, with only minor differences as it can be seen in Figure 6.14 from 250 to 300 failures. At this failure range *TwoResilient* is slightly better, however, from 300 to 350 failures, *Feigenbaum* is more effective in terms of routing success.

The scatter for *Feigenbaum* and *TwoResilient* has even more increased in Figure 6.14, where we plot the routing success rate as function of the subset of connected nodes remaining after 10 clustered disrupted links, indicating a higher distribution in success rate for each individual experiment performed on each graph. The scatter for *SquareOne* has shifted downwards, indicating that the overall effectiveness of this algorithm is even worse with clustered failures than we have seen with random failures.

When comparing stretch in Figure 6.15 we observe a similar pattern as before, where *Feigenbaum* outperforms the other two algorithms by a huge margin by keeping the stretch to a minimum. *TwoResilient* still remains the algorithm with the largest stretch in general, while *SquareOne* continues its effective stretch between 1.5 and 2 hops on average.

6. Experimental Evaluation

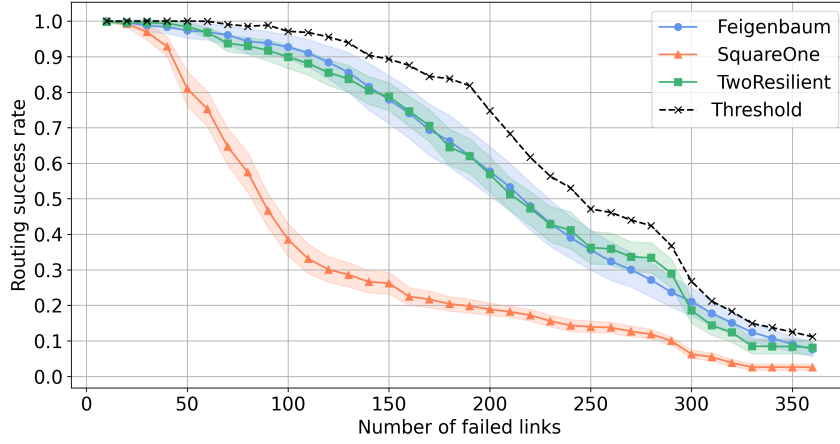


Figure 6.13.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the rings of cliques networks for a range of clustered disrupted links, **where *SquareOne* experiences major routing issues for the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6.3.3. Precomputation Performance

Figure 6.16 depicting the precomputation performance for the Ring of Cliques topology shows similarities with Figure 6.9 depicting the performance for the Zoo Topology, as the distributions mainly for *Feigenbaum* and *SquareOne* are flat, showing almost no variety. Especially *Feigenbaum*'s distribution resembles a horizontal line, which is also reflected by its stochastic parameters for the mean, median and 25% respectively 75% quartile, which all have the value of 0.007 seconds. *TwoResilient*'s precomputation time has the widest distribution compared to the other two algorithms, with an average of 0.04 seconds. *SquareOne*'s average preprocessing time is 0.008 seconds.

6.3.4. Discussion

In general, the performance of all algorithms in regards to the success rate on the Rings of Cliques networks remains similar for *Feigenbaum* and *TwoResilient* to their performance on the Zoo Topology dataset. Both algorithms deliver a solid routing effectiveness. Interestingly on the Rings of Cliques networks, *TwoResilient* performed better than on the Zoo Topology when confronted with clustered failures. Although, *SquareOne* was the least effective algorithm on the Zoo Topology dataset, in terms of success rate, it still provided

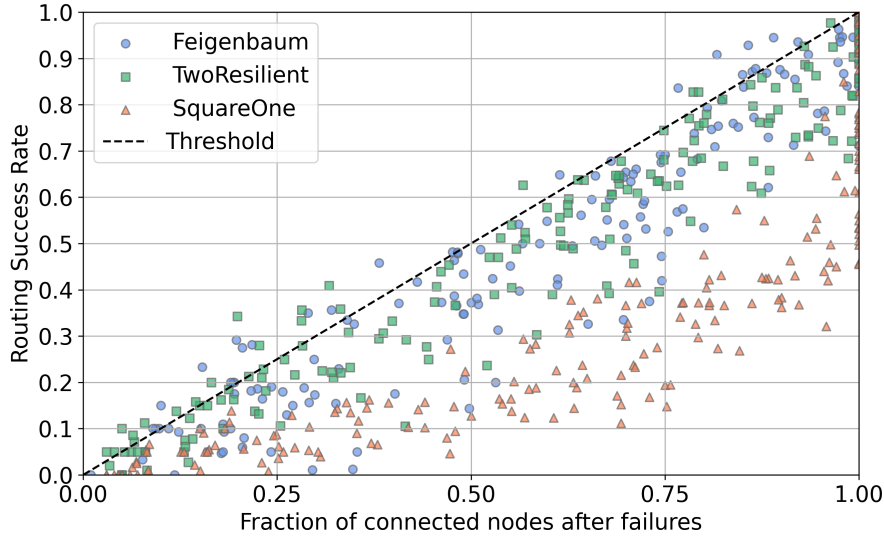


Figure 6.14.: Similar to [FKP⁺21], this chart illustrates the average pathfinding effectiveness on the rings of cliques networks, based on the subset of nodes that remain connected after 10 clustered disrupted links, across 50 individual trials. **Both, *Feigenbaum* and *TwoResilient* show a higher dispersion around the threshold.** The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The coordinates (x, y) on the graph indicate that in a given network topology, where an average of $1 - x$ nodes were disconnected due to 10 random disrupted links, the routing scheme achieved an average pathfinding effectiveness rate of y .

a quite acceptable routing effectiveness, being moderately worse than *TwoResilient* and *Feigenbaum*. However, observing the results of *SquareOne*'s performance on the Rings of Cliques dataset, we can deduct a drastic decline, to the point where routing is not effective anymore. Even though the success rate of both *Feigenbaum* and *TwoResilient* is similar, *Feigenbaum* outmaneuvers *TwoResilient* in stretch and precomputation time of routing tables, being clearly the more efficient algorithm. Furthermore, the low variety in distribution is easily explained when looking at our parameters for graph generation, since all our graphs have the identical amount of nodes $n = 100$ and the identical amount of edges $m = 470$.

6. Experimental Evaluation

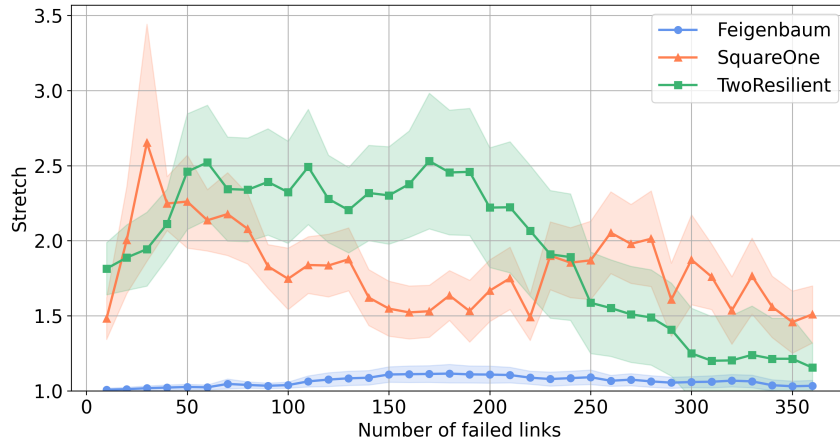


Figure 6.15.: This figure depicts the average normalized stretch on the rings of cliques networks for a range of clustered disrupted links, **where *Feigenbaum* significantly surpasses the other two algorithms and *TwoResilient* shows the largest stretch overall**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

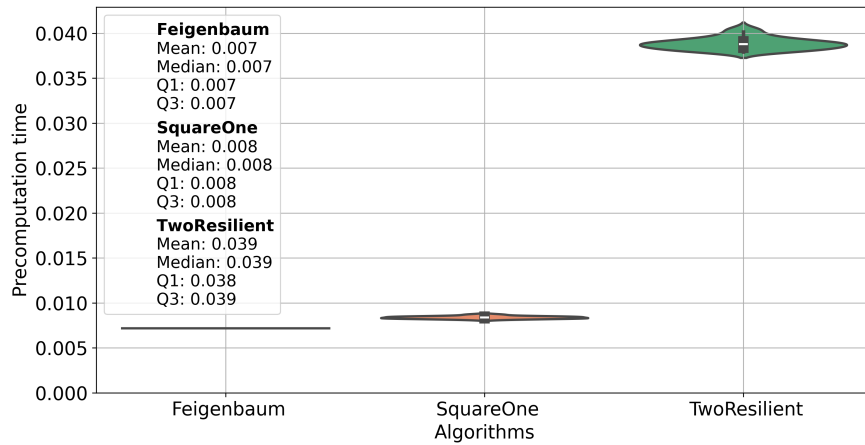


Figure 6.16.: This figure depicts the precomputation time in seconds on the rings of cliques networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, **showing a skewed distribution of the violin plots, especially for *Feigenbaum* and *SquareOne***.

6.4. Experimental Evaluation of Erdős–Rényi Networks

We continue our analysis with the Erdős–Rényi dataset, generated with the probabilistic method $G_{n,p}$ as described in Section 5.1 of Chapter 5. Our graphs all have a node size of $n = 100$ and an edge size between $m = [500, 560]$ and consist of one large connected component having a global connectivity of $k = 1$.

6.4.1. Random Failures

When running our experiments on the Erdős–Rényi networks we observe a noticeable shift in success rate in regards to failures when looking at Figure 6.17. Whereas in our prior testings, *Feigenbaum* and *TwoResilient* have been relatively close to each other, now *TwoResilient* performs better from 50 failures onward than *Feigenbaum*, keeping close distance to the success rate threshold. *SquareOne* remains the least effective algorithm in these terms, already starting with a worse success rate than the other two algorithms when being confronted with merely 10 random link failures in networks with 500 to 560 links.

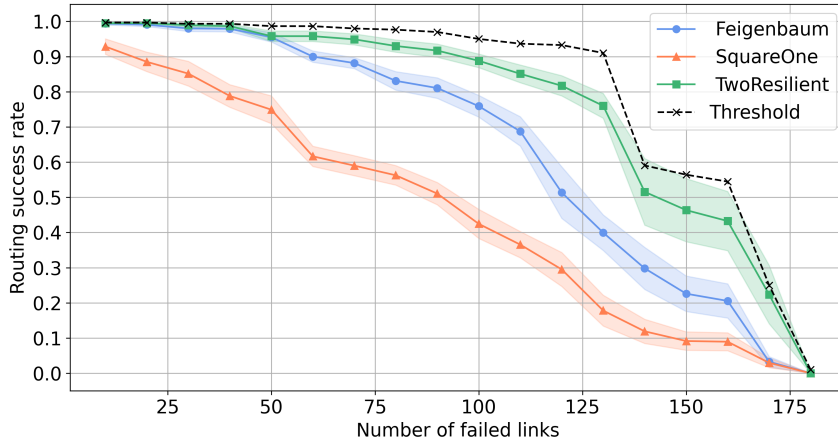


Figure 6.17.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Erdős–Rényi networks for a range of random disrupted links, **where *TwoResilient* performs the best out of all three algorithms**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6. Experimental Evaluation

Even though *TwoResilient* is still not being as efficient as *Feigenbaum* in terms of stretch, we can observe an indubitable increase of stretch for *Feigenbaum*, as it produces a stretch of approximately 1.5 hops between 60 and 110 failures in Figure 6.18. Notably, *SquareOne*, is now generating the longest average stretch when routing from source to destination and *TwoResilient* stays mostly in the interval of 1.5 to 2.5 hops on average than the optimal route.

Figure 6.19 displays the stretch for *TwoResilient* and *Feigenbaum*, however just for runs where both have been successful and only for a success rate equal or higher 50%. We can clearly see that, even though both have been equally successful, *Feigenbaum*, needs less hops to route the package to its destination.

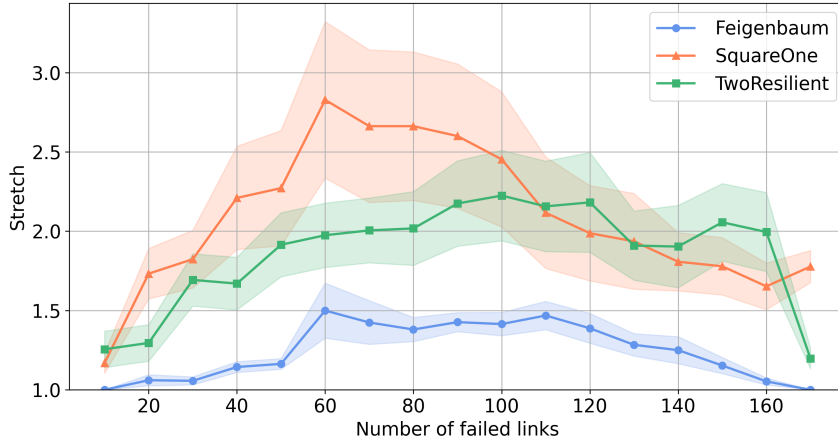


Figure 6.18.: This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, **where *SquareOne* is generating the longest stretch**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

6.4.2. Clustered Failures

Moving on to our results carried out with clustered link failures, Figure 6.20 reveals that until around 50 link failures, *TwoResilient* and *Feigenbaum* perform almost optimally and continue to share a close success rate until 150 link failures, where *TwoResilient* starts to converge towards *SquareOne* and subsequently reaching the success rate of *SquareOne* 200 at clustered link failures.

Astonishingly, *Feigenbaum* in Figure 6.21 displays the hitherto longest stretch reaching 1.75 at 110 failures. *TwoResilient* reaches it's peak stretch at approximately 2.4 and

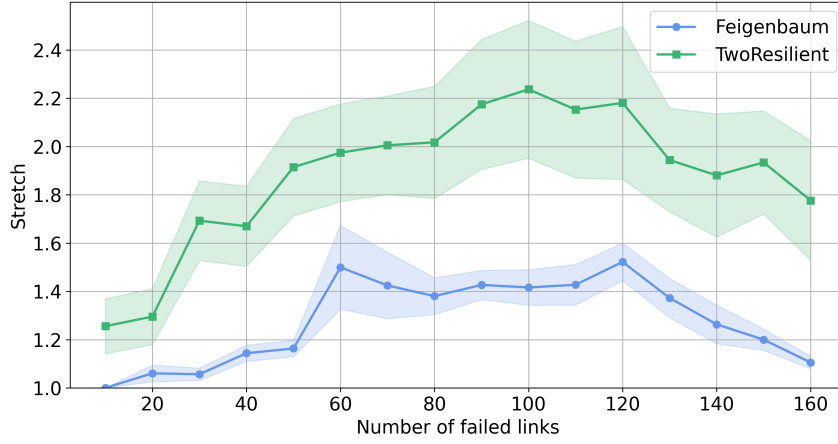


Figure 6.19.: This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, only for runs where both *Feigenbaum* and *TwoResilient* have been successful and had a success rate of ≥ 0.5 . ***Feigenbaum* produces less stretch for the same source-destination pairs compared to *TwoResilient*.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark..

SquareOne at 2.5 hops more than the optimal route on average. Interestingly, all algorithms increase in stretch as the number of clustered failures rises and start to decline at a certain mark, which is different for every algorithm. For *Feigenbaum* the mark is at 110 failures and for *Feigenbaum* and *SquareOne* it is at 100 and 50 clustered link failures.

6.4.3. Precomputation Performance

At the first glance at Figure 6.22 we can highlight the quite unusual distribution for *Feigenbaum*, compared to all our previous figures. Though the mean and median stay at 0.007 seconds and respectively 0.005 seconds, we see that in some cases the precomputation has been significantly higher. *SquareOne* seems to be consistent with the precomputation time, showing a mean and median at 0.008 seconds and having a 25% and 75% quartile of 0.007 to 0.008 seconds. *TwoResilient* needs almost quadruple the time of *Feigenbaum* and *SquareOne* on average by showing off a mean of 0.029 seconds and a median of 0.028 seconds for its precomputation time. All in all, we see that the distributions for all algorithms have increased vertically, compared to the measurements we have conducted in the previous topologies.

6. Experimental Evaluation

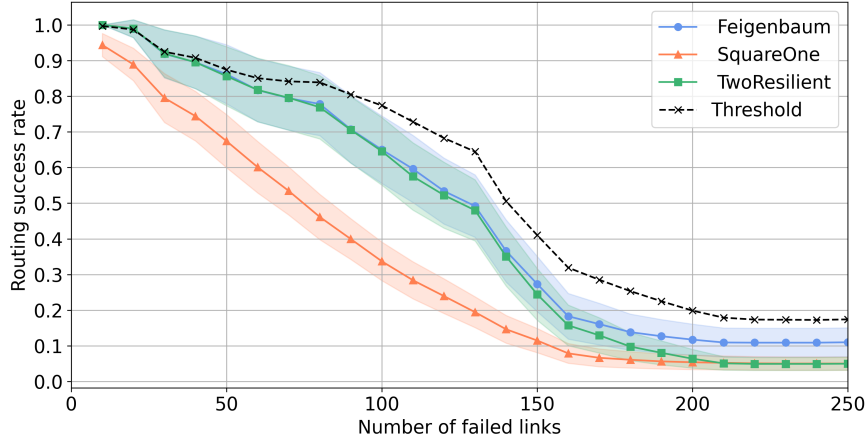


Figure 6.20.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Erdős–Rényi networks for a range of clustered disrupted links, **where *Feigenbaum* and *TwoResilient* performs almost identically**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6.4.4. Discussion

Both *Feigenbaum* and *TwoResilient* perform argumentatively well, however when running the algorithms on our generated Erdős–Rényi graphs, *TwoResilient* seems to have a slight advantage in routing success. Our probabilistic approach generates relatively similar networks, at least in regards to connectivity. *TwoResilient* is highly effective if the network is exactly 2-connected. However, to find a profound explanation we need to dig deeper into our analysis and consider additional parameters when generating our results, however one main factor for the *TwoResilient*'s advantage on this dataset could be the topology of the networks.

6.4. Experimental Evaluation of Erdős–Rényi Networks

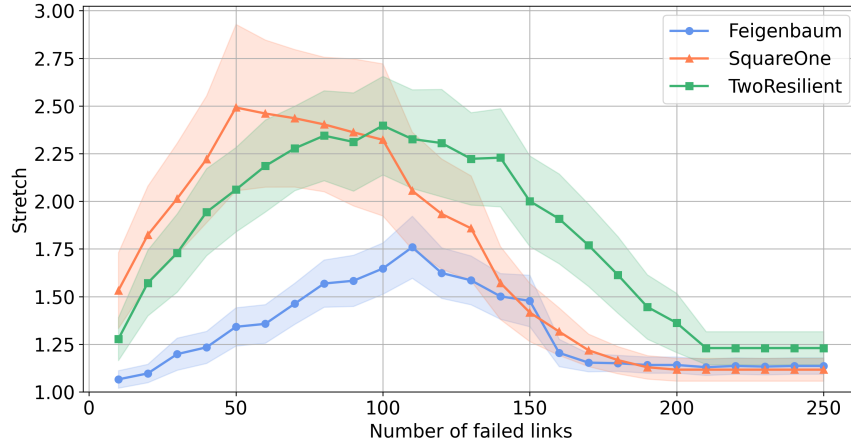


Figure 6.21.: This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of clustered disrupted links, where ***TwoResilient*** displays the largest stretch out of all algorithms. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

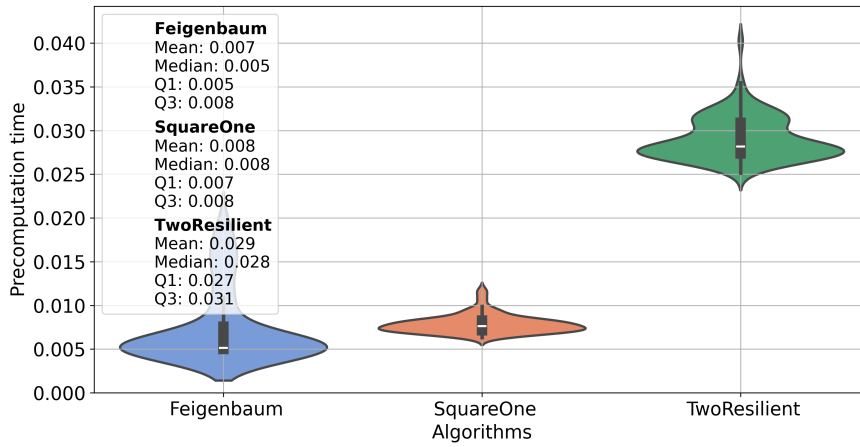


Figure 6.22.: This figure depicts the precomputation time in seconds on the Erdős–Rényis networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, where ***Feigenbaum*** experiences an unusually wide distribution.

6.5. Experimental Evaluation of Regular Networks

Regular graphs are highly homogeneous and therefore provide a uniformity, which offer a predictable environment for studying network algorithms under certain conditions as described in 5.1. In this section of our evaluation we performed our algorithmic analysis on 3-regular graphs also referred to as cubic graphs or trivalent graphs, in which all vertices have a degree of 3. Moreover, our graphs have a node size of $n = 100$ and an edge size of $m = 300$.

6.5.1. Random Failures

As always we start with the random failure model first. Figure 6.23 provides us with some new highly interesting observations, with one of the most noticeable one being the large gap between the threshold and the routing algorithms for the failure range between approximately 20 to 60 failures, indicating that *Feigenbaum* and *TwoResilient* haven't unfolded the full potential of successfully finding a routing path, even though only from a network topology perspective it was possible to do so, by a much higher percentage than they actually could successfully find. Furthermore, see that already at 10 failures *SquareOne* struggles to keep up with the other two algorithms as its success rate significantly underperforms with only 70%, compared to almost 100% for *Feigenbaum* and *TwoResilient*. Also noteworthy is the rapid decline of the fraction of connected nodes denoted by the threshold, after only 50 link failures, which is due to the uniformity of regular graphs. After this, the threshold levels off and keeps gradually declining with the algorithms success rate keeping close to the threshold, indicating only very limited routing possibilities for the algorithms.

We have already seen from our previous results that *TwoResilient* tends to produce the highest stretch on average and *Feigenbaum* tends to do the opposite. Figure 6.24 shows nothing unexpected in that regard, as the results show us the expected outcome of *TwoResilient* having the highest stretch on average and *Feigenbaum* the lowest, with *SquareOne* being in the middle field. For the stretch, we can observe the same phenomenon, we have observed in Figure 6.23, where the stretch drastically declines from 40 link failures onward, due to the uniformity of regular graphs, resulting in identical stretch for all algorithms, as only a few paths at most remain.

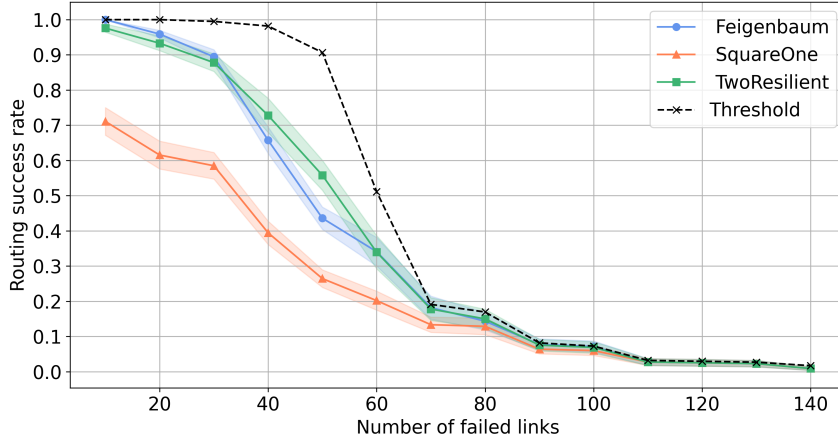


Figure 6.23.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the 3-regular networks for a range of random disrupted links, **where *Feigenbaum* and *TwoResilient* experience routing issues from the beginning**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6.5.2. Clustered Failures

There are two significant differences in Figure 6.25 for clustered failures, compared to Figure 6.23. The first one being the much more moderate decline of success rate, opposed to the much more rapid decline for random failures at the mark of 40 failures. Due to the uniform topology of our regular graphs, where each vertex has a degree of 3, clustered failures have substantially less impact on the network's connectivity, as other areas of the network stay connected, making successfully routing possible even if a certain area that are highly affected by our clustered failures can't be used for routing anymore. The second noticeable observation is the relative identical performance not only for *TwoResilient* and *Feigenbaum*, but also *SquareOne*.

In Figure 6.26 we can see that in terms of stretch *TwoResilient* remains the algorithms producing the highest stretch. Both, *TwoResilient* and *SquareOne* experience a steep increase in stretch at 30 failures. To investigate we take a look at Figure 6.25 and observe an abrupt local decline in success rate from 30 to 40 failures, indicating that the routing paths previously used have been affected by the clustered attack and the algorithms had to take a detour. Surprisingly, *Feigenbaum* has been able to find the ideal route with no rerouting, as it produced no stretch at this failure range. The stretch for *TwoResilient*

6. Experimental Evaluation

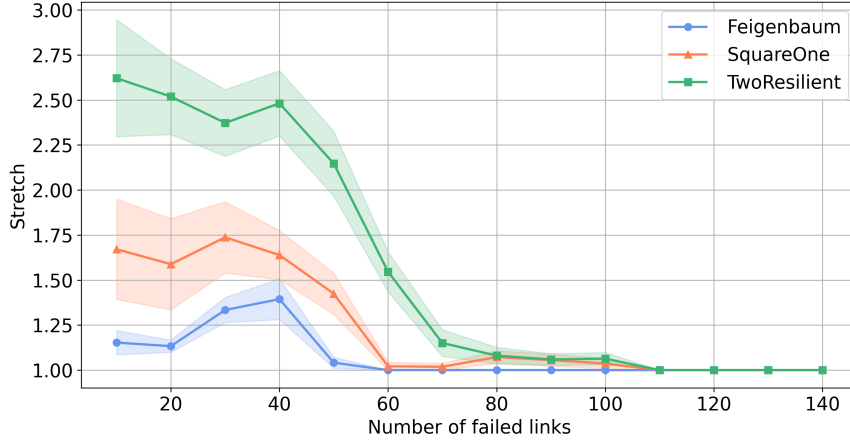


Figure 6.24.: This figure depicts the average normalized stretch on the 3-regular networks for a range of random disrupted links, **where *TwoResilient* produces the highest stretch**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

keeps growing, peaking at almost 3 hops on average at 70 failures, whereas the stretch of *SquareOne* and *Feigenbaum* remain relatively low and steady.

6.5.3. Precomputation Performance

Feigenbaum's and *SquareOne's* precomputation time shown in Figure 6.27 for 3-regular graphs is nearly identical in distribution with both taking on average approximately 0.005 seconds. They also share a median time of 0.005 seconds. *TwoResilient's* precomputation time remains the highest with around 0.017 seconds. Even though all algorithms now exhibit a wider distribution, *TwoResilient* still has the widest, ranging from the 25% quartile of 0.011 seconds to the 75% quartile of 0.022 seconds.

6.5.4. Discussion

From our analysis we can highlight that random link failures have a much more substantial impact on the remaining connectivity of the networks and subsequently the capability of the routing schemes to successfully finding a path from source to destination than it is the case with clustered link failures. Earlier we have already mentioned that this circumstance is due to the uniformity of regular graphs in general. Our clustered link failure model concentrates on a certain area and expands as the amount of failures increase. This makes certain nodes in these impacted areas unreachable, but the majority of the homogeneous cubic graphs remain unaffected, even for higher ranges of link failures. Opposed to random link failures, where we detect that from a certain amount of link failures, in our case from 30 to 70 failures, the connectivity between the nodes in the network declines

6.5. Experimental Evaluation of Regular Networks

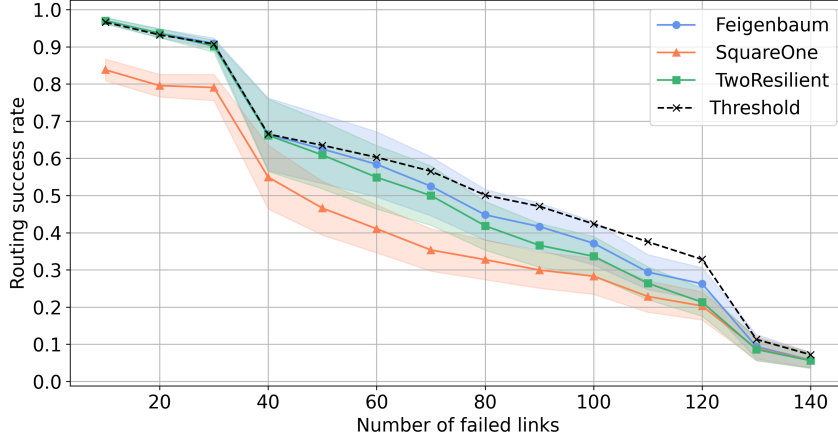


Figure 6.25.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the 3-regular networks for a range of clustered disrupted links, textbfwhere *Feigenbaum* performs slightly better than *TwoResilient*. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 100 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

rapidly, making routing also much more difficult. We even have seen that there is a large discrepancy between the theoretically possible routing success denoted by our threshold and the actual success rate of our routing algorithms in this exact same range.

6. Experimental Evaluation

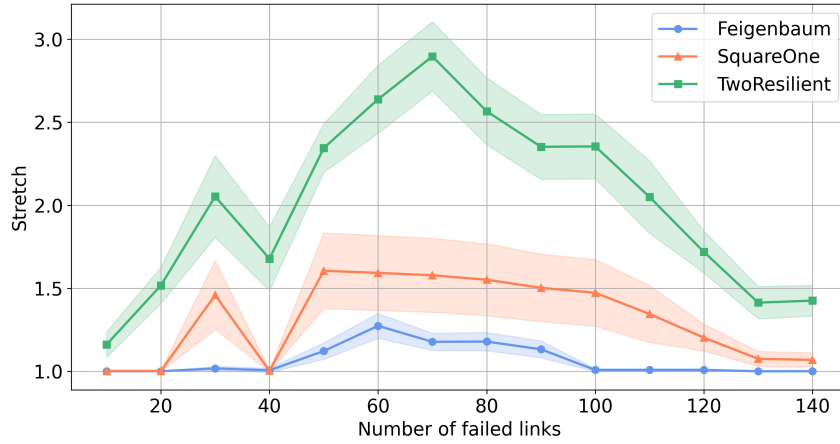


Figure 6.26.: This figure depicts the average normalized stretch on the 3-regular networks for a range of clustered disrupted links, where **TwoResilient** has the highest stretch out of all algorithms. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 100 individual trials carried out for each failure mark.

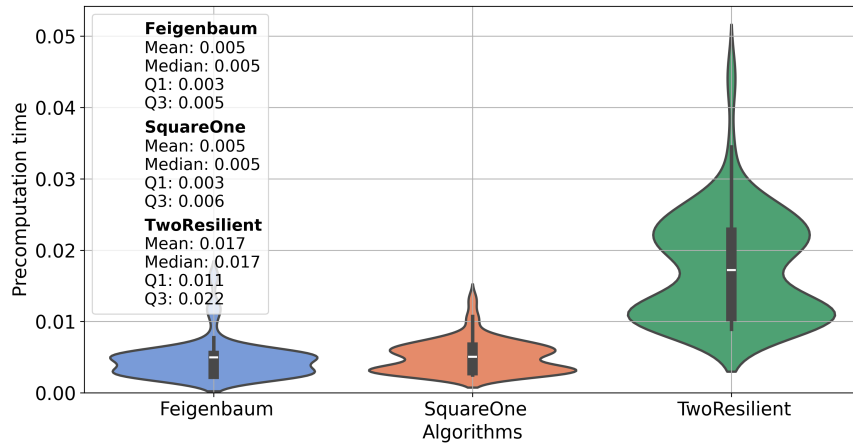


Figure 6.27.: This figure depicts the precomputation time in seconds on the 3-regular networks for all algorithms, carried out across 100 individual trials. The width of the violin plots is proportional to the number of observations, where **TwoResilient** has a wide distribution range.

6.6. Experimental Evaluation of Rocketfuel Networks

Finally, we evaluate the performance of the algorithms using the Rocketfuel networks. As described in Section 5.1 of Chapter 5, these networks have been created from ISP topology mappings and are compared to regular networks we have discussed before, highly heterogeneous reflecting real world networks. The networks we have used for our evaluation range in between $n = [110, 300]$ and $m = [400, 1200]$.

6.6.1. Random Failures

As we can see in Figure 6.28, the threshold for the connected nodes in the network is steady at 100% connectivity until 200 link failures occur, whereas the success rate of the algorithms gradually declines as the number of failures increases. From 200 to 250 link failures there is a swift decline in both the node connectivity and success rate until it again stabilizes at 250 to 400 link failures. In the first section from 50 to 200 link failures *TwoResilient* has a slightly better performance, beginning from 200 failures onward *Feigenbaum*'s success rate improves over the success rate of *TwoResilient*. *SquareOne* continues its pattern of performing worse than the other two algorithms, with no surprise considering its simplistic routing approach.

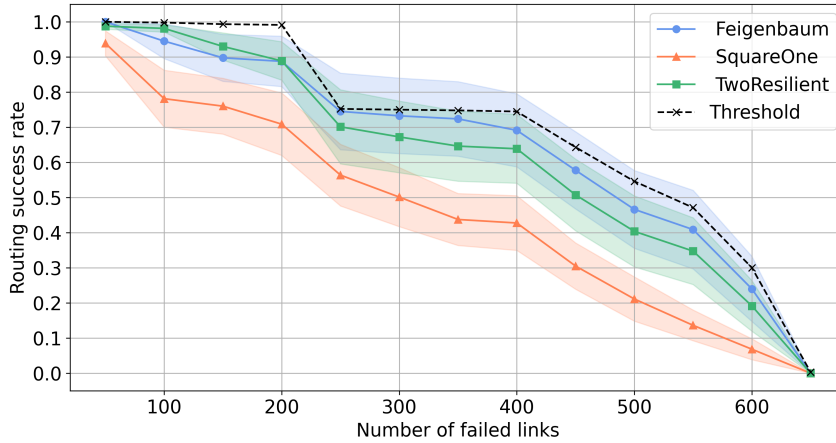


Figure 6.28.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Rocketfuel networks for a range of random disrupted links, **where *Feigenbaum* is slightly ahead of *TwoResilient***. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 10 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6. Experimental Evaluation

Unexpectedly, when observing the stretch for the Rocketfuel dataset in Figure 6.29, *TwoResilient* produced less stretch on average than *SquareOne* as it was already the case for Erdős–Rényi graphs in Figure 6.18. Also, *Feigenbaum* continues to perform well not only in terms of success rate but also in terms of stretch as it keeps detours to a minimum or finds the shortest available path right away.

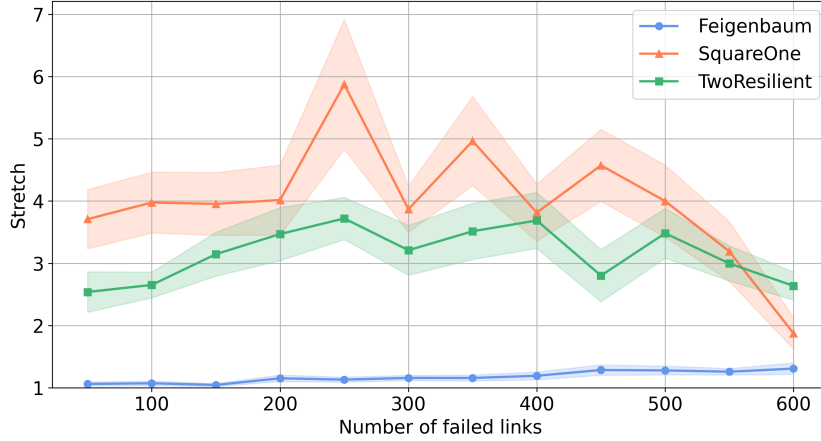


Figure 6.29.: This figure depicts the average normalized stretch on the Rocketfuel networks for a range of random disrupted links, **where *Feigenbaum* displays the lowest and *SquareOne* the highest stretch**. The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark.

6.6.2. Clustered Failures

Taking a peak at our results generated with clustered failures in Figure 6.30 we see that *TwoResilient* keeps close track to the threshold, therefore making it the best performing algorithm in this experiment. *Feigenbaum*, on the other hand, experiences routing issues, which is unusual considering the majority of the results of our prior experiments. Unarguably the worst performance is delivered by *SquareOne*, which declines promptly as the failures increase.

Considering stretch in Figure 6.31, *Feigenbaum* still performs the best, by keeping it close to 1 meaning, the shortest path was taken or no detours occurred. *TwoResilient* keeps a steady stretch of 2 to 2.5 hops on average, which is comparatively lower than with random link failures in Figure 6.29, where it produced a stretch of 3 to 4 hops on average. *SquareOne* has to take inefficient detours as we can recognize at the high stretch in the range of 50 to 550 link failures.

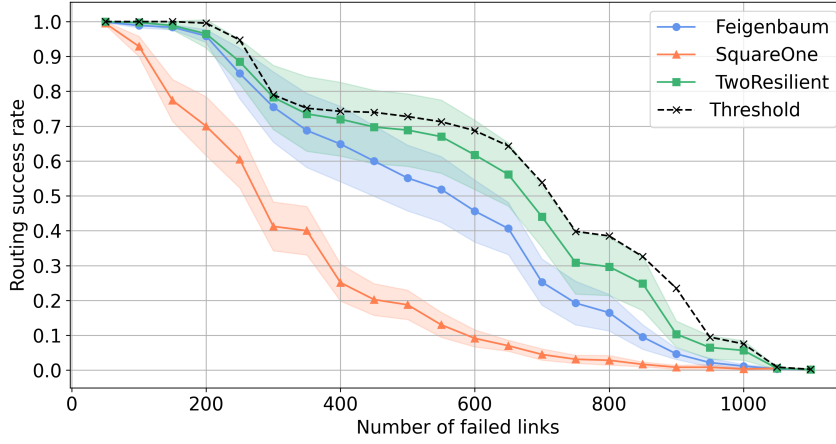


Figure 6.30.: Similar to [FKP⁺21], this figure depicts the average pathfinding effectiveness on the Rocketfuel networks for a range of clustered disrupted links, **where *TwoResilient* is the best performing algorithms**. The threshold signifies the average percentage of nodes that remain connected to the destination component after disruptions, representing the statistical ceiling for the effectiveness. The shaded bands illustrate $\frac{1}{4}$ of the standard deviation across 10 individual trials for each failure point, which may exceed the threshold due to statistical variance. Increasing the number of repetitions can yield a more accurate standard deviation.

6.6.3. Precomputation Performance

Compared to Figure 6.9 for the Zoo dataset, where all algorithms had a small distribution variety, we can see that now for the Rocketfuel dataset in Figure 6.32 the distribution range is much wider, especially for *Feigenbaum* and *SquareOne*. This is interesting as both datasets are based on real world topologies. For all algorithms the broadest density is centered around the median, which is 0.014 seconds for *Feigenbaum*, 0.02 seconds for *SquareOne* and 0.06 seconds for *TwoResilient*.

6.6.4. Discussion

The Rocketfuel dataset gives us valuable insights into the real world performance of the algorithms, as they are mappings of ISP topologies. Random failures don't seem to affect the routing capability of the algorithms as much as clustered failures do in this experimental setup, with the exception of *TwoResilient*, which delivered a favorable performance in both cases when regarding the success rate. However, in other aspects such as the stretch and precomputation time, *Feigenbaum* continues to outperform *TwoResilient* by a large margin, by keeping the stretch to a minimum and the precomputation time low. Disregarding the precomputation time, *SquareOne* is the least favorable algorithm

6. Experimental Evaluation

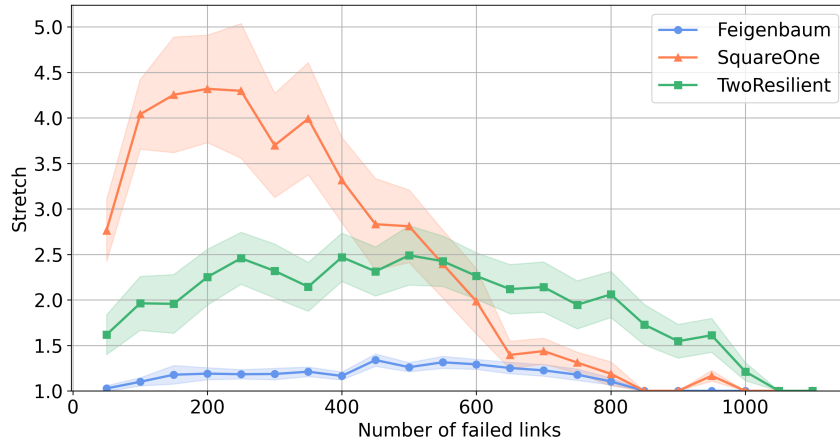


Figure 6.31.: This figure depicts the average normalized stretch on the Rocketfuel networks for a range of clustered disrupted links, **where *SquareOne* has the highest stretch, due to long detours.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark.

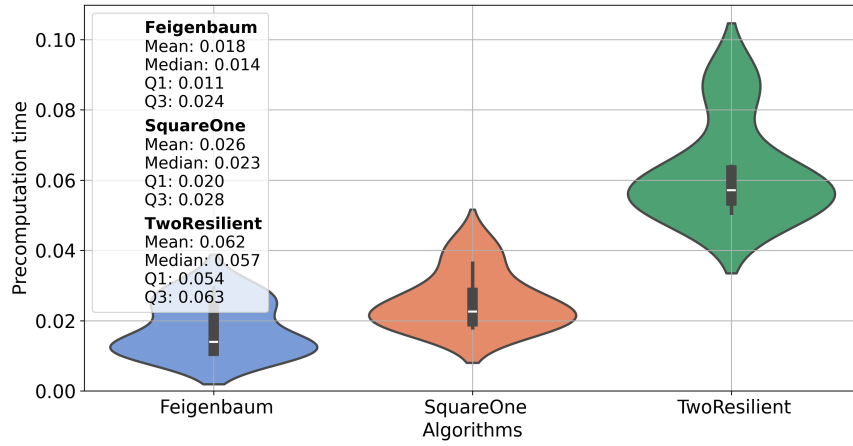


Figure 6.32.: This figure depicts the precomputation time in seconds on the Rocketfuel networks for all algorithms, carried out across 10 individual trials. The width of the violin plots is proportional to the number of observations, **where all algorithms have a wider distribution.**

out of the three, as its success rate doesn't come even close to the other two algorithms and also produces high and irregular stretch.

6.7. Recapitulation and Discussion of Evaluation Results

Summarizing our evaluation results we can conclude that *SquareOne* is overall the least effective routing algorithm out of the three. In terms of the average success rate *TwoResilient*'s and *Feigenbaum*'s performance is close, with meager differences on most topologies. Regarding the average normalized stretch, that is the amount of additional hops taken relative to the possible shortest path in the network, *Feigenbaum* undoubtedly performs the best, by keeping the stretch close to one, in almost all scenarios.

- **Feigenbaum** delivers a consistent and reliable performance in all evaluation aspects. Regarding success rate it is on par with *TwoResilient*, on most of the topologies evaluated. Where *Feigenbaum* really shows its strength and outperforms the other two algorithms, is in the stretch and precomputation time needed. None of the other algorithms keeps the stretch as low as *Feigenbaum* does. Same can be observed regarding the precomputation time, as it is the lowest in all our test cases.
- **TwoResilient**'s success rate is comparable to the of *Feigenbaum*, as we concluded above. However, one curious observation we have made is that in cases where *TwoResilient* performs better, it is by a significant amount as can be seen in Figure 6.17 on the Erdős-Rényi networks and Figure 6.30. There are no cases where *Feigenbaum* performs significantly better than *TwoResilient*. Unfortunately, this can't be stated when discussing the stretch for *TwoResilient*, as it doesn't even come close to *Feigenbaum*. Same can be stated for the precomputation time of *TwoResilient*, which is unsurprisingly high, due to its complexity of finding and processing connected components.
- **SquareOne**'s performance doesn't shine neither in routing success nor in stretch. The precomputation time for *SquareOne* is low compared to *TwoResilient*, however still higher as the preprocessing time of *Feigenbaum*, which yields far better results.

To why *TwoResilient* produces exceedingly more stretch than *Feigenbaum*, we need to conduct a deeper analysis, nonetheless, there are three factors which are undeniably responsible for this fact. On one hand, if a connected component isn't exactly two connected, *TwoResilient* will switch to an algorithm similar to *SquareOne*, where it calculates edge disjoint paths and successively tries to route on them, starting with the shortest one. On the other hand, if the component is exactly two connected and planar for that matter, *TwoResilient* routes on the area of the plane bounded by edges, also known as faces. *TwoResilient* also needs a source and destination to precompute a routing scheme, where *Feigenbaum* only needs a destination, which provably yields a lesser stretch.

Moreover, we have seen that *TwoResilient* and *Feigenbaum* perform alike and in some cases *TwoResilient* performs noticeably better. Our assumption is that in the cases where *TwoResilient* displayed a better performance, it encountered more two connected components and therefore could unfold its full potential. However a more in detail study on i.e. only 2-connected networks is needed to investigate this assumption.

6. Experimental Evaluation

We have seen that in regards to precalculation time, *Feigenbaum* is undoubtedly the superior algorithm. However, as we mentioned at the beginning of this section, *Feigenbaum* only needs to preprocess the network for a given destination, as opposed to *TwoResilient* or *SquareOne*, where both algorithms need to preprocess the network for a given source-destination pair, as we already mention above.

In our conducted tests we only changed the source node, whereas the destination node stayed fixed. This circumstance may have led to skewed results for precomputation time for *Feigenbaum*. Measuring the precalculation time with an always varying destination may have an impact on *Feigenbaum's* performance in these terms.

7. Conclusion and Future Work

This chapter concludes our analysis of the three presented fast failover routing algorithms in Section 7.1 and moreover discusses possible future work and enhancements to further improve the *TwoResilient* algorithm in Section 7.2.

7.1. Conclusion

In this paper, we have performed a thorough evaluation of the three fast failover algorithms *SquareOne*, *Feigenbaum* and *TwoResilient*. Motivated by their guarantees in terms of resilience against k simultaneous link failures for packet delivery, with the premise that the source and destination stay connected, we measured the performance of these algorithms in terms of fault tolerance, stretch and precomputation time on five diverse topologies, simulating random as well as targeted link failures to capture a realistic scenario. Under these conditions we have seen that, due to its simplicity, *SquareOne* provides a relatively short precomputation time, however performs worse than *Feigenbaum* or *TwoResilient* in terms of fault tolerance and stretch.

As far as fault tolerance and stretch are concerned, *Feigenbaum* delivers solid results, by keeping the tolerance near the statistical upper bound of what is achievable and reducing the stretch to a minimum.

TwoResilient's complex calculations for routing tables on the provided networks are reflected in its longer precomputation time. The algorithms performance in terms of fault tolerance is on par with *Feigenbaum* or marginally better on certain topologies such as on Erdős–Rényi networks with random failures. This, however, can't be said about the stretch, as *TwoResilient* takes longer detours than *Feigenbaum*.

Moreover, we have provided the first available implementation of the *TwoResilient* algorithm written in Python. In order to foster reproducibility and encourage future research in this field, we make the source code for both the algorithm and our evaluation framework publicly available [Zab24].

7. Conclusion and Future Work

7.2. Future Work

We believe our research paves the way for numerous intriguing future studies. Specifically, further refinement of the algorithms we discussed would be particularly beneficial, both analytically and empirically. For example, as we have seen, the *TwoResilient* algorithm already shows satisfactory results on most of the topologies, however, by reapplying the algorithm on the remaining structure of the graph as long as the connectivity stays higher or equal to two, we believe to even further enhance the resilience of this routing scheme against link failures.

Another simple enhancement can be applied if the provided graph has a connectivity of greater than two. In this case we normally, wouldn't apply *TwoResilient's* core algorithm, but rather compute edge-disjoint paths, sort them by length and consecutively try them out. This approach, can produce additional stretch as, we have to walk to the starting source node of each path in the case we encounter a failed link. By finding a spanning tree connecting these paths, we could improve the stretch as well as the success rate, as we do not need to walk back to the source node and have additional links available connecting each path.

Furthermore, to improve and simplify the *TwoResilient* algorithm in hindsight of pre-computation time and stretch, a potential solution is the integration of ear decomposition [Sch13].

Lastly, our testing framework also offers multiple opportunities for enhancements and improvements. One of such enhancements could be the routing performance, as performance can be measured manifold. In our evaluation we have mainly focused on the metrics of fault tolerance, stretch and precomputation time. Metrics such as the congestion, often referred to as load, could be also brought into consideration. Other improvements or enhancements could include new attack vectors, topologies and even comparisons with other failover routing algorithms such as DAG-FRR or Custer-FRR, studied in [FKP⁺21].

Bibliography

- [AGM⁺10] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In Shivkumar Kalyanaraman, Venkata N. Padmanabhan, K. K. Ramakrishnan, Rajeev Shorey, and Geoffrey M. Voelker, editors, *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New Delhi, India, August 30 -September 3, 2010*, pages 63–74. ACM, 2010.
- [AZ08] Alia Atlas and Alex Zinin. Basic specification for IP fast reroute: Loop-free alternates. *RFC*, 5286:1–31, 2008.
- [Bon24] Olivier Bonaventure. Computer networking : Principles, protocols and practice. <https://beta.computer-networking.info/syllabus/default/t/principles/network.html>, 2024. Accessed: 2024-04-31.
- [BS13] Michael Borokhovich and Stefan Schmid. How (not) to shoot in your foot with SDN local fast failover - A load-connectivity tradeoff. In Roberto Baldoni, Nicolas Nisse, and Maarten van Steen, editors, *Principles of Distributed Systems - 17th International Conference, OPODIS 2013, Nice, France, December 16-18, 2013. Proceedings*, volume 8304 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2013.
- [BSS14] Michael Borokhovich, Liron Schiff, and Stefan Schmid. Provable data plane connectivity with local fast failover: introducing openflow graph algorithms. In Aditya Akella and Albert G. Greenberg, editors, *Proceedings of the third workshop on Hot topics in software defined networking, HotSDN '14, Chicago, Illinois, USA, August 22, 2014*, pages 121–126. ACM, 2014.
- [BST03] Costas Busch, Srikanth Surapaneni, and Srikanta Tirthapura. Analysis of link reversal routing algorithms for mobile ad hoc networks. In Arnold L. Rosenberg and Friedhelm Meyer auf der Heide, editors, *SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003)*, pages 210–219. ACM, 2003.
- [CE95] M. Scott Corson and Anthony Ephremides. A distributed routing algorithm for mobile wireless networks. *Wirel. Networks*, 1(1):61–81, 1995.

Bibliography

- [CGM⁺16] Marco Chiesa, Andrei V. Gurtov, Aleksander Madry, Slobodan Mitrovic, Ilya Nikolaevskiy, Michael Schapira, and Scott Shenker. On the resiliency of randomized routing against multiple edge failures. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 134:1–134:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
- [CKLS15] Marco Canini, Petr Kuznetsov, Dan Levin, and Stefan Schmid. A distributed and robust SDN control plane for transactional network updates. In *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015*, pages 190–198. IEEE, 2015.
- [CKR⁺21] Marco Chiesa, Andrzej Kamisinski, Jacek Rak, Gábor Rétvári, and Stefan Schmid. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Commun. Surv. Tutorials*, 23(2):1253–1301, 2021.
- [CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- [CNM⁺16] Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Aurojit Panda, Andrei V. Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. The quest for resilient (static) forwarding tables. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, pages 1–9. IEEE, 2016.
- [CNM⁺17] Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Andrei V. Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. On the resiliency of static forwarding tables. *IEEE/ACM Trans. Netw.*, 25(2):1133–1146, 2017.
- [CNP⁺14] Marco Chiesa, Ilya Nikolaevskiy, Aurojit Panda, Andrei V. Gurtov, Michael Schapira, and Scott Shenker. Exploring the limits of static failover routing. *CoRR*, abs/1409.0034, 2014.
- [CSA⁺19] Marco Chiesa, Roshan Sedar, Gianni Antichi, Michael Borokhovich, Andrzej Kamisinski, Georgios Nikolaidis, and Stefan Schmid. PURR: a primitive for reconfigurable fast reroute: hope for the best and program for the worst. In Aziz Mohaisen and Zhi-Li Zhang, editors, *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, CoNEXT 2019, Orlando, FL, USA, December 09-12, 2019*, pages 1–14. ACM, 2019.
- [CSA⁺21] Marco Chiesa, Roshan Sedar, Gianni Antichi, Michael Borokhovich, Andrzej Kamisinski, Georgios Nikolaidis, and Stefan Schmid. Fast reroute on programmable switches. *IEEE/ACM Trans. Netw.*, 29(2):637–650, 2021.

- [dev24a] NetworkX developers. Edmond karp algorithm, networkx. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.flow.edmonds_karp.html, 2024. Accessed: March 2024.
- [dev24b] NetworkX developers. Networkx. <https://networkx.org>, 2024. Accessed: March 2024.
- [dev24c] NetworkX developers. Planarembdding, networkx. <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.planarity.PlanarEmbedding.html>, 2024. Accessed: March 2024.
- [DFS23] Wenkai Dai, Klaus-Tycho Foerster, and Stefan Schmid. A tight characterization of fast failover routing: Resiliency to two link failures is possible. In Kunal Agrawal and Julian Shun, editors, *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2023, Orlando, FL, USA, June 17-19, 2023*, pages 153–163. ACM, 2023.
- [EGR16] Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. IP fast rerouting for multi-link failures. *IEEE/ACM Trans. Netw.*, 24(5):3014–3025, 2016.
- [ER60] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.
- [ERC07] Gábor Enyedi, Gábor Rétvári, and Tibor Cinkler. A novel loop-free IP fast reroute algorithm. In Aiko Pras and Marten van Sinderen, editors, *Dependable and Adaptable Networks and Services, 13th Open European Summer School and IFIP TC6.6 Workshop, EUNICE 2007, Enschede, The Netherlands, July 18-20, 2007, Proceedings*, volume 4606 of *Lecture Notes in Computer Science*, pages 111–119. Springer, 2007.
- [FGP⁺12] Joan Feigenbaum, Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. On the resilience of routing tables. *CoRR*, abs/1207.3732, 2012.
- [FHP⁺21] Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. On the feasibility of perfect resilience with local fast failover. In Michael Schapira, editor, *2nd Symposium on Algorithmic Principles of Computer Systems, APOCS 2020, Virtual Conference, January 13, 2021*, pages 55–69. SIAM, 2021.
- [FKP⁺19] Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Bonsai: Efficient fast failover routing using small arborescences. In *49th Annual IEEE/IFIP International Conference*

Bibliography

- on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 276–288. IEEE, 2019.
- [FKP⁺21] Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Grafting arborescences for extra resilience of fast rerouting schemes. In *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*, pages 1–10. IEEE, 2021.
- [FMB⁺11] Clarence Filsfils, Pradosh Mohapatra, John Bettink, Pranav Dharwadkar, Peter De Vriendt, Yuri Tsier, Virginie Van Den Schrieck, Olivier Bonaventure, and Pierre Francois. Bgp prefix independent convergence (pic) technical report. Technical report, Cisco, 2011.
- [FPCS18] Klaus-Tycho Foerster, Mahmoud Parham, Marco Chiesa, and Stefan Schmid. TI-MFA: keep calm and reroute segments fast. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, April 15-19, 2018*, pages 415–420. IEEE, 2018.
- [FPST18] Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Local fast failover routing with low stretch. *Comput. Commun. Rev.*, 48(1):35–41, 2018.
- [FPST19] Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. CASA: congestion and stretch aware static fast rerouting. In *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*, pages 469–477. IEEE, 2019.
- [GB81] Eli Gafni and Dimitri P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Trans. Commun.*, 29(1):11–18, 1981.
- [GJN11] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In Srinivasan Keshav, Jörg Liebeherr, John W. Byers, and Jeffrey C. Mogul, editors, *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011*, pages 350–361. ACM, 2011.
- [GMK⁺16] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In Marinho P. Barcellos, Jon Crowcroft, Amin Vahdat, and Sachin Katti, editors, *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*, pages 58–72. ACM, 2016.

- [HCCR09] Prashanth Hande, Mung Chiang, A. Robert Calderbank, and Sundeep Rangan. Network pricing and rate allocation with content provider participation. In *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*, pages 990–998. IEEE, 2009.
- [HMvdW⁺20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Pícus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. Accessed: March 2024.
- [HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
- [Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. Accessed: March 2024.
- [ISO02] ISO. Intermediate Ststem-to-Intermediate System (IS-IS) Routing Protocol. ISO/IEC 10589, 2002.
- [Jar13] Aubin Jarry. Fast reroute paths algorithms. *Telecommun. Syst.*, 52(2):881–888, 2013.
- [JKM⁺18] Jesper Stenbjerg Jensen, Troels Beck Krøgh, Jonas Sand Madsen, Stefan Schmid, Jiri Srba, and Marc Tom Thorgersen. P-rer: fast verification of MPLS networks with multiple link failures. In Xenofontas A. Dimitropoulos, Alberto Dainotti, Laurent Vanbever, and Theophilus Benson, editors, *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2018, Heraklion, Greece, December 04-07, 2018*, pages 217–227. ACM, 2018.
- [JRT20] Chuan Jiang, Sanjay G. Rao, and Mohit Tawarmalani. PCF: provably resilient flexible routing. In Henning Schulzrinne and Vishal Misra, editors, *SIGCOMM '20: Proceedings of the 2020 Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, Virtual Event, USA, August 10-14, 2020*, pages 139–153. ACM, 2020.
- [Kam18] Andrzej Kamisinski. Evolution of IP fast-reroute strategies. In *10th International Workshop on Resilient Networks Design and Modeling, RNDM*

Bibliography

- 2018, Longyearbyen, Svalbard, Norway, August 27-29, 2018, pages 1–6. IEEE, 2018.
- [KNF⁺11] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765–1775, october 2011.
- [LCR⁺07] Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. Achieving convergence-free routing using failure-carrying packets. In Jun Murai and Kenjiro Cho, editors, *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, August 27-31, 2007*, pages 241–252. ACM, 2007.
- [LKM⁺14] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. Traffic engineering with forward fault correction. In Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy, editors, *ACM SIGCOMM 2014 Conference, SIGCOMM’14, Chicago, IL, USA, August 17-22, 2014*, pages 527–538. ACM, 2014.
- [LPS⁺13] Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. Ensuring connectivity via data plane mechanisms. In Nick Feamster and Jeffrey C. Mogul, editors, *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013, Lombard, IL, USA, April 2-5, 2013*, pages 113–126. USENIX Association, 2013.
- [MIB⁺08] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational IP backbone network. *IEEE/ACM Trans. Netw.*, 16(4):749–762, 2008.
- [Moy98] John Moy. OSPF version 2. *RFC*, 2328:1–244, 1998.
- [MS15] Michael Markovitch and Stefan Schmid. SHEAR: A highly available and flexible network architecture marrying distributed and logically centralized control planes. In *23rd IEEE International Conference on Network Protocols, ICNP 2015, San Francisco, CA, USA, November 10-13, 2015*, pages 78–89. IEEE Computer Society, 2015.
- [NLY⁺07] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, and Chen-Nee Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Trans. Netw.*, 15(2):359–372, 2007.
- [ONF12] OpenFlow Switch Specification. Technical Report ONF TS-007, Open Networking Foundation, September 2012. Wire Protocol 0x04.

- [oW24] University of Washington. Rocketfuel: An isp topology mapping engine. <https://research.cs.washington.edu/networking/rocketfuel/>, 2024. Accessed: 2024-04-31.
- [pdt20] The pandas development team. pandas-dev/pandas: Pandas. <https://pandas.pydata.org/>, February 2020. Accessed: March 2024.
- [pip] Python package index - pypi. <https://pypi.org/project/pip/21.3.1/>. Accessed: March 2024.
- [PJ13] Rahul Potharaju and Navendu Jain. When the network crumbles: an empirical study of cloud network failures and their impact on services. In Guy M. Lohman, editor, *ACM Symposium on Cloud Computing, SOCC '13, Santa Clara, CA, USA, October 1-3, 2013*, pages 15:1–15:17. ACM, 2013.
- [PSA05] Ping Pan, George Swallow, and Alia Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. *RFC*, 4090:1–38, 2005.
- [PSM⁺18] J. Papán, P. Segeč, M. Moravčík, M. Kontšek, L. Mikuš, and J. Uramová. Overview of ip fast reroute solutions. In *ICETA*, pages 417–424, 2018.
- [PST17] Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. Load-optimal local fast rerouting for resilient networks. In *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017, Denver, CO, USA, June 26-29, 2017*, pages 345–356. IEEE Computer Society, 2017.
- [Pyt24] Python. Python 3.9.0. <https://www.python.org/downloads/release/python-390/>, 2024. Accessed: March 2024.
- [RTEC11] Gábor Rétvári, János Tapolcai, Gábor Enyedi, and András Császár. IP fast reroute: Loop free alternates revisited. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 2948–2956. IEEE, 2011.
- [Sch13] Jens M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Inf. Process. Lett.*, 113(7):241–244, 2013.
- [SFS21] Oliver Schweiger, Klaus-Tycho Foerster, and Stefan Schmid. Improving the resilience of fast failover routing: TREE (tree routing to extend edge disjoint paths). In *ANCS '21: Symposium on Architectures for Networking and Communications Systems, Lafayette, IN, USA, December 13 - 16, 2021*, pages 1–7. ACM, 2021.

Bibliography

- [SIG⁺02] Aman Shaikh, Chris Isett, Albert G. Greenberg, Matthew Roughan, and Joel Gottlieb. A case study of OSPF behavior in a large enterprise network. In *Proceedings of the 2nd ACM SIGCOMM Internet Measurement Workshop, IMW 2002, Marseille, France, November 6-8, 2002*, pages 217–230. ACM, 2002.
- [SYR05] Lu Shen, Xi Yang, and Byrav Ramamurthy. Shared risk link group (srlg)-diverse path provisioning under hybrid service level agreements in wavelength-routed optical mesh networks. *IEEE/ACM Trans. Netw.*, 13(4):918–931, 2005.
- [TLSS10] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage. California fault lines: understanding the causes and impact of network failures. In Shivkumar Kalyanaraman, Venkata N. Padmanabhan, K. K. Ramakrishnan, Rajeev Shorey, and Geoffrey M. Voelker, editors, *Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, New Delhi, India, August 30 -September 3, 2010*, pages 315–326. ACM, 2010.
- [VCR15] Amin Vahdat, David Clark, and Jennifer Rexford. A purpose-built global network: Google’s move to SDN. *ACM Queue*, 13(8):100, 2015.
- [VGO⁺20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. Accessed: March 2024.
- [VHV12] Balajee Vamanan, Jahangir Hasan, and T. N. Vijaykumar. Deadline-aware datacenter tcp (D2TCP). In Lars Eggert, Jörg Ott, Venkata N. Padmanabhan, and George Varghese, editors, *ACM SIGCOMM 2012 Conference, SIGCOMM ’12, Helsinki, Finland - August 13 - 17, 2012*, pages 115–126. ACM, 2012.
- [VRD09] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
- [Was21] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. Accessed: March 2024.

- [YLS⁺14] Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. Keep forwarding: Towards k-link failure resilient routing. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 1617–1625. IEEE, 2014.
- [Zab24] Philipp Zabka. Fast failover evaluation. https://github.com/phili ppzabka/failover_evaluation_2024/tree/main, 2024. Accessed: 2024-05-06.
- [ZDM⁺12] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy H. Katz. Detail: reducing the flow completion time tail in datacenter networks. In Lars Eggert, Jörg Ott, Venkata N. Padmanabhan, and George Varghese, editors, *ACM SIGCOMM 2012 Conference, SIGCOMM '12, Helsinki, Finland - August 13 - 17, 2012*, pages 139–150. ACM, 2012.

A. Appendix

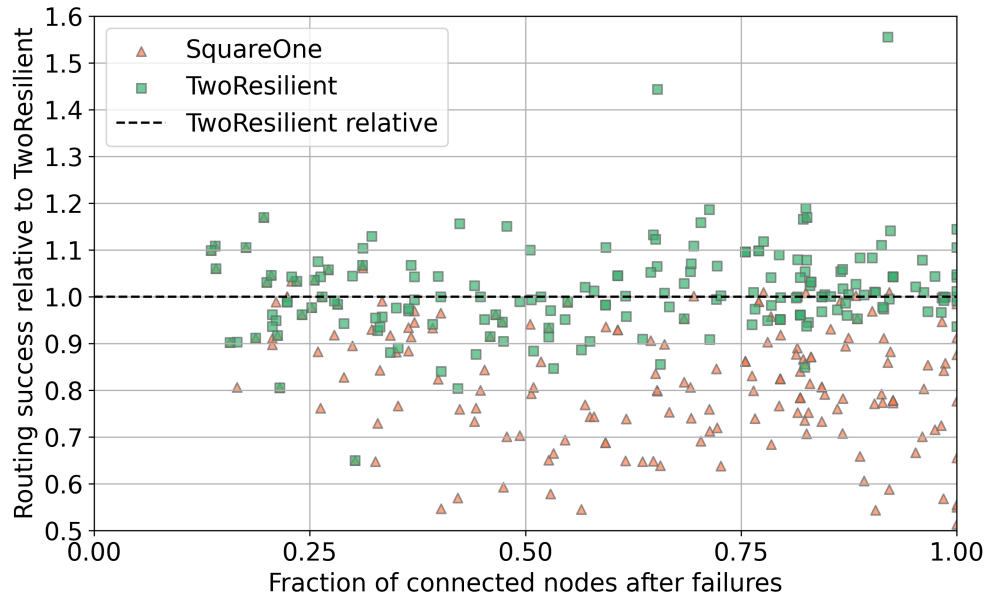


Figure A.1.: Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *Feigenbaum* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. ***TwoResilient*** experiences a dispersion around *Feigenbaum*, however still lies in the vicinity of the line, whereas *SquareOne* lies clearly below.

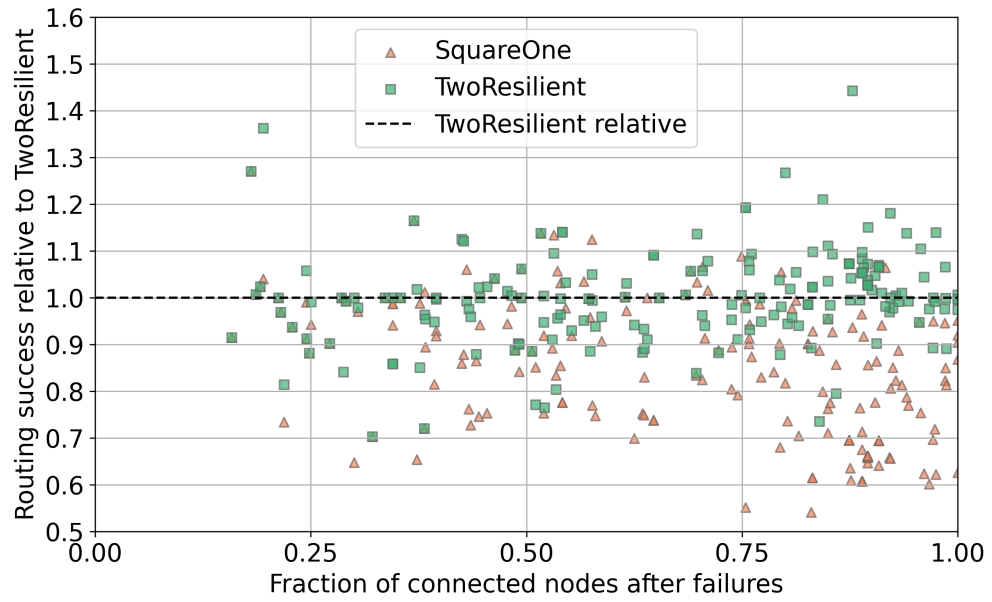


Figure A.2.: Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *Feigenbaum* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 clustered disrupted links, with the analysis conducted across 50 individual trials. **The routing success for *SquareOne*, in some cases, is better than *Feigenbaum*, for the majority however stays below.**

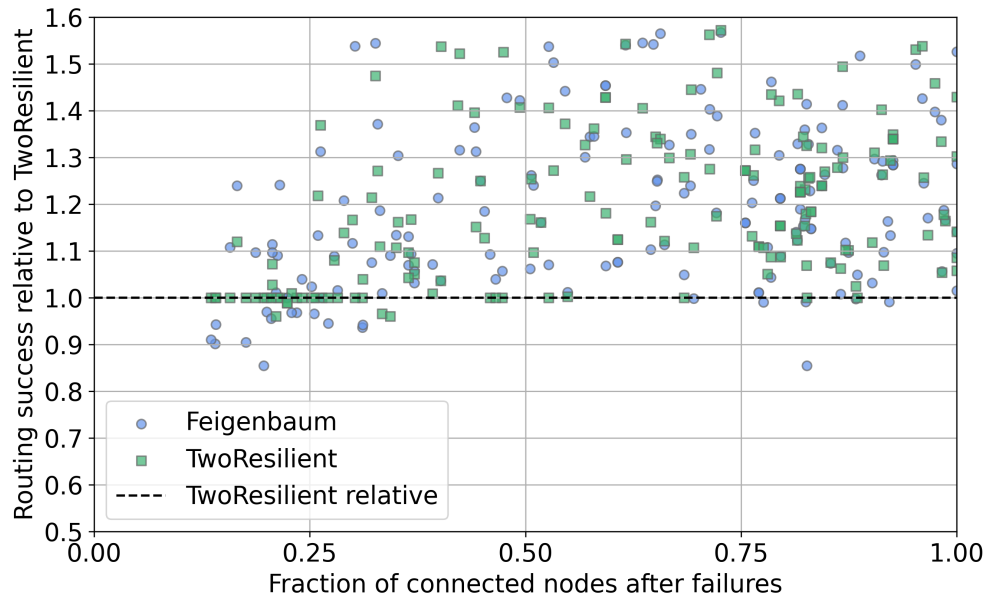


Figure A.3.: Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *SquareOne* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. **In almost all tests both *Feigenbaum* and *TwoResilient* perform better.**

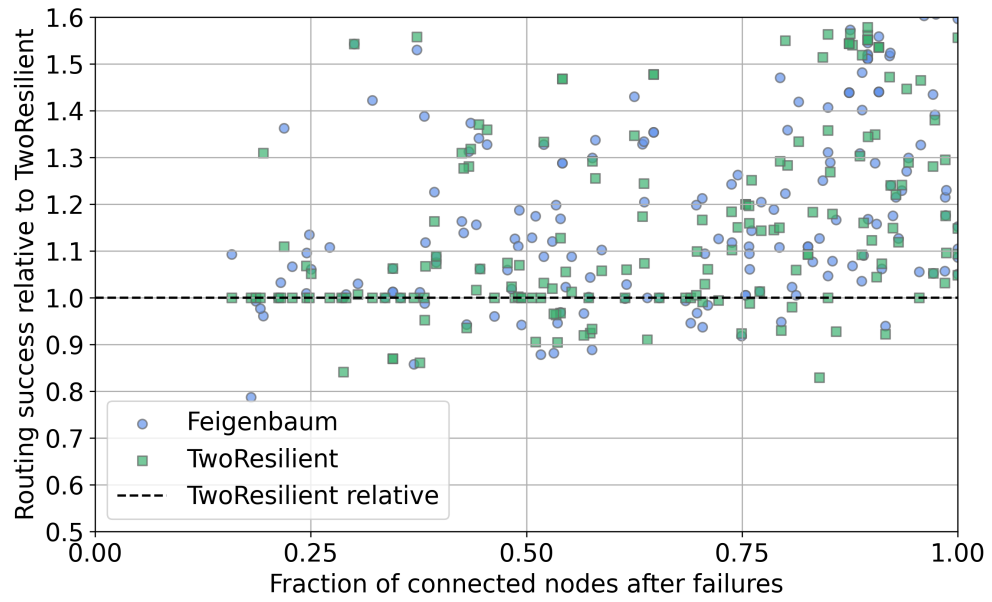


Figure A.4.: Similar to [FKP⁺21], this figure depicts the comparison of the relative average pathfinding effectiveness of all algorithms against the *SquareOne* algorithm on the Zoo topology networks. This comparison is based on the subset of nodes that remain connected to the destination after 10 random disrupted links, with the analysis conducted across 50 individual trials. **In the majority of tests both *Feigenbaum* and *TwoResilient* perform better.**

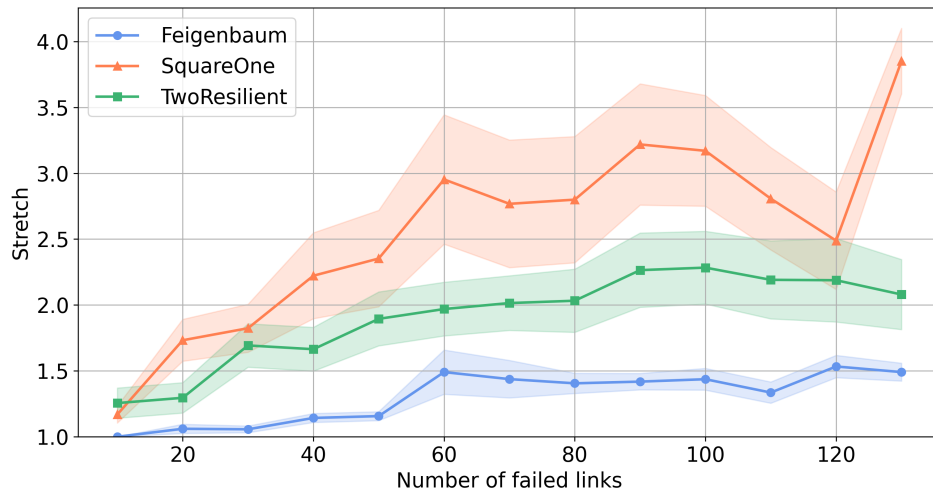


Figure A.5.: This figure depicts the average normalized stretch on the Erdős–Rényi networks for a range of random disrupted links, only for runs where all algorithms have been successful and had a success rate of ≥ 0.5 . **Although, *Feigenbaum* is still taking the shortest detours, *TwoResilient* produces less stretch, when looking at runs with a higher success rate.** The shaded bands represent $\frac{1}{4}$ th of the standard deviation across 10 individual trials carried out for each failure mark.