SIMULAÇÃO E ANÁLISE DE RESILIÊNCIA EM SISTEMAS RECONFIGURÁVEIS

Abstract: The study of system resilience is of great importance today. Within this study, it is highlighted the use of complex network models to analyze the resilience of real systems. In this work, we use this representation to make the resilience analysis of self-healing systems. These systems have the ability to change their topology to recover themselves from a failure. In this work, we analyze the effect of system redundancy on the resilience and how the system topology affects the resilience of self-healing systems. An algorithm that simulates the reconfiguration of the system in case of failures was implemented. The three topologies that were studied are: *Planar Grids*, *Small-world* and *Scale-free*. The analyzes were performed for a variable number of failures, with faults in edges, nodes or both. The resilience investigation was done using the fraction of nodes connected to the system after the system reconfiguration over the total number of nodes in the system. With this data it was possible to infer favorable or unfavorable characteristics for self-healing systems.

Keywords: Resilience, Self-healing Systems, Small-world, Planar Grid, Scale-free

Resumo: O estudo da resiliência de sistemas é de grande importância nos dias atuais. Dentro deste estudo destaca-se a utilização de modelos de redes complexas para se fazer análises da resiliência de sistemas reais. Neste trabalho, utilizou-se esta representação para fazer a análise de resiliência de sistemas reconfiguráveis. Esses sistemas têm a capacidade de alterar a sua topologia para se reabilitar de uma falha. Neste trabalho, foi analisado o efeito da redundância do sistema na resiliência e como a topologia do sistema afeta a resiliência de sistemas reconfiguráveis. Foi implementado um algoritmo que simula a reconfiguração do sistema no caso de falhas. As três topologias que foram estudadas são: *Planar Quadrado*, *Pequeno Mundo* e *Livre de Escala*. As análises foram realizadas para um número variável de falhas, com falhas em arestas, em nós ou em ambos. A averiguação da resiliência foi feita a partir da proporção de nós conectados ao sistema após a reconfiguração do sistema em relação ao número total de nós do sistema. Com esses dados foi possível inferir características favoráveis ou desfavoráveis para sistemas reconfiguráveis.

Palavras-chave: Resiliência, Sistemas Reconfiguráveis, *Pequeno Mundo*, *Planar Quadrado*, *Livre de Escala*

1 Introdução

No mundo de hoje há uma grande necessidade no que se refere ao estudo de sistemas utilizando modelos de redes complexas (Costa et al., 2011). Este fato pode ser observado pelo uso da representação de redes complexas para simular sistemas do mundo real, tais como: relações pessoais (Costa et al., 2011; Barbieri, 2010), internet (Costa et al., 2011), mercado financeiro (Barbieri, 2010), redes de transporte (Cui, Kumara e Albert, 2010), redes de distribuição de eletricidade (Barbieri, 2010), redes proteicas (Costa et al., 2011; Barbieri, 2010), redes genéticas (Barbieri, 2010) entre outras (Barbieri, 2010; Cui, Kumara e Albert, 2010; Costa et al., 2011). Há também um grande esforço em se estudar a resiliência e robustez de sistemas complexos (National Research Council, 2012; Panteli, 2015; Bie et al., 2017) com o objetivo de avaliar a resiliência dos sistema e reduzir o impacto de tragédias (National Research Council, 2012) ou a vulnerabilidade de sistemas elétricos à ataques e falhas (Panteli, 2015; Bie et al., 2017).

Nota-se que existe uma discrepância nas definições para resiliência (Woods, 2015). Alguns trabalhos definem este termo como sendo a capacidade do sistema de se recuperar de falhas. Outros como sendo equivalente a robustez onde se estudam a definição e medida (Hosseini, Barker e Ramirez-Marquez, 2016; Woods, 2015), realizam estudos genéricos de resiliência (Jin, Li e Kang, 2017; Zhang, 2016) e estudos em áreas aplicadas, como a de distribuição de energia elétrica (Panteli, 2015; Bie et al., 2017). Há ainda os autores que definem resiliência como a capacidade de se adaptar a eventos que desafiam os limites do sistema e, por fim, como a capacidade do sistema de continuar a funcionar corretamente mesmo com o crescimento do mesmo ao longo do tempo (Woods, 2015). Dentro dessas definições existe um grande esforço para o estudo da resiliência dos sistemas, utilizando-se da representação por redes complexas como se pode observar pelos trabalhos de Barbieri (2010), Bessani et al. (2017), Wang e Chen (2003), Amaral e Ottino (2004), Quattrociocchi, Caldarelli e Scala (2014) e Ghedini e Ribeiro (2014) .]

Dentre os trabalhos que utilizam-se da representação por redes complexas para estudar a resiliência de sistemas, pode-se ressaltar o estudo dos sistemas reconfiguráveis, os quais têm a capacidade de se reconfigurar para se recuperar de falhas ou para se adaptar a novas situações, como o de Gallos e

Fefferman (2015) e o de Quattrociocchi, Caldarelli e Scala (2014).

Apesar da grande quantidade de trabalhos, ainda existem poucos trabalhos que abordam a resiliência do sistema e o estudo da reconfiguração do sistema (Quattrociocchi, Caldarelli e Scala, 2014). Desta forma, neste trabalho foi feito um estudo da resiliência de sistema reconfiguráveis se utilizando da representação por sistemas complexos. A definição de resiliência utilizada para esta análise foi a de resiliência equivalente a robustez.

Para fazer este estudo, foi desenvolvido um *software* que simula falhas e reconfigurações de sistemas para gerar os dados de análise. Por se tratar de sistemas complexos e em geral grandes, teve-se que explorar abordagens computacionais paralelas a fim de se reduzir significamente os tempos de execução das simulações. Para tal desenvolvimento a linguagem **Julia** foi escolhida por se tratar de uma linguagem nativamente paralela (Bezanson et al., 2017) e que conta com interface de utilização para outras linguagens. Esta plataforma permite a utilização em seus programas, de códigos desenvolvidos em outras linguagens, neste caso a linguagem **Python** e as suas bibliotecas como **NetworkX** e **Matplotlib**. A escolha também se baseou no fato de que a linguagem apresenta desempenho superior ao de **Python** e **Matlab** por vezes chegando a superar o desempenho do **C**.

A seguir serão descritos a Teoria, na Seção 2, onde será descrito os modelos de grafos utilizados e aspectos de paralelização do algoritmo. Nesta seção também será descrita a nomenclatura utilizada no texto. A Seção 3 apresenta os Materiais e Métodos utilizados incluindo uma descrição do ambiente computacional e *softwares* utilizados. Os Resultados obtidos são apresentados na Seção 4 e na Seção 5 haverá a discussão e a conclusão do trabalho.

2 Teoria

Esta seção é para conceituar os principais modelos usados neste trabalho e definir a nomenclatura utilizada.

2.1 Computação Paralela

A computação paralela é um tipo de computação na qual diversos cálculos de processos são executados simultaneamente, de forma que um problema maior pode ser dividido em subtarefas que são executadas ao mesmo tempo e tem os seus resultados combinados ao final da execução (Quinn, 2004).

O potencial de aceleração do algoritmo em uma plataforma de computação paralela é dada pela lei de Amdahl

$$\psi \leq (f + (1-f)/p)^{-1} \quad (1)$$

onde f é a fração de operações que devem ser executadas sequencialmente, $0 \leq f \leq 1$. ψ é a máxima aceleração alcançada para um computador paralelo com p processadores executando o programa (Quinn, 2004).

Para um bom desempenho da computação paralela em um *cluster* o sistema exige uma banda larga e um sistema de interconexão com baixa latência (Quinn, 2004).

2.2 Grafos

Um grafo é um sistema descrito por $G = (V, E)$, onde V é um conjunto não vazio de objetos denominados nós e E é um subconjunto de pares não ordenados de V, chamados arestas. As arestas podem ter direção e peso (Barabasi, 2016).

Para o caso de um grafo pesado cada aresta e está associada com uma capacidade $c(e) > 0$. Além disso, também considera-se que o sistema tem dois nós especiais: fonte s e dreno t, sendo que $s \neq t$. Considerando-se isto, o sistema deve respeitar duas limitações: o fluxo na aresta e não pode ser superior a $c(e)$ e para todo nó $V \neq s, t$ o fluxo que chega ao nó é $i = \{Fluxo\ que\ sai\ do\ nó\ i\}$, logo obtendo-se a equação de equilíbrio de fluxo nos nós

$$\sum_{(k,i)\in E} x_{ki} = \sum_{(i,j)\in E} x_{ij} \quad (2)$$

onde x é o fluxo que passa pela aresta.

Com essa representação é possível modelar qualquer sistema que possa ser representado pela conexão de objetos, onde os nós são os objetos e as arestas indicam e caracterizam as suas conexões.

2.3 Caminho em grafo

Considerando-se um grafo $G = (V, E)$ com dois nós referenciados como fonte s e o dreno t, sendo que $s \neq t$, pode-se obter um caminho entre o dreno e a fonte utilizando-se da aplicação do algoritmo de busca em largura (*BFS*) (Lee, 1961) na fonte e no dreno. Quando as duas buscas se encontram em um nó, sabe-se que um caminho entre a fonte e o dreno foi encontrado.

2.4 Fração de nós conectados

O resultado das simulações é a fração de nós conectados do sistema (**FoS**), que é calculada pela proporção de nós conectados ao sistema após a reconfiguração sobre o total de nós do sistema.

2.5 Planar Quadrado

O grafo *Planar Quadrado* é um grafo planar, ou seja, um grafo que pode ser imerso em um plano de

forma que as suas arestas sejam retas (Netto, 2003). Além disso, o seu desenho forma quadrados.

2.6 Pequeno Mundo

O grafo *Pequeno Mundo* é um grafo onde a distância típica entre dois nós escolhidos aleatoriamente cresce proporcionalmente ao logaritmo do número de nós na rede (Watts e Strogatz, 1998). É uma rede regular com introdução de aleatoriedade (Watts e Strogatz, 1998).

2.7 Livre de Escala

O grafo *Livre de Escala* é um grafo, cuja distribuição de grau segue uma lei de potência, pelo menos assintoticamente. Ou seja, a fração *P(k)* dos nós na rede que tem *k* conexões com outros nós, tende para grandes valores de *k* já que $P(k) \sim k^{-\gamma}$ (Barabasi, Albert e Jeong, 1999).

2.8 Árvore Expandida

A árvore expandida *T* de um grafo *G* é um subgrafo conexo minimal de *G*, logo sendo o menor subconjunto de arestas que forma um subgrafo conexo (Rabuske, 1992).

3 Materiais e Métodos

Neste estudo, empregou-se a linguagem de programação **Julia** 0.6.0 (Bezanson, 2017), juntamente com a biblioteca **NetworkX**, biblioteca da linguagem de programação **Python**, para fazer a simulação dos sistemas representados por redes complexas, utilizando-se de grafos. Para utilizar esta biblioteca em um programa na linguagem **Julia** será utilizada a biblioteca **PyCall**. Além disso, foi utilizado o *cluster* disponível no **LPS**[1] para executar as simulações.

3.1 Softwares Utilizados

A linguagem de programação **Julia** é uma linguagem de programação dinâmica de alto nível para computação numérica que foi anunciada como um projeto de código aberto em Fevereiro de 2012 (Bezanson et al., 2017). Apresenta um compilador *just-in-time* extremamente eficiente que permite um desempenho próximo e às vezes até melhor do que C (Bezanson et al., 2017).

Python é uma linguagem de programação de alto nível, interpretada, de *script*, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte que foi criada em 1991 e conta com uma grande comunidade que oferece suporte e aplicações. Ela foi escolhida por ser amplamente utilizada em aplicações científicas (Oliphant, 2007) e por ser *open-source*, logo não sendo necessário uma licença para a sua utilização.

A biblioteca **NetworkX** é uma biblioteca **Python** para a criação, manipulação e estudo da estrutura, dinâmica e funcionamento de redes complexas (Hagberg, Schult e Swart, 2008). Ela foi escolhida por sua ampla comunidade que oferece documentação e suporte e por ser *open-source*.

A biblioteca **Matplotlib** é uma biblioteca **Python** para criação de gráficos 2D para o desenvolvimento de aplicações, scripts interativos e geração de imagens de qualidade para publicações. Tudo isso, para diversas interfaces de usuário e sistemas operacionais (Hunter et al., 2007). Ela foi escolhida por sua ampla comunidade que oferece documentação e suporte e por ser *open-source*.

PyCall é uma biblioteca da linguagem **Julia** que permite a chamada direta e a interoperação completa com Python (Johnson, 2017). Dessa forma, permitindo a utilização de funções do **Python** em **Julia**, incluindo a importação de bibliotecas, como a **NetworkX**, para um programa em **Julia** e a sua utilização. Logo, a biblioteca **PyCall** será utiliza para importar a biblioteca **NetworkX** e **Matplotlib** para a criação do código.

3.2 Descrição do Cluster/LPS

O *cluster* é um *Beowulf Cluster* (Sterling, 2002) que conta com 13 máquinas, sendo uma máquina servidor e doze máquinas escravas. Todas utilizando **Julia** 0.6.0, **Python** 2.7.13, **Matplotlib** 2.2.0, **Networkx** 2.1, **Kernel** 4.6.6-300.fc24.x86 64 e Ethernet Gigabit. Além disso, todas as máquinas, exceto a *host* e a lps09, usam **Fedora** 24 Server atualizado em 16/08/2016, enquanto a *host* e a lps09 utilizam **Fedora** 24 Workstation atualizado em 17/08/2016 e 16/08/2016, respectivamente. As máquinas do *cluster* contam com processadores e configurações de memória i7-4770 CPU @ 3.40GHz com 8 núcleos e 8 + 8 GiB de RAM com velocidade de 1333MHz. O *cluster* conta com um *switch* com 24 canais ethernet 10 Gbit.

3.3 Descrição da Simulação

Deve-se observar que como foi descrito na introdução, um dos objetivos deste projeto foi a criação de uma função de reconfiguração do sistema, de forma a simular as alterações topológicas do sistema após a ocorrência de falhas. Para isso criou-se um programa que ao receber um sistema em falha, realiza uma busca de caminho entre o nó em falha e a fonte, utilizando como grafo de busca um grafo composto pela união do grafo ativo do sistema com o grafo de *backup*. O grafo de *backup* é gerado considerando-se uma variável *r* que representa a

[1] http://www.sel.eesc.usp.br/lps/

probabilidade de uma aresta, não inclusa no sistema ativo, poder ser utilizada para religar o sistema.

Por fim, para a realização das simulações e testes do projeto, utilizou-se a princípio modelos teóricos de redes, como as redes *Planares Quadradas*, a fim de se fazer uma rápida análise de erros básicos e implementações iniciais. Após essa etapa, foram feitos testes com modelos mais modernos, como as redes de *Pequeno Mundo* e as redes *Livres de Escala*.

Em todos os casos os sistemas foram considerados como sendo redes de distribuição, logo contendo um nó que é a fonte e os outros nós são os consumidores. Exemplos desse tipo de rede são os de distribuição de água, energia e gás. Estes sistemas, para simplificação da simulação, foram considerados com arestas de capacidade infinita e não direcionadas. Ou seja, assumiu-se que a simples conexão de uma aresta a um nó é suficiente para suprir a demanda do nó e que todas as arestas são bidirecionais

As redes *Planares Quadradas* serão criadas utilizando a função grid_2d_graph da biblioteca **NetworkX** (Hagberg, Schult e Swart, 2008), utilizando-se o sistema não periódico. As redes de *Pequeno Mundo* serão criadas utilizando a função watts_strogatz_graph, que utiliza o método de Watts e Strogatz para gerar as redes (Hagberg, Schult e Swart, 2008) com valores de conexão de vizinhos igual a 5 e de reconexão igual a 0,2. Já as redes *Livre de Escala* serão criadas utilizando a função barabasi_albert_graph, que utiliza o método de Barabasi-Albert para gerar as redes (Hagberg, Schult e Swart, 2008) com o valor de conexão igual a 2.

As simulações foram feitas seguindo o procedimento descrito a seguir:

1. Criação da rede por um dos três métodos mencionados. Em seguida, é criada uma árvore expandida que representa uma rota de conexão do grafo utilizando um dos nós passados como sendo a fonte do sistema;
2. Realização da simulação de uma ou mais falhas em aresta ou nó pela retirada do elemento do grafo;
3. Após a simulação da falha, o algoritmo de reconfiguração é acionado, procurando reconectar todo o sistema;
4. Os processos 2 e 3 são repetidos consecutivamente até se completar a simulação de todas as falhas requeridas.
5. Feito isso, o resultado fornecido pela simulação é a proporção de nós conectados ao sistema após a reconfiguração do sistema em relação ao número total de nós do sistema.

O processo também pode ser descrito através do fluxograma da Figura 1.

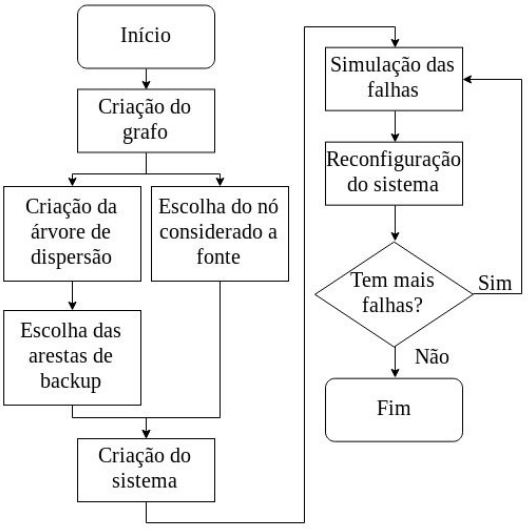

Figura 1. Fluxograma do programa implementado. Nele é descrito o procedimento de execução das simulações. A criação do grafo utiliza funções que geram as topologias requeridas. O sistema é criado pelo agrupamento da árvore expandida, arestas de *backup* e pelo nó fonte. A simulação das falhas é feita pela remoção do objeto do grafo.

As Figuras 2 e 3, a seguir, mostram o método de simulação de falha e reconfiguração do sistema e o formato dos três tipos de sistema em grafos e as árvores expandidas geradas.

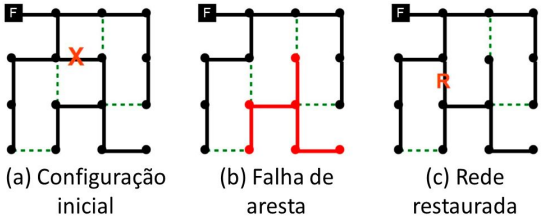

Figura 2. Exemplo de reconfiguração do sistema após falha. Em (a) se vê o sistema original e a simulação de falha na aresta marcada pelo X. Na imagem (b) observa-se o estado do sistema após a falha e marcado em vermelho a parte do sistema que ficou desconectada. Na imagem (c) mostra-se a reconfiguração e restauração do sistema, com a aresta utilizada para reconexão marcada com o R. Figura adaptada de (Quattrociocchi, Caldarelli e Scala, 2014).

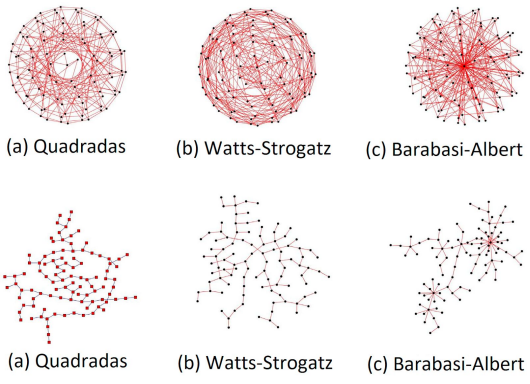

Figura 3. As diferentes topologias dos sistemas. No painel superior são exibidas as três topologias em representações de grafos,

enquanto no painel inferior são mostradas as respectivas árvores expandidas geradas a partir dos sistemas. Figura adaptada de (Quattrociocchi, Caldarelli e Scala, 2014).

4 Resultados

Todos os resultados apresentados foram obtidos para sistemas de 256 nós. Seguem os gráficos dos resultados de duas simulações para se caracterizar o estilo das respostas obtidas, Fig. 4 e Fig. 5. Além disso, para todas as simulações são apresentados os valores de falhas para uma média de nós servidos de 90%, 50% e 10%, o que caracteriza bem a resiliência dos sistemas e a sua taxa de queda.

As Tabelas 1, 2 e 3 mostram os resultados das três topologias de sistemas para os três tipos de falha. Nelas o Tipo se refere a topologia do sistema, Redundância à probabilidade de redundância do sistema. 90%, 50% e 10% se referem ao número de falhas necessárias para atingir essa porcentagem da fração de nós servidos e Dif. se refere ao aumento percentual do números de falhas necessárias para atingir 90% da fração de nós servidos, em relação a probabilidade de redundância anterior. A Tabela 1 mostra os resultados para falhas em arestas, a Tabela 2 para falhas em nós e a Tabela 3 para falhas mistas.

A Fig. 6 representa o tempo computacional de cada simulação.

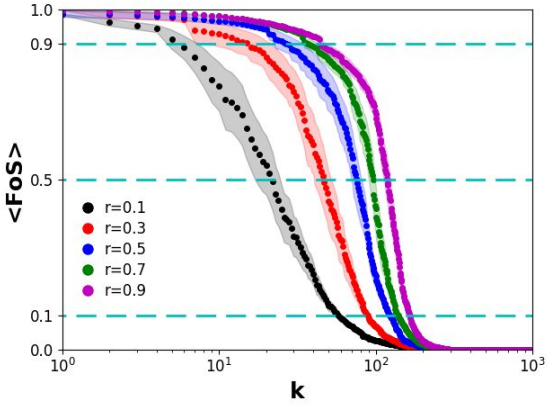

Figura 4. Gráfico da curva média com dispersão dos dados para a simulação de um sistema *Planar Quadrado,* para falhas mistas, sendo k o número de falhas, <FoS> a fração de nós servidos e a variação de cores refere-se a variação da probabilidade de redundância, conforme indicado na legenda.

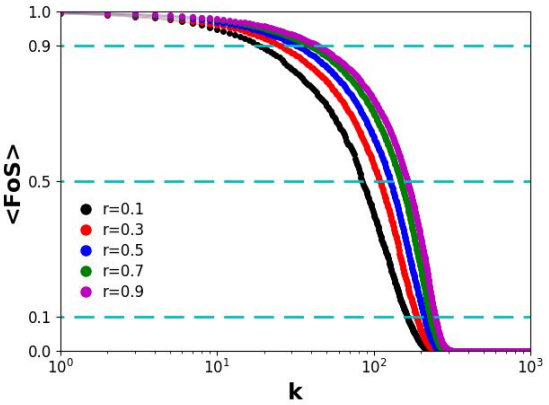

Figura 5. Gráfico da curva média com dispersão dos dados para a simulação de um sistema *Livre de Escala* para falhas mistas, sendo k o número de falhas, <FoS> a fração de nós servidos e a variação de cores refere-se a variação da probabilidade de redundância, conforme indicado na legenda.

Tabela 1. Resultados para falhas em arestas. Tipo se refere a topologia do sistema e Redundância à probabilidade de redundância do sistema. 90%, 50% e 10% se referem ao número de falhas necessárias para atingir essa porcentagem da fração de nós servidos e Dif. se refere ao aumento percentual do número de falhas necessárias para atingir 90% da fração de nós servidos, em relação a probabilidade de redundância anterior.

Tipo	Redundância	90%	50%	10%	Dif. (%)
Planar Quadrado	0,1	9	26	59	-
	0,3	35	68	118	288,9
	0,5	69	108	159	97,1
	0,7	106	154	204	53,6
	0,9	144	198	252	35,8
Pequeno Mundo	0,1	14	39	78	-
	0,3	38	85	125	171,4
	0,5	63	127	176	65,8
	0,7	96	176	226	52,4
	0,9	160	225	277	66,7
Livre de Escala	0,1	27	103	175	-
	0,3	47	162	240	74,1
	0,5	74	214	296	57,4
	0,7	114	266	341	54,1
	0,9	160	314	389	40,4

Tabela 2. Resultados para falhas em nós. Tipo se refere a topologia do sistema e Redundância à probabilidade de redundância do sistema. 90%, 50% e 10% se referem ao número de falhas necessárias para atingir essa porcentagem da fração de nós servidos e Dif. se refere ao aumento percentual do número de falhas necessárias para atingir 90% da fração de nós servidos, em relação a probabilidade de redundância anterior.

Tipo	Redundância	90%	50%	10%	Dif. (%)
Planar Quadrado	0,1	9	28	59	-
	0,3	15	49	80	66,7
	0,5	22	68	101	46,7
	0,7	23	83	116	4,5
	0,9	26	96	128	13,0
Pequeno Mundo	0,1	8	32	66	-
	0,3	16	55	93	100,0
	0,5	23	72	110	43,8
	0,7	25	76	119	8,7
	0,9	26	90	130	4,0
Livre de Escala	0,1	13	68	135	-
	0,3	17	80	144	30,8
	0,5	19	88	155	11,8

	0,7	22	96	164	15,8
	0,9	24	104	172	9,1

Tabela 3. Resultados para falhas mistas. Tipo se refere a topologia do sistema e Redundância à probabilidade de redundância do sistema. 90%, 50% e 10% se referem ao número de falhas necessárias para atingir essa porcentagem da fração de nós servidos e Dif. se refere ao aumento percentual do número de falhas necessárias para atingir 90% da fração de nós servidos, em relação a probabilidade de redundância anterior.

Tipo	Redundância	90%	50%	10%	Dif. (%)
Planar Quadrado	0,1	6	22	58	-
	0,3	16	47	90	166,7
	0,5	27	76	123	68,8
	0,7	38	97	139	40,7
	0,9	45	119	165	18,4
Pequeno Mundo	0,1	6	37	75	-
	0,3	29	72	109	383,3
	0,5	38	97	144	31,0
	0,7	43	116	165	13,2
	0,9	46	136	191	7,0
Livre de Escala	0,1	19	85	164	-
	0,3	25	110	188	31,6
	0,5	33	128	211	32,0
	0,7	39	149	230	18,2
	0,9	44	165	247	12,8

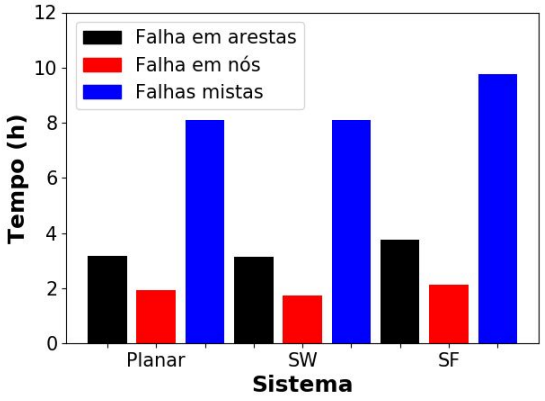

Figura 6. Gráfico do tempo computacional para os sistemas simulados, sendo o eixo x o tipo do sistema, o eixo y o tempo de execução em horas e a variação de cores o tipo da falha da simulação, conforme indicado na legenda.

5 Discussão e Conclusão

Dos resultados obtidos, pode-se observar que em todos as topologias, para todos os tipos de falhas e valores de probabilidade de redundância os resultados tendem a convergir nos extremos, o que mostra que poucas falhas raramente afetam a conexão do sistema, enquanto muitas falhas quase sempre desconectam todo o sistema. Este é um resultado esperado e relativamente óbvio, porém mostra que a simulação não contém nenhum erro grosseiro.

Além disso, pode-se observar, em todos os resultados, que os sistemas que apresentaram uma maior discrepância entre os valores de probabilidade de redundância foram, respectivamente, o *Planar Quadrado*, o *Pequeno Mundo* e o *Livre de Escala*. Sendo que no *Livre de Escala* as diferenças encontradas foram bem pequenas, o que mostra que não há necessidade de se realizar um grande investimento em redundância do sistema. Enquanto para o sistema *Planar Quadrado* e o *Pequeno Mundo* a diferença encontrada entre os resultados fica menor a partir dos 50% de probabilidade de redundância, o que indica que um investimento até esse ponto é relevante.

Apesar da constatação de que os sistemas apresentam pequenas diferenças, a partir de um nível de probabilidade de redundância, os resultados da diferença do número de falhas para atingir 90% dos nós servidos de um sistema com 0,1 para um sistema com 0,3 de probabilidade de redundância tem um valor mínimo geral de 30,8% e um máximo de 383,3% o que mostra que um investimento mínimo em redundância traz grandes benefícios para o sistema.

Considerando as diferenças de resiliência das topologias, pode-se observar que o desempenho do *Planar Quadrado* e do *Pequeno Mundo* foram próximos, apesar de o *Pequeno Mundo* apresentar um desempenho ligeiramente melhor. Já o *Livre de Escala* apresentou um desempenho bem superior aos outros dois no geral, tendo geralmente um índice de 90% da fração dos nós servidos duas vezes maior que o das outras topologias. O que é condizente com os resultados de Quattrociocchi, Caldarelli e Scala (2014).

Considerando-se as diferentes falhas, pode-se observar que os sistemas são mais sensíveis a falhas em nós, o que já era esperado pelo fato de a falha em um nó corresponder a uma falha em todas as arestas ligadas ao nó em falha. As falhas mistas apresentaram um desempenho intermediário, o que também é condizente por ser uma mistura dos dois tipos.

Possíveis melhorias para este estudo são: obter uma maior otimização do código e fazer com que ele consiga trabalhar com grafos direcionados e pesados. Assim, permitindo a análise de resiliência de sistemas reais com ordem de dezenas de milhares de nós.

Agradecimentos

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro, processo nº 2017/21941-9.

Referências Bibliográficas

Amaral, L. A. N. e Ottino, J. M. (2004). Complex networks: Augmenting the framework for the study of complex systems. *The European Physical Journal B*, Vol. 38, pp. 147-162.

Barbieri, A. L. (2010). Análise de Robustez em Redes Complexas. Mestrado. Universidade de São Paulo.

Barabasi A.-L., Albert, R. e Jeong, H. (1999). Mean-field theory for scale-free random networks. *Physica A*, 272, pp. 173–187.

Bessani, M., Massignan, J. A. D., London, J. B. A., Maciel, C. D., Fanucchi, R. Z. e Camillo, M. H. M. (2017). A Hierarchical Framework for Complex Networks Robustness Analysis to Errors. Em: *Systems Conference (SysCon)*, Montreal:IEEE.

Bezanson, J., Edelman, A., Karpinski, S. e Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, Vol. 59 (1), pp. 65-98.

Bie, Z., Lin, Y., Li, G. e Li, F. (2017). Battling the Extreme: A Study on the Power System Resilience. *Proceedings of the IEEE*, Vol. 105 (7), pp. 1253-1266.

Costa, L. da F., Oliveira Jr., O. N., Travieso, G., Rodrigues, F. A., Boas, P. R. V., Antiqueira, L., Viana, M. P. e da Rocha, L. E. C. (2011). Analyzing and Modeling Real-World Phenomena with Complex Networks: A Survey of Applications. Advances in Physics, Vol. 60 (3), pp. 329-412.

Cui, L. Y., Kumara, S. e Albert, Reka. (2010). Complex networks: An engineering view. *IEEE Circuits and Systems Magazine*, Vol. 10 (3), pp. 10-25.

Gallos, L. K. e Fefferman, N. H. (2015). Simple and efficient self-healing strategy for damaged complex networks. *Physical Review E*, Vol. 92 (052806), pp. 1-9.

Ghedini, C. G. e Ribeiro, C. H. C. (2014). Improving Resilience of Complex Networks Facing Attacks and Failures Through Adaptive Mechanisms. *Advances in Complex Systems*, Vol. 17 (2), pp. 1-25.

Hagberg, A. A., Schult, D. A. e Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. Em: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena:Eds, pp. 11-16.

Hosseini, S., Barker, K. e Ramirez-Marquez, J. E. (2016). A review of definitions and measures of system resilience. *Reliability Engineering and System Safety*, Vol. 145, pp. 47-61.

Jin, C., Li, R. e Kang, R. (2017). Maximum flow-based resilience analysis: From component to system. *PLoS ONE*, Vol. 12 (5), pp. 1-15. Disponível em: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0177668 [Acesso em 31 Mar. 2018].

Johnson, S. G. (2017). Calling Python functions from the Julia language. [online] PyCall.jl. Disponível em: https://github.com/JuliaPy/PyCall.jl [Acesso em 22 Set. 2017].

Lee, C. Y. (1961). An Algorithm for Path Connections and Its Applications. *IRE Transactions on Electronic Computers*, Vol. EC-10 (3), pp. 346-365.

National Research Council. (2012). *National Research Council Disaster Resilience: A National Imperative*. Washington: The National Academies Press, pp. 1-260.

Netto, P. O. B. (2003). *Grafos: Teoria, Modelos, Algoritmos*. 3ª ed. São Paulo: E. Blücher.

Oliphant, T. E. (2007). Python for Scientific Computing. *Computing in Science & Engineering*, vol. 9 (3), pp. 10-20.

Panteli, M. e Mancarella, P. (2015). The Grid Stronger, Bigger, Smarter Presenting a Conceptual Framework of Power System Resilience. *IEEE Power and Energy Magazine*, Vol. 13 (3), pp. 58 - 66.

Quattrociocchi, W., Caldarelli, G. e Scala, A. (2014). Self-Healing Networks: Redundancy and Structure. *PLoS ONE*, [online] Vol. 9 (2), pp. 1-7. Disponível em: https://doi.org/10.1371/journal.pone.0087986 [Acesso em 31 Mar. 2018].

Quinn, M. J. (2004). *Parallel programming in C with MPI and OpenMP*. Nova York: McGraw-Hill.

Rabuske, M. A. (1992). *Introdução à teoria dos grafos*. Florianópolis: Ed. da UFSC.

Sterling, T. L. (2002). *Beowulf cluster computing with Linux*. Cambridge: MIT press.

Strogatz, S. H. (2001). Exploring Complex Networks. *Nature*, Vol. 410, pp. 268-276.

Wang, X. F. e Chen, G. (2003). Complex Networks: Small-World, Scale-Free and Beyond. *IEEE Circuits and Systems Magazine*, Vol. 3 (1), pp. 6-20.

Woods, D. D. (2015). Four concepts for resilience and the implications for the future of resilience engineering. *Reliability Engineering & System Safety*, Vol. 141, pp. 5-9.

Watts, D. J. e Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, Vol. 393, pp. 440-442.

Zhang, Y. A Comprehensive Analysis Method for System Resilience Considering Plasticity. (2016). Em: *Industrial Engineering, Management Science and Application (ICIMSA)*. Jeju: IEEE.