Aceleradores com CGRAs para Redes Reguladoras de Genes*

Olavo Barros[1], Caio Von Rondow [1], Jeronimo Penha[1,3], Michael Canesche[2], José Augusto M. Nacif[1], Ricardo Ferreira[1]

[1]Universidade Federal de Viçosa (UFV)
Avenida Peter Henry Rolfs – 36570-900 – Viçosa – MG – Brasil
[2] Dept. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
[3] Centro Federal de Formação Tecnológica de Minas Gerais (CEFET-MG)
email: {olavo.barros,caio.rondow,jnacif,ricardo}@ufv.br

Resumo. *As redes reguladoras de genes são modelos usados para estudar a evolução e o comportamento celular. Uma rede pode ser representada por um grafo Booleano. Os algoritmos podem explorar paralelismo com a implementação dos grafos em FPGAs, porém devido a alta flexibilidade, o mapeamento para o FPGA pode demorar horas. Uma solução é usar um CGRA específico, que pode reduzir o tempo de mapeamento para poucos segundos, entretanto, esta abordagem gera novos desafios. Primeiro, as redes são grafos livres de escala. Segundo, as métricas de custo são diferentes dos CGRAs usuais. Este trabalho aplica a técnica de mapeamento de Simulated Annealing com novas funções de custo considerando especificidades das redes. Três padrões de interconexão foram avaliados para um conjunto de 21 redes reguladoras da literatura.*

1. Introdução

As redes reguladoras de genes auxiliam na compreensão do funcionamento celular, no desenvolvimento de pesquisas de novos medicamentos e evolução de doenças como o câncer. Entre os diversos modelos matemáticos para redes, pode-se destacar o modelo de grafo Booleano [Aldana 2003]. Os vértices representam os genes e as arestas implementam as iterações. Entretanto, os algoritmos que estudam a dinâmica das redes têm um custo exponencial, sendo fundamental o uso da computação de alto desempenho.

Uma solução é o uso de *hardware* reconfigurável como os FPGAs [Manica 2019]. Um FPGA é um circuito que pode ser configurado a nível de *hardware* para se adaptar ao problema. As redes podem ser diretamente mapeadas em *hardware* e as interações podem ser avaliadas em paralelo. Os FPGAs têm a vantagem da granularidade a nível de bit, que pode ser explorada pelas implementações das redes Booleanas. Entretanto, o uso de FPGAs ainda é um desafio para a maioria dos programadores que não possuem conhecimentos aprofundados de *hardware*. Outro desafio é o tempo de compilação que pode consumir horas, desestimulando os desenvolvedores. Uma solução é uma arquitetura pré-mapeada que pode ser configurada após a gravação do FPGA. Este trabalho propõe um *Coarse Grain Reconfigurable Architecture* (CGRA) específico para redes reguladoras com estrutura de *malha*. O CGRA é um arranjo de elementos de processamento (EP). O

*Agradecimentos: FAPEMIG (PIBIC, APQ–01203-18), CNPq (PIBIC, 440.087/2020–1, 236.290/2018–9), NVIDIA, Xilinx, Funarbe, Laboratório Nacional de Computação Científica (GenRegAcc). O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Código de Financiamento 001.

CGRA proposto tem a granularidade dos EP ajustada para o problema das redes. Cada EP irá implementar a função de um gene. As ligações entre os EPs irão fazer a comunicação para a evolução da dinâmica da rede [Bragança et al. 2021].

O *Simulated Annealing* (SA) é um dos algoritmos mais eficiente para mapeamento de grafos em estrutura de *malha* [Murray et al. 2020]. Entretanto, para grafos de circuitos mapeados em FPGAs e para os grafos de códigos mapeados em CGRA tradicionais, as propriedades e métricas são diferentes dos grafos de redes reguladoras. Nos FPGAs, o mais importante é reduzir o custo total de fios [Murray et al. 2020]. No CGRA para grafos de código, além da redução de distâncias é importante buscar soluções com caminhos equilibrados [Canesche et al. 2020]. A operação básica para as redes reguladoras é a atualização de todos os vértices ao mesmo tempo. Diferentes dos CGRAs e FPGAs tradicionais, a função objetivo busca minimizar a maior distância e não o comprimento total dos fios. Além disso, os grafos das redes são mais densos e nem sempre é possível a conectividade em um único passo. Portanto, a comunicação será decomposta em vários passos, reduzindo o desempenho.

Este trabalho propõe novas funções de custo para o SA no problema de mapeamento das redes reguladoras. Além disso, três arquiteturas são avaliadas com redes reguladoras de problemas reais da literatura [UFV 2022], em contraste com outros trabalhos de FPGAs, que usam redes sintéticas [Aldana 2003, Ferreira and Vendramini 2010].

Este artigo apresenta uma solução heurística para um problema NP-completo, explorando uma implementação de alto desempenho para solucionar de forma eficiente o problema. O artigo está estruturado da seguinte forma. A seção 2 apresenta as redes reguladoras. A seção 3 discute o mapeamento das redes, suas diferenças e as novas funções de custo. A seção 4 apresenta os experimentos para a validação das novas funções. A seção 5 compara este trabalho com o estado da arte para soluções em FPGA para redes reguladoras. Finalmente, a seção 6 destaca os pontos principais e direções futuras.

2. Redes Complexas e Redes Reguladoras

As redes reguladoras de genes podem ser classificadas como redes livres de escala [Aldana 2003], pela teoria de redes complexas [Barabási 2009]. Apesar do grau médio dos vértices da rede variar entre 2 e 4 arestas, a rede pode ter um ou dois vértices com grau 10, ou maior. Estes vértices são chamados de *hubs* da rede. A Figura 1(a) ilustra o grafo da rede reguladora livre de escala, no qual pode-se observar a presença de *hubs*. A Figura 1(b) mostra um histograma do grau de distribuição dos nós, considerando o grau de entrada. Podemos observar que, o grau médio é baixo, sendo igual a 2,25, porém têm-se dois vértices com graus de entrada 5 e outros dois vértices com graus de entrada 8. Ou seja, ao mapear esta rede em uma *malha* simples com 4 vizinhos (norte, sul, leste, oeste) serão necessários no mínimo dois ciclos para atualizar estes 4 vértices com 5 e 8 vértices adjacentes. No primeiro ciclo são atualizados 4 vizinhos diretos e no segundo ciclo poderão ser atualizados os vértices adjacentes restantes, que serão vizinhos indiretos.

No modelo booleano [Aldana 2003], a dinâmica de cada vértice da rede reguladora é gerida por uma equação booleana. A equação representa a função de ativação do vértice e depende dos vértices adjacentes que estão diretamente conectados. A dinâmica da rede é controlada pela atualização de todos os vértices ao mesmo tempo, no mo-

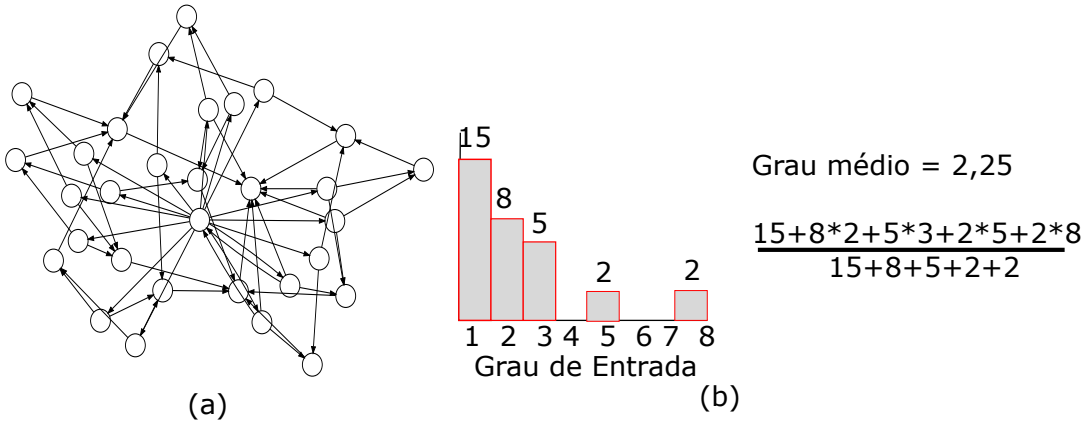

Figura 1. (a) Rede Reguladora; (b) Histograma e Grau Médio

delo síncrono. Este processo pode ser realizado de forma eficiente em um FPGA, onde todas as funções são atualizadas em paralelo. As implementações apresentadas em [da Silva et al. 2017] e [Manica 2019] mapeiam a rede reguladora no FPGA, gerando ganhos de desempenho de 10 a 1.000 vezes em relação a uma implementação em CPU. Apesar do alto desempenho, caso seja necessário modificar a rede, o processo de compilação para o FPGA deve ser repetido, o que pode demorar horas. Uma alternativa é propor um modelo que possa ser atualizado dinamicamente.

[Ferreira and Vendramini 2010] apresentam uma arquitetura dinâmica para mapear uma rede reguladora em FPGA. A arquitetura utiliza uma rede multiestágio cujo o custo é $O(n \log n)$, sendo n o número de nós. Apesar da rede multiestágio reduzir o custo de comunicação em comparação a uma rede *crossbar* que é $O(n^2)$, o custo ainda inviabiliza a replicação do acelerador para ter várias simulações simultâneas dentro de um FPGA. Este trabalho propõe arquiteturas com *malhas* de duas dimensões que tem um custo de conexão $O(1)$, pois cada elemento de processamento (EP) irá se conectar apenas com os vizinhos diretos. Três arquiteturas são avaliadas: *malha* com grau 4, a *chess* com grau médio 6 e a *1-hop* com grau 8. Os desafios são: (a) determinar qual é o grau de conectividade que a arquitetura deve ter, (b) como mapear redes livres de escala de forma eficiente; (c) quantos ciclos serão necessários para atualizar os valores dos vértices da rede. No caso malha considera-se que um vértice pode-se comunicar com 8 ou mais vértices adjacentes, mas só possui conexão direta com 4 vizinhos. Portanto, será necessário repassar a informação através da malha, que irá consumir 2 ou mais ciclos.

3. Mapeamento Dinâmico em Arquiteturas Reconfiguráveis

O problema de mapeamento de um grafo em uma *malha* é NP-completo. No caso de uma Rede Reguladora que tem um grau variado de entrada, nem sempre é possível posicionar todos os vértices do grafo com conexões diretas aos seus adjacentes, mesmo em redes com grau médio baixo, como já foi apresentado na Figura 1.

Em geral, o *Simulated Annealing* (SA) tem como função de custo a minimização do comprimento total de fios. Figura 2 apresenta um exemplo simples de uma arquitetura em linha com apenas dois EPs vizinhos. Os EPs do meio tem dois vizinhos diretos (norte e sul) e os EPs das extremidades apenas 1 vizinho. A Figura 2(a) apresenta o mapeamento de um grafo com apenas três arestas: $a \rightarrow d$, $a \rightarrow b$ e $b \rightarrow c$ na arquitetura em linha com 4

células ou EPs. As arestas têm custo 3, 1 e 1, respectivamente. O custo igual a 3 significa que são necessárias três unidades de tempo para propagar o valor do vértice a para o vértice d, pois a está fisicamente mapeado em um EP com distância 3 da célula que d está posicionado, conforme ilustra a Figura 2(a). O valor de a é transmitido para b no primeiro ciclo, depois de b para c no segundo ciclo e apenas no terceiro ciclo repassa de c para d. O comprimento total de fios é 5. A Figura 2(b) ilustra outra solução de posicionamento em que o custo total também é 5. Para o mapeamento de redes reguladoras, a segunda solução é melhor que a primeira, pois o pior caso é uma aresta de comprimento 2. Ou seja, em duas unidades de tempo (2 ciclos) todos os vértices estão atualizados na solução da Figura 2(b), enquanto a solução da Figura 2(a) serão necessários 3 ciclos. Neste exemplo, existe uma solução ótima ilustrada na Figura 2(c), com um custo total de 3 e maior distância de 1. Porém devido à distribuição irregular dos graus de vértices nas redes livres de escala, nem sempre é possível achar uma solução ótima. Ademais, o número de possibilidades cresce exponencialmente.

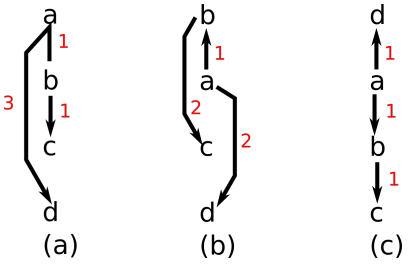

Figura 2. Três Posicionamentos.

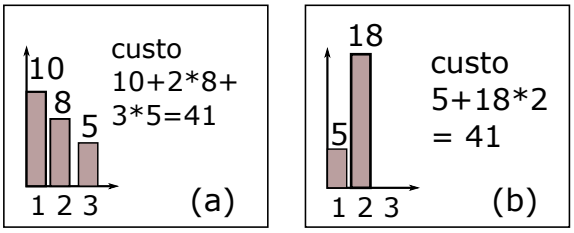

Figura 3. (a) Histograma com 23 arestas; (b) Redução do pior caso.

O algoritmo SA resolve bem o posicionamento, porém como a métrica é minimizar o pior caso, ao contrário de outras implementações, é necessário buscar melhores alternativas de função de custo. A Figura 3(a) mostra um histograma de posicionamento com 23 vértices mapeados; 10 ligações com distância 1; 8 vértices com distância 2; e 5 vértices com distância 3, totalizando um comprimento total de fios com o custo de 41. Esta solução será gerada pelo SA com uma função de custo para otimizar o comprimento total dos fios. A Figura 3(b) mostra um deslocamento no histograma da solução se o posicionamento usar um número menor de ligações de distância 1, aumentar o número de mapeamentos com distância 2 e eliminar qualquer mapeamento de distância 3. Apesar do custo total ser 41 ligações, a segunda opção irá executar em 2 ciclos ao invés de 3, pois a maior distância foi reduzida de 3 para 2, sendo 1,5$\times$ mais rápida que a primeira.

3.1. Função de Custo para Pior Caso

A função custo usual do SA é $custo = \sum_i^N L_i$, onde L_i é o custo da distância da i-ésima ligação. Neste trabalho, essa função é denominada como custo linear e será utilizada como métrica de comparação com as outras soluções. Uma primeira abordagem seria aumentar o custo das arestas de maior distância. Desse modo, avaliamos três funções: quadrática, exponencial e limiar para aplicar uma penalização mais eficaz e assim evidenciar o pior custo ao invés do custo total. A função quadrática ou polinomial tem o custo avaliado pela equação $\sum_i^N L_i^2$. Para o exemplo da Figura 2(d), o custo é $10 * 1^2 + 8 * 2^2 + 5 * 3^2 = 87$ e para Figura 2(e) é $5 * 1^2 + 18 * 2^2 = 77$. A função exponencial tem o custo avaliado pela equação $\sum_i^N 2^{L_i - 1}$. Para o exemplo da Figura 2(d),

o custo é $10 * 2^0 + 8 * 2^1 + 5 * 2^2 = 46$ e para Figura 2(e) é $5 * 2^0 + 18 * 2^2 = 41$. Em relação à função limiar, o valor do custo é L_i, para valores abaixo do limiar. Se o custo da aresta for acima do limiar, o custo é o número de arestas multiplicado pelo comprimento da aresta. Então o custo é $\sum_1^N F(L_i)$ onde F é:

$$F(L_i) = \begin{cases} L_i & \text{se } L_i \leq Limiar; \\ L_i * Total\ de\ Arestas & \text{se } L_i > Limiar. \end{cases}$$

Supondo um limiar de 2 para a maior distância, para o exemplo da Figura 2(d), o custo seria $10 * 1 + 8 * 2 + 5 * 3 * 23 = 371$. Para as 18 arestas com comprimento 1 e 2, o custo é multiplicado por L_i. Para as 5 arestas de comprimento 3, que estão acima do limiar, o custo é multiplicado por 23, que é o total de arestas. Para Figura 2(e) o custo é apenas de $5 * 1 + 18 * 2 = 41$, pois não há nenhuma aresta acima do limiar.

3.2. Arquiteturas

O custo da arquitetura irá crescer com o número de conexões. Por exemplo, em uma *malha*, cada nó tem 4 vizinhos, enquanto que em uma arquitetura *1-hop*, cada nó tem 8 vizinhos. Para as redes avaliadas, o grau médio de conectividade varia de 1,2 a 4,1. Porém, mesmo um grafo com grau médio 2 pode-se ter um vértice com grau 8 (*hub*) que não pode ser mapeado em uma ciclo numa malha com 4 vizinhos diretos.

A Figura 4 apresenta as três arquiteturas avaliadas. A Figura 4(a) ilustra a arquitetura em *malha* que conecta os 4 vizinhos (norte, sul, leste, oeste), com destaque para a célula central marcada com o rótulo 0. Os rótulos das outras células mostram a distância da célula central. Observe que, uma célula tem 4 vizinhos com distância 1. Já com distância 2, tem-se 8 vizinhos e com distância 3 tem-se 12. Ou seja, o número de vizinhos com distância i é $4i$. Isto significa que, se um vértice tem 7 vértices adjacentes, apenas 4 serão mapeados nos vizinhos diretos. Os outros três serão mapeados em EPs com distância 2 e será necessário 2 ciclos para enviar a informação deles para o vértice alvo. Primeiro, o EP de distância 2 envia para um EP de distância 1, que no ciclo seguinte repassa para o EP de destino. Esta informação é importante para estimar o melhor caso para a maior distância em uma arquitetura em *malha*. Suponha que, a rede reguladora possua um vértice com o grau de conexão 13. Em um caso ótimo no posicionamento, a distância ótima deve ser no mínimo 3, pois só existem 4 células com distância 1 e 8 células com a distância 2, totalizando 12 células e, portanto, será necessária uma célula com distância 3. Entretanto, a distância mínima será de 4 ciclos, pois na etapa seguinte de roteamento, cada EP só recebe 4 comunicações por ciclo. Portanto, para receber 13 comunicações, o EP receberá 4, depois mais 4 no próximo ciclo, mais 4 no ciclo seguinte e, por fim, mais uma, totalizando 4 ciclos.

A Figura 4(b) apresenta a arquitetura *1-hop* com destaque para a célula central. Todas as células têm um padrão de 8 ligações como ilustrado para célula central. A célula é ligada nas duas células em cada direção (norte, sul, leste e oeste), com a conexão direta e com um salto (*1-hop*). Os rótulos da Figura 4(b) mostram as distâncias da célula central. Note que, a arquitetura *1-hop* reduz, significativamente, a distância e todas as células no exemplo de arquitetura 5×5 têm distância igual ou menor que 2.

Entretanto, a arquitetura *1-hop* tem o custo de até 8 conexões por célula (exceto das bordas e quinas). Uma maneira de manter as vantagens da *1-hop* e reduzir o número

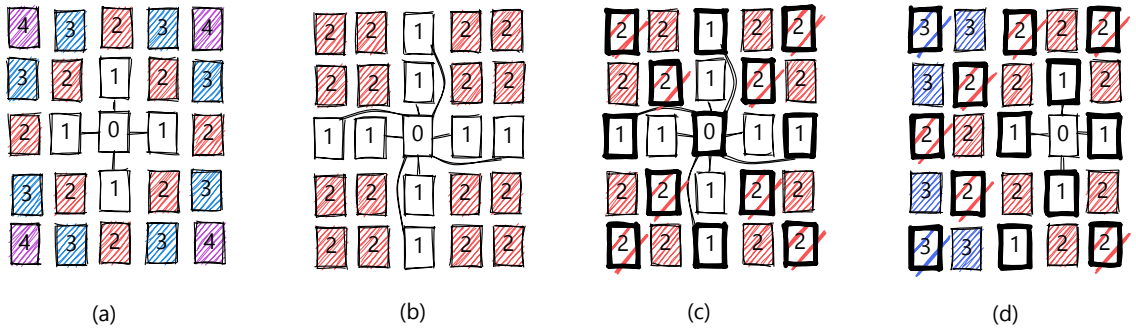

Figura 4. (a) Malha; (b) 1-hop; (c) Chess com destaque para nó 1-hop; (d) Chess com destaque para nó malha.

total de conexões é a arquitetura *chess* proposta em [Carvalho et al. 2020], que mescla a arquitetura *1-hop* com a arquitetura *malha* como um tabuleiro de xadrez. As células em negrito nas Figuras 4(c) e (d) tem o padrão de conexão *1-hop* com 8 vizinhos. As células sem negrito tem o padrão *malha* com 4 vizinhos. A Figura 4(c) destaca uma célula central *1-hop*. Pode-se observar que, as distâncias da célula central para este exemplo da *chess* são as mesmas da *1-hop*. Porém, depende do ponto de partida (célula *1-hop* ou *malha*). A Figura 4(d) destaca as distâncias da célula *malha* à direita da célula central. Observa-se 4 vizinhos com distância 1. Com distância 2 tem-se 14 células em uma arquitetura 5×5, mas teria 20 células com distância 2 se fosse uma arquitetura 7×7.

Outra contribuição deste trabalho é avaliar geometrias retangulares, além do quadrado mínimo. Suponha que a rede tenha 35 vértices. Pode-se propor uma arquitetura 6×6 que é o quadrado mínimo para 35 vértices. Mas, poderíamos propor uma arquitetura retangular com 5×7, que tem 35 nós ou uma arquitetura 6×7 com 42 nós. Em geral, quando mais próximo for o tamanho da arquitetura do tamanho do grafo, o posicionamento será mais compacto. Outra vantagem de explorar geometrias retangulares é que, para redes com 50 a 200 vértices (tamanho típico das redes reguladoras), teremos mais opções. Enquanto o quadrado mínimo está restrito à sequência 64, 81, 100, 121, 144, 169, 196 e 225 células, se considerarmos os retângulos, temos mais opções como $64, 70, 72, 80, 81, \ldots, 225$.

4. Resultados

Esta seção está estruturada da seguinte forma. A seção 4.1 apresenta as redes reguladoras utilizadas e suas propriedades. A seção 4.2 avalia as novas funções de custo nas 3 arquiteturas.

4.1. Redes Reguladoras

A Tabela 1 apresenta 21 redes reguladoras selecionadas por possuírem métricas e propriedades diferentes umas das outras como, por exemplo, diferenças em número de nós, arestas e graus máximos e mínimos. Dessa forma, tem-se um conjunto diversificado e representativo. A Tabela 1 apresenta um identificador e o nome de cada rede. Pode-se notar que modelam problemas de comportamento celular e doenças. Por questões de espaço, mais informações podem ser obtidas em [UFV 2022]. As colunas seguintes da Tabela 1 apresentam o número de arestas e vértices além de um histograma resumido da

distribuição de graus dos vértices. As colunas 1-3 mostram quantos vértices têm grau entre 1 e 3. Considerando a média, tem-se que 53,3% dos vértices estão nesta categoria. As colunas 4-7 e 8+ mostram a quantidade de vértices com grau entre 4 e 7, 8 ou mais, respectivamente. Na segunda categoria dispõe 34,6% e os vértices com grau 8 estão mais presentes nas redes mais densas. A distribuição dos graus é importante para entender-se a qualidade dos resultados. Depois são apresentados o grau máximo e médio. Se considerarmos que o maior *hub* do grafo irá determinar o limite inferior, apenas as redes 0 e 2 (em negrito na Tabela 1) têm a solução ótima. A Tabela 1 também mostra uma primeira avaliação dos resultados do posicionamento considerando a arquitetura em *malha*. Neste trabalho avalia-se a melhor geometria considerando o quadrado mínimo e as várias opções de dimensões retangulares com tamanho próximo do grafo. A coluna *Dim* mostra as dimensões em linhas e colunas da *malha*, por exemplo, a rede 1 tem tamanho 6x8.

A última coluna mostra a maior distância para o melhor mapeamento. O mapeamento inclui o posicionamento e o roteamento. Pode-se observar que a rede 1, cujo maior grau de vértice é 8, poderia ser mapeada com distância 2 em uma *malha*, já que é possível alcançar 4 vizinhos com um ciclo e mais 4 com dois ciclos, totalizando 8. Porém, o histograma da rede 1 apresenta 16 vértices com grau entre 4 e 7. Portanto, a rede 1 é mais densa que as redes 0 e 2 que foram mapeadas de forma ótima. Assim, o melhor mapeamento para a rede 1 necessita de 3 ciclos. Outro ponto importante é avaliar a escalabilidade da solução. Para as maiores redes que estão nas últimas linhas da tabela, o nível de dificuldade aumenta. Por exemplo, para a rede 16, considerando-se apenas o maior grau 22, poderia-se ter uma solução com custo 4, porém a solução encontrada tem maior distância igual a 6. Pode-se observar que o grau médio é 2,1, que é maior que as médias das redes e além disso temos 12 vértices com grau maior que 8 e mais da metade dos vértices têm grau entre 4 e 7. Finalmente, pode-se observar que as redes 17 e 19 não tiveram roteamento válido na *malha*. Estas redes têm grau médio 3,5 e 4,1 que é muito próximo e até maior que o grau médio de conexão da *malha* que é 4. Este fato justifica a escassez de recursos de roteamento.

Além do algoritmo de posicionamento usando *Simulated Annealing* é necessário realizar a etapa de roteamento para verificar ou legalizar se a solução tem recursos de conexão para ser efetivada. Como a função de custo no posicionamento não legaliza o roteamento, uma alternativa é executar várias instâncias de modo a garantir que teremos pelo menos uma solução válida. Para cada rede foram realizados N posicionamentos, cada um começando de uma solução inicial aleatória. O valor de N foi fixado em 1000, baseado nos experimentos realizados em [Carvalho et al. 2020]. Para todos os testes foram usados dois algoritmos de roteamento. O primeiro recorre a uma estratégia gulosa. Cada aresta é roteada buscando a distância mínima, alocando em cada passo a primeira conexão livre. A segunda abordagem aplica o algoritmo *maze route* [Lee 1961]. Essa abordagem permite legalizar mais soluções de posicionamento que o algoritmo guloso, porém pode aumentar a distância mínima. Ademais, os vértices das redes foram roteados em uma ordem decrescente da distância, ou seja, os vértices com a maior distância foram roteados primeiro.

4.2. Avaliação do Mapeamento e Funções de Custo

Para cada uma das redes trabalhadas, foram realizadas 1.000 execuções do algoritmo *Simulated Annealing* nas arquiteturas de *malha*, *1-hop* e *chess*, considerando-se

Tabela 1. Caracterização das Redes Reguladoras.

N	Nome	V/A	Grau de Total			G_{max}	$\overline{G}$	Dim.	Custo
			1-3	4-7	8+				
0	**Cholesterol Regulatory Pathway**	34/43	32	2	0	6	1.2	6x6	2
1	Apoptosis network	43/73	26	16	1	8	1.7	6x8	3
2	**Castration-Resistant Prostate**	43/65	35	7	1	15	1.5	7x7	3
3	Guard Cell Abscisic	48/78	27	19	2	11	1.7	7x7	3
4	B bronch. and T retort.	63/146	20	28	6	11	2.4	9x6	3
5	T lymphocytes	59/97	34	20	5	8	1.7	7x8	3
6	Cancer Cell Network	61/103	30	26	5	11	1.8	6x10	3
7	PC12 Cell Differentiation	65/109	49	13	3	20	1.7	7x10	4
8	T-LGL Survival Network 2011	66/195	15	42	9	53	3.1	7x9	6
9	Bortezomib Human Myelo	72/130	37	30	5	12	1.9	7x11	4
10	Signaling in Keratinocytes	74/103	46	25	3	12	1.5	7x11	3
11	Glucose Repression	85/106	57	23	5	12	1.4	9x9	3
12	Yeast Apoptosis	86/112	59	23	4	16	1.5	7x12	4
13	chronic lymphocytic leukemia	91/150	74	11	6	21	1.3	8x12	4
14	Lymphopoiesis Network	95/158	56	32	7	11	1.8	7x12	4
15	IL-6 Signalling	103/149	60	36	7	16	1.7	10x9	4
16	EGFR & ErbB Signaling	104/255	57	63	12	22	2.1	8x14	5
17	transduction in fibroblasts	148/546	27	70	51	35	3.5	11x13	11
18	CD4 T cell signaling	188/380	111	94	17	15	1.8	15x13	5
19	Erb-reception	247/1114	59	112	95	46	4.1	12x21	-
20	macrophave activation	321/540	245	59	17	83	1.6	19x17	8

todas quatro funções de custo. O valor de limite usado na função limiar foi baseado nos resultados da função exponencial após o roteamento. Para os casos em que todas as soluções propostas não foram legalizadas pelo roteamento guloso, a estratégia de *maze route* foi usada, já que ela gera mais chances de validação, mas pode aumentar a distância máxima ao contornar células.

A Tabela 2 apresenta todos os resultados para *malha*. A primeira coluna da tabela (N) é referente às redes trabalhadas, como apresentado na Tabela 1. A próxima coluna *V/A,* G_{max} *e M* contém as informações de vértices e arestas, o maior grau de um nó G_{max} e a distância mínima M necessária para mapear o vértice de maior grau. As quatro colunas seguintes têm a maior distância Antes $\rightarrow$ Depois do roteamento. A distância pode aumentar caso não tenha recursos suficientes de roteamento.

As funções quadrática e exponencial encontraram as melhores soluções em 7 e 8 casos, respectivamente. Para maioria das redes, a *malha* acha uma solução próxima do mínimo, exceto para as redes grandes e densas. Os melhores resultados estão destacados com um asterisco ($*$) e com negrito. No caso de mesmo custo de distância, é levado em consideração o critério de número de soluções válidas como critério de desempate. Estes valores estão apresentados nas últimas quatro colunas. Para alguns casos ocorre uma redução drástica, pois muitos posicionamentos não são validados. Por exemplo, a rede 7 tem mais de 900 soluções com custo 4, porém apenas 1 ou 2 após a validação do roteamento. O motivo é um vértice de grau 20, 3 vértices com grau 8 ou mais e 13 vértices do grau entre 4 e 7, sendo mapeada em uma *malha* com 7×10 elementos. Isto ocorre devido a dois fatores. Primeiro, devido à presença e interconexão dos *hubs*. Segundo, devido à simplicidade do algoritmo de roteamento adotado que é guloso ou faz

Tabela 2. Avaliação das funções de custo para Arquitetura em Malha.

N	V/A, G_{max}, M	$Distancia_{Max}$ Antes → Depois				Soluções Antes → Depois			
		Linear	Quad	Expo	Limiar	Linear	Quad	Expo	Limiar
0	34/43, 6, 2	2→2	2→2	2→2	2→**2****	84→84	437→437	616→616	970→**811****
1	43/73, 8, 2	3→3	3→**3***	3→3	3→3	3→3	209→**70***	614→32	13→13
2	43/65, 15, 3	3→3	3→3	3→3	3→**3****	12→12	276→232	627→214	883→**256****
3	48/78, 11, 2	3→3	3→**3***	3→3	3→3	5→5	88→**33***	377→12	801→11
4	63/146, 11, 2	4→4	4→4	3→**3***	3→4	2→1	255→15	4→**1***	33→3
5	59/97, 8, 2	3→3	3→**3***	3→3	3→3	37→37	298→**52***	591→36	864→10
6	61/103, 11, 2	4→4	3→**3***	3→**3***	4→4	9→9	3→**3***	73→**3***	987→35
7	65/109, 20, 3	4→4	4→4	4→4	4→**4***	80→1	634→1	995→1	995→**2***
8	66/195, 53, 5	6→6	6→6	6→**6***	6→6	117→16	471→81	655→**142***	618→44
9	72/130, 12, 2	4→4	4→4	4→**4***	4→4	2→2	202→11	832→**25***	993→1
10	74/103, 12, 2	4→4	3→**3***	3→3	3→3	50→50	44→**44***	291→30	1→1
11	85/106, 12, 2	3→3	3→**3***	3→3	3→3	4→4	106→**41***	415→28	611→31
12	86/112, 16, 3	4→4	3→4	3→**4***	4→4	31→5	1→14	52→**19***	988→21
13	91/150, 21, 3	5→5	4→4	4→**4***	4→4	8→6	36→2	519→**7***	811→3
14	95/158, 11, 2	4→4	4→4	3→**4***	4→4	1→1	282→15	1→**26***	999→24
15	103/149, 16, 3	5→5	4→4	4→4	4→**4***	15→1	66→2	589→30	849→**34***
16	104/255, 22, 4	5→6	4→**5***	4→5	4→6	1→9	1→**5***	106→3	300→47
17	148/546, 35, 4	10→**11***	-	-	-	3→**1***	-	-	-
18	188/380, 15, 3	7→8	5→6	4→**5***	4→6	8→28	32→45	1→**2***	3→4
19	247/1114, 46,7	-	-	-	-	-	-	-	-
20	321/540, 83, 6	11→12	8→8	15→15	7→**8***	1→2	5→1	2→1	17→**3***

o *maze routing* de forma gulosa também. Ainda, pode ocorrer um aumento no número de soluções encontradas após o roteamento, em casos nos quais a distância aumenta. Por exemplo, a rede 16 tinha uma solução com custo 4 no posicionamento. Porém só validou roteamentos com distância 5 para função quadrática, onde encontrou 5 soluções.

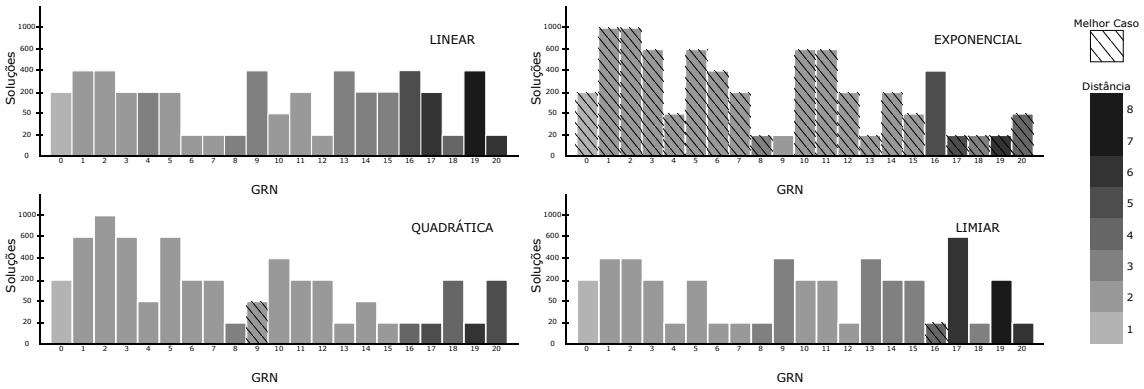

Figura 5. Número de soluções validadas com distância máxima para 1-hop.

A função limiar apresenta 5 soluções melhores que a quadrática e exponencial. De uma forma geral, a inserção das novas funções de custo produziram resultados melhores mesmo para *malha* que tem menos recursos de conexão. Esses resultados chegam a ter uma diferença entre os resultados da função linear e exponencial na ordem de três passos, na Rede 18, por exemplo. Além disso, o número de soluções legalizadas aumentou para os casos em que a maior distância se manteve igual entre as funções.

Para os resultados das arquiteturas *1-hop* e *chess*, optamos por uma forma gráfica. A Figura 5 apresenta 4 gráficos, um para cada função de custo na arquitetura *1-hop*.

Para cada rede e cada função de custo, é apresentada uma barra em escala de cinza que informa a distância máxima e a quantidade de soluções após o roteamento. A cor mais clara mostra as soluções com distância máxima 1, que ocorrem apenas para rede 0. Cores mais escuras representam distâncias maiores. Pode-se observar que, as funções quadrática e exponencial encontram 14 soluções de custo 2. A função limiar não se destaca para a arquitetura *1-hop*, exceto para as duas redes que são a rede 16 e 18.

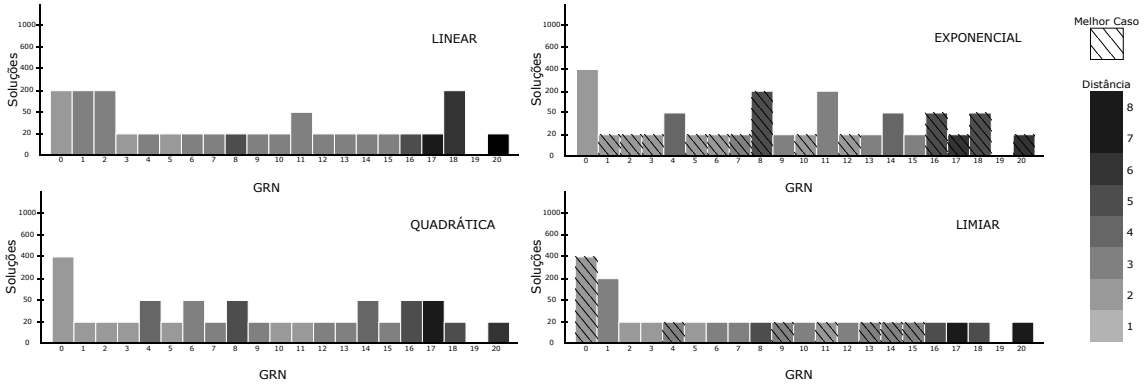

Figura 6. Número de soluções validadas com distância máxima para chess.

O tamanho da barra representa o número de soluções válidas após o roteamento. Barras hachuradas são as melhores soluções dentre as quatro funções de custo. Como pode-se ter até 1.000 soluções válidas, para visualizar o nível de dificuldade com poucas ou muitas soluções, foi realizado a seguinte quantificação em 6 grupos: 0–20, 21–50, 51–200, 201–400, 401–600 e mais de 600. Cada grupo equivale a uma graduação na escala de 1 a 6 da barra. Considere a seguir as seguintes redes da Figura 5, para a função linear. A rede 0 tem graduação 3, isto é, tem de 51 a 200 soluções válidas. Já a rede 1, tem graduação 4, que representa de 201 a 400 soluções. Por fim, a rede 6 tem graduação 1, tendo entre 1 a 20 soluções, ou seja, é mais difícil obter um roteamento válido. Podemos observar que, a exponencial e a quadrática, além de gerar soluções que reduzem a distância máxima, ainda geram mais soluções com roteamento válido. Para as redes mais complexas, que são as redes 17 e 19, a quadrática e exponencial dão a melhor solução e para a rede 20, a melhor solução é da função exponencial.

A Figura 6 mostra os resultados para a arquitetura *chess*. Pode-se observar que apenas 7 redes tem solução de custo 2, enquanto a *1-hop* tem 14 para as funções exponencial e quadrática. Para as redes com distância 3, a função limiar mostrou melhores resultados e para as redes maiores e mais complexas, a função exponencial foi melhor. O número de soluções válidas também é bem menor. Devido à irregularidade da *chess*, o roteamento precisa ser melhorado para explorar, sem muitas perdas, o espaço de soluções.

5. Trabalhos Relacionados

Uma rede reguladora representada como grafo Booleano tem um mapeamento direto em FPGAs, que foi explorado por vários trabalhos [Zerarka et al. 2004, Ferreira and Vendramini 2010, Miskov-Zivanov et al. 2011, da Silva et al. 2017, Manica 2019]. Uma implementação com rede Booleana probabilística usando redes sintéticas foi apresentada por [Zerarka et al. 2004] e comparada com resultados em Matlab. Redes usando lógica de limiar foram propostas

em [Gowda 2007] para implementação em silício avaliando a complexidade dos números de entradas e saídas de cada gene. Outras implementações em FPGA descritas em Verilog para redes Booleanas foram apresentadas em [Miskov-Zivanov et al. 2011] e [Manica 2019] mostrando um ganho de desempenho comparado ao simulador Booleanet implementado em *software* [Albert et al. 2008]. Enquanto os trabalhos anteriores foram apenas simulados ou avaliados em placas isoladas, para simplificar a tarefa de projeto e disponibilizar as ferramentas para comunidade em geral, um gerador de código para FPGA com acesso nas plataformas de nuvem foi proposto em [da Silva et al. 2017] e [Bragança et al. 2021] que executam na plataforma HARP da Intel e nos FPGAs da Xilinx na Amazon. Os geradores mostraram ganhos de 12× e 64× em comparação com uma GPU V100 e um processador com 64 núcleos, respectivamente.

Entretanto, a compilação para FPGA pode demorar algumas horas, o que dificulta o uso dessas ferramentas em uma etapa de desenvolvimento, onde algoritmos dinâmicos [Ferreira et al. 2014] podem mitigar este problema. Apenas o trabalho proposto em [Ferreira and Vendramini 2010] faz o mapeamento de um acelerador CGRA dinâmico no FPGA que permite a reprogramação sem alterar a configuração do FPGA usando uma camada virtual já pré-programada no FPGA. Porém, a comunicação dos genes é feita com uma rede de interconexão global que consome muitos recursos e não escala. Neste artigo estamos propondo o uso de CGRA com uma rede local de vizinhança escalável. Em comparação com a solução direta em FPGA, ao executar uma computação em 2 ou 3 ciclos, o acelerador será apenas 2-3× vezes mais lento, porém é flexível e ajustável em poucos milissegundos, eliminando o processo de compilação para o FPGA.

6. Considerações Finais

Este trabalho apresentou uma solução para mapeamento de redes reguladoras com grafos livres de escala em arquiteturas reconfiguráveis. O mapeamento foi implementado com o algoritmo *Simulated Annealing*, avaliando quatro funções de custo. Vinte e uma redes reguladoras da literatura foram mapeadas em três arquiteturas *malha*, *chess* e *1-hop*, resultando em mapeamentos próximos do ótimo. O tempo de execução do mapeamento é da ordem de poucos segundos, que permitiu que fossem avaliadas 4 funções de custo diferentes e, para cada uma delas foram avaliadas 1.000 soluções. A melhor função depende da rede e da arquitetura. As três novas funções propostas e avaliadas neste trabalho se mostraram superior à função linear que é utilizada em posicionamentos de FPGA e CGRA tradicionais. Os experimentos mostraram que, em geral, as funções limiar e exponencial penalizam o pior caso gerando resultados melhores. Trabalhos futuros irão desenvolver um gerador de arquiteturas para disponibilizar o acesso às ferramentas de simulação de redes Booleanas na nuvem usando FPGA de alto desempenho com a reconfiguração dinâmica proposta neste trabalho. O uso de posicionamento em *hardware* [Vieira et al. 2021] e *overlays* [Silva et al. 2019] também serão explorados.

Referências

Albert, I., Thakar, J., Li, S., Zhang, R., and Albert, R. (2008). Boolean network simulations for life scientists. *Source code for biology and medicine*, 3(1):1–8.

Aldana, M. (2003). Boolean dynamics of networks with scale-free topology. *Physica D: Nonlinear Phenomena*, 185(1):45–66.

Barabási, A.-L. (2009). Scale-free networks: a decade and beyond. *science*, 325(5939):412–413.

Bragança, L., Penha, J., Canesche, M., Ribeiro, D., Nacif, J. A. M., and Ferreira, R. (2021). An open-source cloud-fpga gene regulatory accelerator. In *Anais do XXII Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 240–251. SBC.

Canesche, M., Menezes, M., Carvalho, W., Torres, F. S., Jamieson, P., Nacif, J. A., and Ferreira, R. (2020). Traversal: A fast and adaptive graph-based placement and routing for cgras. *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*.

Carvalho, W., Canesche, M., Reis, L., Torres, F., Silva, L., Jamieson, P., Nacif, J., and Ferreira, R. (2020). A design exploration of scalable mesh-based fully pipelined accelerators. In *International Conf on Field-Programmable Technology (ICFPT)*. IEEE.

da Silva, L. B., Almeida, D., Nacif, J. A. M., Sánchez-Osorio, I., Hernández-Martínez, C. A., and Ferreira, R. (2017). Exploring the dynamics of large-scale gene regulatory networks using hardware acceleration on a heterogeneous cpu-fpga platform. In *IEEE Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig)*.

Ferreira, R., Denver, W., Pereira, M., Carro, L., and Wong, S. (2014). A run-time modulo scheduling by using a binary translation mechanism. In *Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*.

Ferreira, R. and Vendramini, J. (2010). Fpga-accelerated attractor computation of scale free gene regulatory networks. In *Field Programmable Logic and Applications FPL*.

Gowda, T. t. (2007). Threshold logic gene regulatory networks. In *2007 IEEE International Workshop on Genomic Signal Processing and Statistics*, pages 1–4. IEEE.

Lee, C. Y. (1961). An algorithm for path connections and its applications. *IRE transactions on electronic computers*, (3):346–365.

Manica, M. t. (2019). Fpga accelerated analysis of boolean gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Miskov-Zivanov, N., Bresticker, A., Krishnaswamy, D., Kashinkunti, P., Marculescu, D., and Faeder, J. R. (2011). Regulatory network analysis acceleration with reconfigurable hardware. In *IEEE Engineering in Medicine and Biology Society*.

Murray, K. E., Petelin, O., Zhong, S., Wang, J. M., Sha, E., Graham, A. G., Wu, J., Walker, M. J., et al. (2020). Vtr 8: High-performance cad and customizable fpga architecture modelling. *ACM Trans on Reconfigurable Technology and Systems (TRETS)*.

Silva, L. B. D., Ferreira, R., Canesche, M., Penha, J., Jamieson, P., and Nacif, J. A. M. (2019). Ready: A fine-grained multithreading overlay framework for modern cpu-fpga dataflow applications. *ACM Transactions on Embedded Computing Systems (TECS)*.

UFV (2022). Grn. `https://github.com/lesc-ufv/grn_hw_accelerator`.

Vieira, M., Canesche, M., Bragança, L., Campos, J., Silva, M., Ferreira, R., and Nacif, J. A. (2021). Reshape: A run-time dataflow hardware-based mapping for cgra overlays. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE.

Zerarka, M., David, J., and Aboulhamid, E. (2004). High speed emulation of gene regulatory networks using fpgas. In *Midwest Symp on Circuits and Systems*. IEEE.