UNIVERSIDADE FEDERAL DO ABC
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Carlos Reynaldo Portocarrero Tovar

Estabilidade e Sincronização em Redes Dinâmicas Discretas Acopladas: uma Abordagem Computacional

Santo André - SP
Junho de 2019

Carlos Reynaldo Portocarrero Tovar

Estabilidade e Sincronização em Redes Dinâmicas Discretas Acopladas: uma Abordagem Computacional

Texto de Qualificação apresentado ao Centro de Matemática Computação e Cognição da Universidade Federal do ABC para obtenção do título de Mestre em Ciências da Computação.

Orientador: Luiz Carlos da Silva Rozante

Santo André - SP
Junho de 2019

Agradecimentos

Agradeço ao meus pais por me ajudar durante toda minha vida, e aos meus orientadores por me ajudar cumprir meus sonhos.

Resumo

Os processos de estabilização e sincronização em redes de entidades dinâmicas interagentes são quase onipresentes na natureza e desempenham um papel muito importante em muitos contextos diferentes, da biologia à sociologia. Redes Dinâmicas Discretas Acopladas (RDDA) são uma classe de modelos para redes de entidades dinâmicas interagentes, que incluem redes Booleanas acopladas, e que apresentam um amplo leque de potenciais aplicações, principalmente em Biologia de Sistemas. Apesar da sua importância, existem relativamente poucos estudos focados em estabilidade e sincronização envolvendo essa classe específica de modelos, em particular, estudos baseados em abordagens computacionais. O objetivo desse trabalho consiste em desenvolver um método computacionalmente eficiente que, dada uma RDDA como entrada, seja capaz de responder se ela contém ou não estados de sincronização estáveis, bem como identificá-los.

Palavras-chave: Redes Dinâmicas Discretas Acopladas, Estabilidade, Sincronização, Satisfazibilidade, Problema da Transversal Mínima.

Abstract

The processes of stabilization and synchronization in networks of dynamic interacting entities are almost ubiquitous in nature and play a very important role in many different contexts, from biology to sociology. Coupled Discrete Dynamic Networks (RDDA) are a class of models for interagent dynamic entity networks, which include coupled Boolean networks, and which present a wide range of potential applications, especially in Systems Biology. Despite their importance, there are relatively few studies focused on stability and synchronization involving this particular class of models, in particular studies based on computational approaches. The goal of this project is to develop a computationally efficient method that, given an RDDA as input, is able to respond whether or not it contains stable synchronization states, as well as to identify them.

Keywords: Coupled Discrete Dynamic Networks, Stability, Synchronization, SAT, Hitting Set.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Estabilidade e sincronização, como fenômenos emergentes em redes de entidades dinâmicas interagentes, despertam interesse há um bom tempo, desde pelo menos o século XVII com Christiaan Huygens e seu estudo sobre sincronismo de relógios de pêndulo. Os processos de estabilização e sincronização são quase onipresentes na natureza e desempenham um papel muito importante em muitos contextos diferentes, como biologia, ecologia, climatologia, sociologia e engenharia, entre outros (ARENAS et al., 2008), daí a existência de uma enormidade de estudos já desenvolvidos nessa área envolvendo diversas abordagens e técnicas para uma ampla variedade de classes de modelos e aplicações que vão, por exemplo, de equações de estabilidade mestre (PECORA; CARROLL, 1998) a mapas acoplados (JALAN; AMRITKAR, 2003), passando por sistemas multi-agentes (WU et al., 2017), entre outros.

Modelos onde as entidades individuais correspondem a redes dinâmicas discretas – denominados *redes dinâmicas discretas acopladas* (RDDA) –, em particular as redes Booleanas acopladas, apresentam um amplo leque de potenciais aplicações em áreas que vão da biologia à física. Apesar da sua reconhecida aplicabilidade, existe relativamente pouco estudo envolvendo essa classe específica de modelos. Apenas recentemente foram desenvolvidos estudos envolvendo estabilidade e sincronização em redes Booleanas acopladas, com ênfase àqueles baseados em produto semi-tensorial (STP, do Inglês *semi-tensor product*) de matrizes (CHENG, 2007).

1.1 Objetivos

O objetivo básico desse estudo consiste em desenvolver e implementar um método eficiente que, dada uma RDDA como entrada, seja capaz de responder se ela contém ou não campos atratores estáveis (ver definição na Seção 2.5) e, em caso positivo, seja capaz de identificá-los.

Grosso modo, um campo atrator estável é uma atribuição de atratores "locais", para cada entidade individual da rede, tal que torna o sistema globalmente estável, no sentido de que a dinâmica de toda rede local fica "confinada" – para todo t – no atrator definido pela atribuição.

Quando rotulamos um método como "eficiente", estamos nos referindo a um método cujo tempo de resposta está na ordem de 2 ou 3 dezenas de horas, para entradas (RDDAs) com centenas de redes locais, sendo que cada uma delas possa ter até algumas milhares de variáveis.

A fim de alcançarmos esse objetivo será adota uma estratégia em dois passos: num primeiro momento, nosso método calculará os atratores de todas as redes locais; num segundo momento, com os atratores de cada RDD em mãos, construímos atribuições estáveis como combinações de atratores locais.

Para o primeiro passo, desenvolveremos duas abordagens:

- Primeiramente, adaptaremos o algoritmo de Dubrova e Teslenko (2011), de modo a combiná-lo com o algoritmo de Carastan-Santos et al. (2017), que é um algoritmo de alto-desempenho baseado na arquitetura GPU/CUDA e originalmente desenvolvido para o problema *Hitting Set*. Essa abordagem implica num desafio subjacente, cuja superação por si só representa um objetivo implícito: como um dos requerimentos do algoritmo de Dubrova e Teslenko é avaliação de sentenças lógicas, teremos que fazer a redução [*SAT* —> *Hitting Set*] a fim de podermos usar o algoritmo de Carastan-Santos *et al* como um SAT-*solver*.

- Em seguida, para fins de comparação e avaliação com a abordagem acima, implementaremos a abordagem original (baseada em SAT "classico") proposta por Dubrova e Teslenko; porém, adaptada

de modo a considerar a presença de sinais externos, que é uma característica necessária à aplicação no contexto de RDDAs.

Outro objetivo implícito nesse trabalho é desenvolvimento de implementações e experimentos a fim de realizar prova de conceito e avaliar desempenho.

1.2 Justificativa

Devido a grande importância dos problemas de estabilidade e sincronização em redes de entidades dinâmicas interagentes, uma enorme quantidade de estudos analíticos já foram desenvolvidos nessa área, em especial envolvendo modelos contínuos, como por exemplo todos aqueles desenvolvidos a partir dos clássicos trabalhos de Kuramoto (KURAMOTO, 1975) e das funções de estabilidade mestre de Pecora e Carrol (PECORA; CARROLL, 1998), entre outros.

Com relação a modelos discretos, vale mencionar que há trabalhos analíticos envolvendo mapas acoplados (mapas logísticos, mapas sine-circle, mapas quadráticos, etc) (JALAN; AMRITKAR, 2003), visto que estes sistemas são conhecidos por terem topologias complexas. Assim, as populações de mapas acoplados através de um padrão complexo de interações são candidatos naturais para estudos de estabilidade e sincronização como uma característica da população.

Há também muitos trabalhos envolvendo aspectos estruturais da rede e dos processos de estabilização e sincronização, como por exemplo aqueles que mostram a influência da distribuição dos graus (conectividade) na rede (NISHIKAWA et al., 2003), da distribuição das direções e pesos das arestas (MOTTER; ZHOU; KURTHS, 2005), da estrutura do acoplamento (SCHAUB et al., 2016) e dos limitantes espectrais da rede (MOTTER, 2007).

Convém mencionar também estudos de sincronismo com abordagens baseadas em sistemas multi-agentes. O foco desses trabalhos em geral está relacionado com aspectos de controle (WU et al., 2017; XIANG; LI; HILL, 2017) e robustez (SEYBOTH et al., 2015) desses sistemas, sendo que há muitas aplicações baseadas nessa abordagem.

É importante destacar que com relação a estudos envolvendo estabilidade e sincronização em redes dinâmicas discretas acopladas, especificamente em redes Booleanas acopladas, que é um dos focos dessa proposta de trabalho, conforme mencionamos acima, há alguns estudos recentes que são baseados em produto semi-tensorial (STP, do Inglês *semi-tensor product*) de matrizes (CHENG, 2007).

Todavia, uma questão a ressaltar nesses trabalhos envolvendo STP é que eles, ou tratam de casos que lidam com poucas redes, como por exemplo os trabalhos de Li e Lu (LI; LU, 2013) e de Chen *et al* (CHEN; LIANG; LU, 2017) que estudam casos envolvendo apenas duas redes Booleanas acopladas; ou tratam de casos com padrões de conexão mais específicos, como por exemplo Li e Chu (LI; CHU, 2012), que estudam o caso de um *array* de redes Booleanas acopladas.

Uma outra importante restrição associada a esses métodos consiste no fato de que o tamanho da representação matricial das regras lógicas das redes Booleanas – necessárias quando a abordagem é baseada em STP – cresce exponencialmente com o número total de variáveis do sistema, o que torna esses modelos de difícil tratamento em termos de gerência de memória.

Por fim, outro aspecto a ressaltar, é que os trabalhos nessa linha (envolvendo STP) são na sua ampla maioria estudos analíticos focados na descrição das condições necessárias e suficientes para existência de estabilidade e sincronização, não constituindo, portanto, em contribuições baseadas numa abordagem computacional voltados para identificação dos estados de sincronização estáveis, como a que propomos no presente projeto de pesquisa.

1.3 Contribuições

As principais contribuições parciais (já desenvolvidas) deste trabalho são:

1. Desenvolvemos uma modelagem de sistemas de entidades interagentes por redes dinâmicas discretas acopladas (RDDAs).

2. Desenvolvemos e implementamos uma abordagem inovadora para o problema de detecção de atratores em redes Booleanas, sendo que para isso tivemos que:

   - adaptar o algoritmo de Dubrova e Teslenko (2011), de modo a combiná-lo com o algoritmo de Carastan-Santos et al. (2017), que é um algoritmo de alto-desempenho baseado na arquitetura GPU/CUDA e originalmente desenvolvido para o problema *Hitting Set*;
   - desenvolver uma redução [SAT —> *Hitting Set*] a fim de podermos usar o algoritmo de Carastan-Santos *et al* como um SAT-*solver*.

3. Implementamos a abordagem original (baseada em SAT "clássico") proposta por Dubrova e Teslenko; porém, adaptada de modo a considerar a presença de sinais externos, que é uma característica necessária à aplicação no contexto de RDDAs.

1.4 Organização do Trabalho

As Seções seguintes estão organizadas da seguinte forma: na Seção 2 e em algumas de suas subseções introduzimos definições preliminares, notação e uma formulação mais precisa do problema em questão, bem como alguns exemplos de aplicações de RDDAs; na Seção 3 e em suas subseções descrevemos um método para cálculo de atratores em RDDAs, em especial descrevemos na subseção 3.2 detalhes das contribuições parciais acima mencionadas; na Seção 4 descrevemos alguns resultados experimentais preliminares e, por fim, na Seção 5 e em suas subseções tecemos as considerações finais, incluindo a identificação dos próximos passos a seguir (subseção 5.3) para a conclusão da dissertação e o correspondente cronograma de trabalho (subseção 5.4).

2 Definições Preliminares, Notação e Exemplos de Aplicações

Aqui introduzimos os conceitos e definições fundamentais à descrição do trabalho, bem como fixamos uma nomenclatura e notação. Além disso, ao final do Capítulo, mencionamos algumas potenciais aplicações.

2.1 Redes Dinâmicas Discretas (RDDs)

Um *sistema dinâmico discreto* consiste em um espaço de estados X e um mapeamento $f : X \to X$ tal que f define a evolução no tempo do sistema. Um *sistema dinâmico discreto finito* (SDF) é um sistema dinâmico discreto (X, f) onde X é finito.

Um SDF pode ser usado para modelar um número finito de elementos (variáveis) conectados entre si e que têm seus estados internos atualizados por funções que representam suas interações. Em outras palavras, seja um conjunto de variáveis $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ que assumem valores num conjunto finito X e sejam as funções $f_1, f_2, \ldots, f_n$ tal que a função $f_i : X^n \to X$ determina o estado da variável x_i.

Não é difícil ver que um conjunto variáveis $\mathcal{X}$, um espaço de estados X^n e uma função $F : (f_1, \ldots, f_n) : X^n \to X^n$ correspondem a um SDF.

Chamamos um sistema j desse tipo de ***Rede Dinâmica Discreta*** (RDD) (WUENSCHE, 1998), o qual denotaremos por $(\mathcal{X}, X^n, F^j)$.

Se as funções em F^j são aplicadas simultaneamente dizemos que a RDD é ***síncrona***. Se o domínio das variáveis em $\mathcal{X}$ é Booleano e as funções em F^j são Booleanas, então j é dita ser uma ***Rede Booleana*** (RB) (KAUFFMAN, 1993).

Assumindo passos de tempo discreto e atualização síncrona das variáveis em $\mathcal{X}$, o próximo estado (tempo $t+1$) de uma Rede Booleana é dado pelas funções de transição F aplicadas ao estado atual (tempo t) como segue:

$$\begin{cases} x_1^{t+1} = f_1(x_1^t, \ldots, x_n^t), \\ x_2^{t+1} = f_2(x_1^t, \ldots, x_n^t), \\ \ldots \\ x_n^{t+1} = f_n(x_1^t, \ldots, x_n^t). \end{cases} \tag{2.1}$$

2.2 Grafo de Transições de Estados (GTE)

A dinâmica de uma rede dinâmica discreta $j = (\mathcal{X}, X^n, F^j)$ pode ser representada por um grafo dirigido $G_j = (V, E)$, denominado *Grafo de Transições de Estado* (GTE), o qual é assim definido: $V = X^n$ e $(x, y) \in E$ se e somente se $F^j(x) = y$. Assim, cada aplicação de F^j, que corresponde a uma *transição* de estado da RDD, é representada por uma aresta no GTE. Se existe algum $k \geq 2$, tal que $(F^j)^k(x) = x$, então dizemos que x é *periódico* e denominamos o conjunto $\{x, (F^j)^1(x), (F^j)^2(x), \ldots, (F^j)^k(x)\}$ de *atrator periódico* (ou *período/ciclo atrator*), onde $(F^j)^i(x)$, $1 \leq i \leq k$, denota a i-ésima aplicação de F^j a partir do estado x. Dizemos que $x \in X^n$ é um *atrator pontual* (ou *ponto fixo* ou *estado estacionário*) se $(F^j)^1(x) = x$.

Dado um estado $x \in X^n$, dizemos que $(F^j)^2(x)$ é o *sucessor* de $(F^j)^1(x)$ e x é a *pré-imagem* (ou *predecessor*) de $(F^j)^1(x)$. Um estado pode ter mais do que uma pré-imagem, sendo o número total de pré-imagens de um estado o seu *grau de entrada*. Os estados que são pré-imagens podem ter suas próprias pré-imagens e assim por diante. Se um estado não tem pré-imagens ele é dito ser um estado *jardim do*

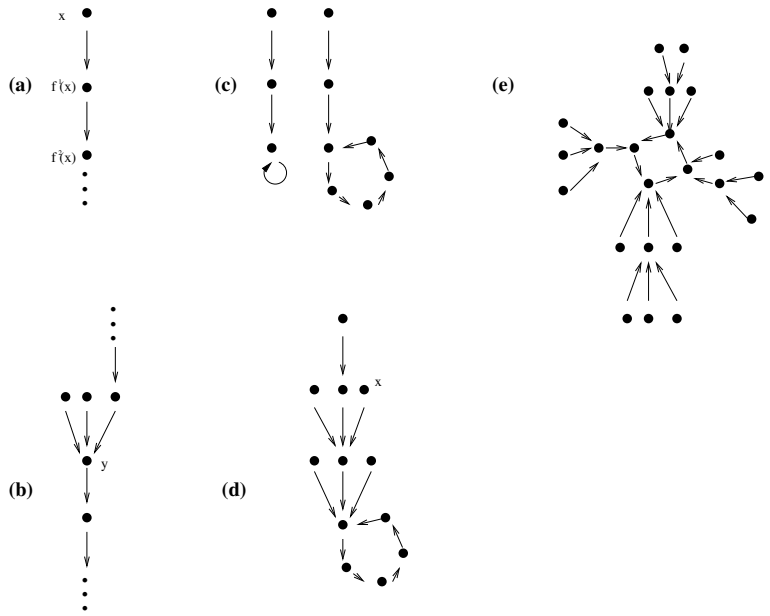

Figura 1 – (a): Parte de uma órbita $O(x)$ em um espaço de estados X, onde $(F^j)^2(x)$ é o *sucessor* de $(F^j)^1(x)$ e x é a *pré-imagem* (ou *predecessor*) de $(F^j)^1(x)$. (b): O estado y tem grau de entrada 3. (c): Exemplo de atrator pontual (à esquerda) e periódico (à direita); (d): Exemplo de atrator periódico com uma de suas árvores transientes. O estado x é um exemplo de estado do tipo *jardim do Éden*. (e): Exemplo de *bacia de atração*.

Éden. Representaremos como $\mathcal{A}^j$ ao conjunto de todos os atractores $\{a_1, a_2, a_3, ..., a_n\}$ pertencentes à rede j, já sejam pontuais o periódicos.

Em RDDs as trajetórias sempre levam à atratores, que podem ser pontuais (contendo um estado) ou periódicos (contendo dois ou mais estados); o comprimento (ou período) de um atrator é o seu número de estados e a parte de uma trajetória que está fora do atrator é chamada de parte transiente. Tome um estado de um atrator e as suas pré-imagens, excluindo a pré-imagem que está no próprio atrator; agora considere as pré-imagens destas pré-imagens e assim por diante até que os estados do tipo *jardim do Éden* sejam alcançados; a parte do grafo do espaço de estados que contém estes estados é uma árvore transiente enraizada no estado que está no atrator. Considerando os estados de um atrator e suas correspondentes árvores transientes; o grafo contendo todos estes estados é chamado de *bacia de atração*, sendo que, podem existir bacias de atração sem as árvores transientes, ou seja, apenas com os estados do atrator. Agora considerando o grafo contendo o conjunto de todas as bacias de atração de um espaço de estados; este grafo é chamado *campo atrator* e o seu número de vértices é igual ao número de estados do espaço de estados. A Figura 1 ilustra essas definições e termos associados à dinâmica de RDDs.

2.3 Computação de Atratores em Redes Dinâmicas Discretas

Uma questão importante em RDDs é: dado uma rede dinâmica discreta $(\mathcal{X}, X^n, F^j)$, é possível predizer estruturas (atratores, bacias de atração, etc) de sua dinâmica?

Com relação ao problema específico de identificação de atratores, podemos classificar os métodos já desenvolvidos nos seguintes grupos:

i) Métodos enumerativos ou parcialmente enumerativos, tais como o proposto por Berntenis e Ebeling (2013), o qual opera em subespaços selecionados do espaço inteiro da rede.

ii) Métodos baseados em simulação, que tentam encontrar atratores escolhendo heuristicamente vários estados iniciais e simulando a evolução do sistema para cada condição inicial (JONG; GEISELMANN;

HERNANDEZ, 2003; KARL; DANDEKAR, 2013; KLAMT; SAEZ-RODRIGUEZ; GILLES, 2007). Esses métodos apresentam a deficiência de potencialmente não cobrirem todos os atratores devido à geração aleatória dos estados iniciais.

iii) Métodos baseados na formulação do problema em termos do problema do diagrama de decisão binária (BDD, do Inglês *binary decision diagram*) (BRYANT, 1986; CARA et al., 2007), que é uma estrutura de dados usada para descrever as funções Booleanas e apoiar a computação das operações lógicas, cujo tamanho cresce exponencialmente no número de variáveis da rede e na representação das funções, limitando, portanto, sua aplicação apenas a redes muito simples devido os excessivos requerimentos de memória envolvidos.

iv) Métodos baseados na formulação do problema de agregação (ZHAO; KIM; FILIPPONE, 2013), cuja ideia central consiste em particionar a rede Booleana de entrada em sub-redes, sendo que em cada sub-rede é aplicado um algoritmo (por exemplo o algoritmo de Johnson (JOHNSON, 1975)) para encontrar atratores.

v) Métodos baseados em uso do semi-tensor product(STP) de matrizes (CHENG; QI, 2010), cuja ideia básica é criar uma matriz de representação de variáveis e funções lógicas da rede booleana. A partir da análise e manipulação das propriedades algébricas desta matriz e possível identificar estruturas (por exemplo atratores) do GTE.

vi) Métodos baseados na formulação do problema em termos do problema de satisfazibilidade (SAT) (DUBROVA; TESLENKO, 2011; GUO et al., 2014). Esses métodos baseados em SAT requerem menos espaço do aqueles baseados em BDD, além de em geral levarem a buscas mais eficientes – em termos de tempo – porque operam sem varrer todo o espaço de estados. Esses algoritmos usam um SAT-solver para identificação de caminhos no grafo de transição de estados (STG), da seguinte forma: primeiro, geram uma fórmula proposicional que representa a aplicação da função de transição da rede por p passos de tempo; depois, checam uma atribuição satisfazível para essa fórmula proposicional, sendo que uma atribuição satisfazível corresponde a um caminho válido no STG. Esse processo é repetido iterativamente para valores cada vez maiores de p até que todos os atratores sejam identificados.

A presente proposta de trabalho usa intensamente em uma de suas etapas de desenvolvimento essa ideia de identificação de atratores via SAT, em particular o algoritmo de Dubrova e Teslenko (2011) (ver Seção 3.2).

2.4 Redes Dinâmicas Discretas Acopladas (RDDAs)

Considere uma rede de m entidades dinâmicas interagentes, onde cada uma delas entidades é modelada como uma RDD. Nesse sistema (uma "rede de redes") podemos distinguir três elementos constituintes:

1. O padrão de conexão – a estrutura da rede – entre as entidades dinâmicas, representado por um grafo dirigido G;

2. As regras que governam a dinâmica local das entidades, representadas por RDDs;

3. A natureza da interação entre as entidades, isto é, a forma como os acoplamentos inter-entidades (entre RDDs) ocorrem, que é definida por uma função de acoplamento.

Denominamos um sistema como esse de ***Rede Dinâmica Discreta Acoplada*** (RDDA)[1].

Essa descrição de RDDA é bastante genérica e permite, portanto, a identificação de várias configurações de RDDAs, as quais têm relação com restrições associadas aos elementos acima: i) restrições impostas pelo padrão de conexão (grafos regulares ou algum modelo de rede complexa, por exemplo); ii) restrições impostas pelas características das redes locais (RDDs), por exemplo o tipo de função, se são ou não idênticas, se contém ou não o mesmo número de variáveis, etc e iii) restrições impostas pelas propriedades da função de acoplamento.

Todas essas restrições impactam a dinâmica global da rede e, quando combinadas, definem classes mais específicas de RDDAs. No contexto desse trabalho assumiremos que as RDDAs em estudo são tal que:

i) Os grafos que definem a estrutura da rede (o padrão de conexão) são arbitrários.

ii) As RDDs podem ser não idênticas[2], mas assumimos que o número de variáveis de estado em todas elas é o mesmo e, por facilidade de notação, assumiremos que todas têm o mesmo padrão de nomeação; ou seja, todas as RDDs têm n variáveis de estado nomeadas $x_1, \ldots, x_n$. Vale realçar que embora diferentes RDDs tenham o mesmo número de variáveis e estas sejam igualmente nomeadas, suas funções de transição F^j, $1 \leq j \leq m$, são diferentes. Além disso, assumiremos também que as RDDs são todas redes Booleanas[3] síncronas.

iii) Com relação à forma como os acoplamentos entre as RDDs ocorrem, dado uma rede local j, ela pode tanto influir como também ser influenciada por outras RDDs. Nesse contexto, suponha u e j duas RDDs; se existe uma aresta orientada (u, j) em G, então dizemos que u é *convizinha* de j e dizemos que j é *vizinha* de u.

Agora suponha uma rede local u com $\mathcal{X}^u$ variáveis de estado. Se u influi em alguma RDD vizinha, então existe um subconjunto $\mathcal{S}^u \subseteq \mathcal{X}^u$ de variáveis de estado cujos valores (estados) são utilizados indiretamente pelas RDDs vizinhas. Denominamos as variáveis em $\mathcal{S}^u$ de *variáveis de saída* de u.

Se u é convizinha de j, assumimos que u emite (a partir de $\mathcal{S}^u$) através de uma função $h_u : X^n \to X$ um sinal y_u que serve ao acoplamento com j. Chamamos $I = [h_1, \ldots, h_m]$ de *função de acoplamento* e $Y = [y_1, \ldots, y_m]$ de *sinal de acoplamento*.

Se j é influenciada por RDD convizinhas, então existe um subconjunto $\mathcal{E}^j \subseteq \mathcal{X}^j$ de variáveis de estado sobre as quais incidem sinais delas oriundos (sinais externos). Denominamos as variáveis em $\mathcal{E}^j$ de *variáveis de entrada* de j. A Fig. 2 contém um diagrama ilustrativo dessa modelagem.

Assim, tem-se que o estado de uma variável i num tempo $t+1$ de uma RDD j, considerando a influência simultaneamente exercida por sinais externos que derivam do acoplamento com outras RDDs e por sinais locais com origem na própria j, é definido por

$$x_i^j(t+1) = \begin{cases} f_i^j(x_k^j(t), \ldots, x_l^j(t), y_u(t), \ldots, y_r(t)), & \text{se } i \in \mathcal{E}^j \\ f_i^j(x_k^j(t), \ldots, x_l^j(t)), & \text{se } i \in \mathcal{X}^j \setminus \mathcal{E}^j, \end{cases} \tag{2.2}$$

$1 \leq i \leq n$, $1 \leq j \leq m$, onde:

[1] Ao invés de modelar o sistema como uma rede de m RDDs acopladas, com cada uma delas contendo n variáveis, poderíamos modelá-lo como única RDD com mn variáveis, o que provavelmente tornaria o modelo estruturalmente mais simples porque, afinal, quando fazemos "rede de redes" injetamos um elemento adicional (a função de acoplamento) ao modelo que complica sua manipulação; porém, esta é uma opção de modelagem que captura mais claramente a natureza identitária das entidades locais, característica que é comumente observada e fortemente intuitiva em muitos sistemas reais (mencionamos alguns exemplos em Aplicações / Seção 2.9)

[2] No caso de serem idênticas há impacto no custo de computação, visto que bastaria calcular os atratores locais de uma única RDD.

[3] Assumimos as RDDs como sendo redes Booleanas por facilidade de análise; no entanto, na prática admitiremos entradas onde as RDDs são funções lógicas com domínio multi-valor porque nossa abordagem pressupõe um procedimento de pré-processamento que converte funções lógicas multi-valor para funções booleanas, que pode ser adaptado de Dubrova, Teslenko e Ming (2010).

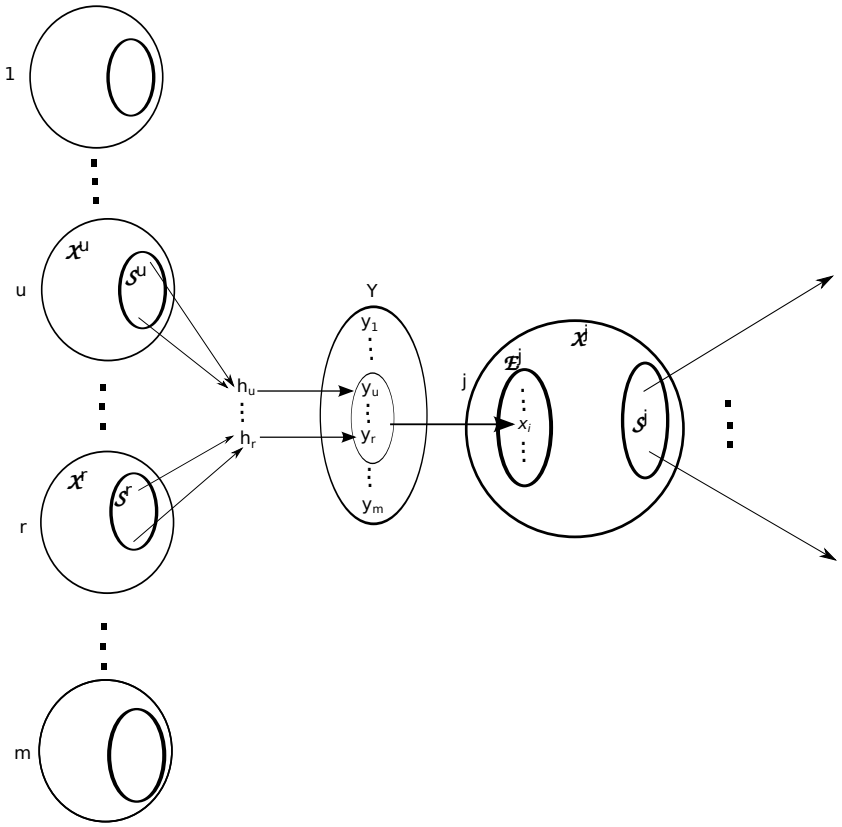

Figura 2 – Uma representação gráfica da forma pela qual as RDDs $u, \ldots, r$ (que são convizinhas de j) exercem influência sobre j.

- $f_i^j : X^{n+m} \to X$ é a componente da função de transição que atualiza x_i^j;
- $x_k^j, \ldots, x_l^j$, $1 \leq k, \ldots, l \leq n$, denotam os sinais locais (têm origem na própria j) que incidem sobre x_i^j;
- $y_u, \ldots, y_r$, $1 \leq u, \ldots, r \leq m$, denotam os sinais externos (têm origem nas convizinhas de j) que incidem sobre x_i^j, sendo que

  $$y_w(t) = h_w(x_p^w(t), \ldots, x_q^w(t)),$$

  $u \leq w \leq r$, com $x_p^w, \ldots, x_q^w \in \mathcal{S}^w$, onde $h_w : X^n \to X$ é a componente da função de acoplamento responsável por atualizar o sinal de acoplamento y_w.

2.5 Sincronização e Estabilidade em RDDAs

Considere uma RDD j com $y_u, \ldots, y_r$ sinais externos (sinais de acoplamento) incidindo sobre j. Seja $\mathcal{A}^j$, $1 \leq j \leq m$, o conjunto dos atratores locais da dinâmica associada a j, em presença dos possíveis valores de $y_u, \ldots, y_r$.

Uma *atribuição* A é um mapeamento $[(1, \mathsf{a_w}), \ldots, (j, \mathsf{a_v}), \ldots, (m, \mathsf{a_z})]$, onde $\mathsf{a_w} \in \mathcal{A}^1, \ldots, \mathsf{a_v} \in \mathcal{A}^j, \ldots, \mathsf{a_z} \in \mathcal{A}^m$. Isto é, uma *atribuição* A é um mapeamento $1:1$, um atrator para cada RDD, sendo que cada atrator é oriundo da dinâmica da RDD correspondente (ver exemplo na Fig. 3). Diremos que uma atribuição A é um *campo atrator estável* (ou uma *atribuição estável*) se para toda rede local j, $1 \leq j \leq m$, observamos a seguinte condição, que chamaremos de *condição de estabilidade*: o(s) valor(res) de saída proporcionado(s) pelo(s) atrator(es) local(is) em A e que provém das células convizinhas à célula j é(são) compatível(eis) com o(s) valor(res) do(s) sinal(is) de acoplamento requerido(s) pelo atrator mapeado para j em A.

Em outras palavras, suponha uma atribuição A na qual o atrator $\mathsf{a_v}$ está mapeado em j e os atratores $\mathsf{a_w}, \ldots, \mathsf{a_z}$ estão mapeados em $u, \ldots, r$, respectivamente, onde as RDDs $u, \ldots, r$ são convizinhas da RDD j. Diremos que A satisfaz a condição de estabilidade se, para todo tempo t e para toda rede local j, $1 \leq j \leq m$, vale que:

$$x_i^j[\mathsf{a_v}] = f_i^j(x_k^j[\mathsf{a_v}], \ldots, x_l^j[\mathsf{a_v}], y_u[\mathsf{a_w}], \ldots, y_r[\mathsf{a_z}]), \forall x_i \in \mathcal{E}^j,$$

sendo que:

- $x_i^j[\mathsf{a_v}]$ denota o valor da variável x_i na RDD j quando j encontra-se no estado corrente do atrator $\mathsf{a_v}$;
- $x_k^j[\mathsf{a_v}], \ldots, x_l^j[\mathsf{a_v}]$ denotam, respectivamente, os valores dos sinais locais $x_k, \ldots, x_l$ quando j encontra-se no estado corrente do atrator $\mathsf{a_v}$;
- $y_u[\mathsf{a_w}], \ldots, y_r[\mathsf{a_z}]$ denotam os valores dos sinais de acoplamento $y_u, \ldots, y_r$ quando as RDDs $u, \ldots, r$ estiverem nos estados correntes dos atratores $\mathsf{a_w}, \ldots, \mathsf{a_z}$, respectivamente.

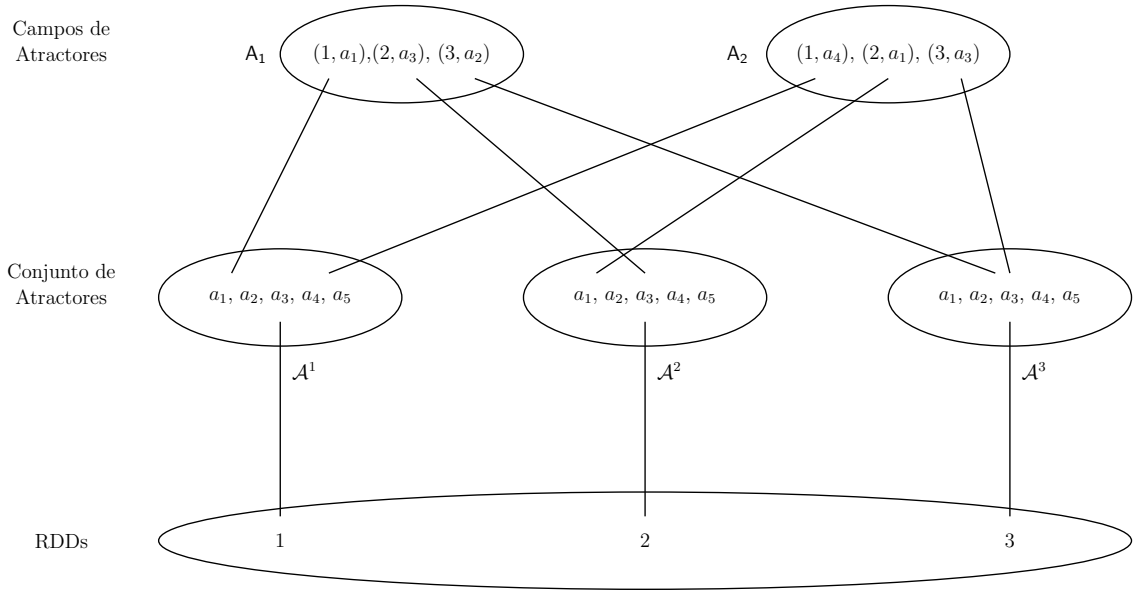

Figura 3 – Exemplo de uma atribuição num sistema com três RDDs. $\mathcal{A}^1$, $\mathcal{A}^2$ e $\mathcal{A}^3$ denotam os conjuntos de atratores das redes locais 1, 2 e 3, respectivamente. A atribuição $\mathsf{A_1} = [(1, \mathsf{a_1}), (2, \mathsf{a_3}), (3, \mathsf{a_2})]$ contém o atrator $\mathsf{a_1}$, oriundo da RDD 1; o atrator $\mathsf{a_3}$ oriundo da RDD 2 e o atrator $\mathsf{a_2}$ oriundo da RDD 3. A atribuição $\mathsf{A_2} = [(1, \mathsf{a_4}), (2, \mathsf{a_1}), (3, \mathsf{a_3})]$ contém o atrator $\mathsf{a_4}$, oriundo da RDD 1; o atrator $\mathsf{a_1}$ oriundo da RDD 2 e o atrator $\mathsf{a_3}$ oriundo da RDD 3.

Dito de outro modo, uma atribuição é um campo atrator estável quando sua aplicação torna o sistema globalmente estável, no sentido de que a dinâmica de toda rede local fica "confinada" – para todo t – no atrator definido pela atribuição.

No contexto desse trabalho o problema de estabilidade consiste em, dado um sistema com m RDDs, queremos encontrar seus campos atratores estáveis, caso existam. Estados estacionários globais correspondem a atribuições estáveis onde os atratores da atribuição são todos eles pontos fixos.

Sincronização completa refere-se ao cenário no qual todas as m RDDs convergem para uma mesma órbita (com respeito a todas as suas n variáveis), de modo que se denotarmos por X^j o estado n-dimensional da j-ésima RDD, *sincronização completa* corresponde a

$$X^1(t) = X^2(t) = X^3(t) = \ldots = X^m(t), \tag{2.3}$$

para todo t.

Sincronização parcial aqui refere-se ao cenário no qual um subconjunto $S = \{x_i, \ldots, x_k\}$, $1 \leq i, \ldots, k \leq n$, das variáveis das RDDs convergem para uma mesma órbita (com respeito a todas variáveis em S), de modo que se denotarmos por S^j o estado $|S|$-dimensional (associado às variáveis em S) da j-ésima RDD, *sincronização parcial* com relação a S corresponde a

$$S^1(t) = S^2(t) = S^3(t) = \ldots = S^m(t), \tag{2.4}$$

para todo t.

Sincronização de saída é um caso de sincronização parcial onde S é igual ao conjunto das variáveis de saída, assumindo que o conjunto de variáveis de saída é o mesmo para toda RDD.

2.6 Problema de Satisfatibilidade Booleana

Na teoria da complexidade computacional, *Boolean Satisfiability Problem* (SAT) foi o primeiro problema identificado como pertencente à classe de complexidade NP-completo (COOK, 1971). O SAT é o problema de determinar se existe uma valoração para as variáveis de uma dada fórmula Booleana F tal que esta valoração satisfaça F. Por exemplo, tomando x_1, x_2, x_3, x_4 como as variáveis Booleanas e a expressão

$$(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4)$$

como a fórmula F, caso exista uma atribuição de valores para as variáveis x_1, x_2, x_3, x_4 tal que torne F *verdadeira*, então ela é dita ser *satisfazível*, caso contrario ela é considerada *insatisfazível*.

2.7 Forma Normal Conjuntiva

Na lógica booleana, uma fórmula está em Forma Normal Conjuntiva do inglês *Conjuntive Normal Form* (CNF) se é uma conjunção de cláusulas, onde uma cláusula é uma disjunção de literais. A formula representada na seção 2.6 e um exemplo de formula em CNF. Qualquer fórmula booleana F pode-se converter a CNF, para fazer a conversão são usadas sucessivamente a lei de De Morgan e a propriedade distributiva em F, ate chegar a uma fórmula booleana equivalente. Quando o número de variáveis n dentro de cada cláusula de uma fórmula em CNF e $n \leq 3$, dizemos que ela se encontra em 3CNF. A maioria dos SAT-solvers precisam que as fórmulas booleanas que estejam em 3CNF. A continuação apresentamos um exemplo de equivalência entre formulas booleanas:

$$(x_1 \leftrightarrow x_2) \rightarrow (\neg x_1 \wedge x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

2.8 Problema *Hitting Set*

O *Hitting Set problem* (HSP) pode ser definido como: Dado um conjunto finito $\mathbb{X}$ e a coleção $\mathcal{S}$ de subconjuntos de $\mathbb{X}$, deve-se achar um subconjunto $H \subseteq \mathbb{X}$ com a menor cardinalidade, tal que:

$$H \cap S \neq \emptyset, \forall S \in \mathcal{S} \tag{2.5}$$

Mais de um subconjunto de $\mathbb{X}$ pode satisfazer as condições da cima. Nos chamamos de $\mathcal{H} = \{H_1, H_2, ..., H_{|\mathcal{H}|}\}$ a coleção de possíveis soluções. Se considerarmos que $\mathcal{S}$ define um hiper-grafo em $\mathbb{X}$ (onde cada conjunto em $\mathcal{S}$ constitui uma hiper-relação), então vemos que o HSP é equivalente ao problema de Cobertura de Vértices em Hiper-grafos. O HSP é um problema NP-difícil (DINUR; SAFRA, 2005).

2.9 Aplicações

A seguir mencionamos algumas diferentes potenciais aplicações relacionadas aos conceitos e modelagem anteriormente descritos.

Redes intracelulares podem ser modeladas como RDDs (em particular redes Booleanas são muito utilizadas devido à sua simplicidade, robustez e compatibilidade com dados qualitativos) e tipos celulares podem ser interpretados como suas bacias de atração (KAUFFMAN, 1993; WUENSCHE, 1998; WUENSCHE, 2004), dentro das quais a dinâmica celular estabiliza-se a partir de vários estados iniciais, de modo que cada atrator (pontual ou cíclico) corresponde a um padrão de expressão e regulação intracelular associado a um tipo celular. Assim, as bacias de atração fornecem um modelo para homeostase celular em resposta às mutações e às pertubações no estado padrão de ativação dos genes.

Se a rede intracelular numa RDDA corresponde a uma RDD, então podemos conceber as bacias de atração global das RDDAs como padrões celulares: um tipo celular corresponde a uma bacia de atração local da rede intracelular e um padrão celular (estrutura no nível tecidual) corresponde a uma bacia de atração global, que é composta por um arranjo de bacias locais posicionadas em células interagentes, tais que as saídas e as entradas das dinâmicas locais em cada célula são compatíveis entre si (ver na Seção 2.5 essa noção de compatibilidade). Um estado estacionário global corresponde a um padrão de regulação intracelular e de comunicação célula-célula associado a um determinado padrão celular. Essa interpretação tem várias aplicações, como por exemplo em estudos de biologia do câncer (HUANG; ERNBERG; KAUFFMAN, 2009) ou biologia do desenvolvimento (BENíTEZ et al., 2008).

Há métodos (MIYANO; TSUTSUI, 2007) de mineração de dados baseados em agrupamentos de dados espontâneos usando redes de osciladores acoplados localmente. Esse tipo de método está intimamente relacionado com a determinação da estrutura da comunidade através de processos de sincronização (BOCCALETTI et al., 2007). A ideia é codificar vetores de dados multivariados (que são os elementos do banco de dados) em vetores de frequências naturais para um modelo dinâmico de osciladores, semelhante ao modelo de Kuramoto (1975), esperando que a dinâmica do sistema agrupe dados semelhantes em *clusters* de sincronização, o que sugere o uso de sincronização parcial conforme descrito anteriormente.

O problema de detecção descentralizada de mudanças abruptas em redes de sensores móveis sem fio podem representar falhas na comunicação ou ataques à determinados sensores. Há situações (HONG; SCAGLIONE, 2004) onde os dados do sensor e a interação entre eles só podem ser codificados no momento da emissão do pulso, o que conduz ao problema da detecção descentralizada de mudanças abruptas via observação de sincronização de osciladores de pulso acoplados em rede após uma perturbação local dos mesmos.

Em logística descentralizada as vezes o gerenciamento é limitado pela capacidade de manter o conhecimento global e/ou a comunicação global, de modo que em muitos sistemas de fluxo de materiais e recursos a coordenação de tarefas de maneira descentralizada é crítica. Existem modelos (LäMMER et al., 2006) de controle descentralizado para fluxo de material em redes baseados na sincronização de serviços oscilatórios nos nós da rede, cuja aplicação demanda técnicas de estabilização e sincronismo.

A produção, dissipação, transmissão e consumo de energia elétrica representam um problema dinâmico e as suas redes de distribuição pode ser modeladas como sistemas de osciladores acoplados. Vários estudos (YANG; NISHIKAWA; MOTTER, 2017; NISHIKAWA; MOLNAR; MOTTER, 2015) envolvendo projeto e controle dessas estruturas envolvem ingredientes de estabilidade e sincronismo.

Em estudos de formação de opinião em grupos ou sociedades uma ideia subjacente é que os indivíduos têm opiniões que mudam sob a influência de outros indivíduos, dando origem a uma espécie de comportamento coletivo, daí que um dos objetivos nessa área é descobrir se é quando um consenso completo ou parcial pode emergir de opiniões inicialmente diferentes. E para isso há estudos (PLUCHINO; LATORA; RAPISARDA, 2005) que fazem uso explícito de modelos de osciladores acoplados que veem da formação

de grupos de opiniões como um processo de estabilização de estados sincronizados.

3 Um Método para Cálculo de Atratores em RDDAs

A fim de alcançarmos o objetivo anteriormente mencionado, será adotada uma abordagem em dois passos: num primeiro momento calculamos os atratores de todas as redes locais; num segundo momento, com os atratores de cada RDD em mãos, construímos atribuições estáveis como combinações de atratores locais. Adiante, descrevemos duas abordagens para o cálculo de atratores locais, um delas usando *Hitting Set Problem* (HSP) e outra usando *Boolean Satisfiability Problem* (SAT). Para ambas abordagens usaremos como base o algoritmo de Dubrova e Teslenko (2011), que é capaz de identificar atratores em redes Booleanas. Este algoritmo faz chamadas a o `SAT-solver` de Eén e Sörensson (2003) chamado *MiniSat*, com o objetivo de identificar caminhos (que são usados para compor as sequências de estados que definem os atratores) no grafo de transições de estados, usados para representar a dinâmica das redes em questão.

3.1 O Algoritmo de Duvrova e Teslenko

O algoritmo de Dubrova e Teslenko adota uma representação simbólica para as transições de estado, que correspondem a arestas no GTE. Isso permite detectar a existência de caminhos no GTE por meio da avaliação da satisfatibilidade de expressões lógicas, com a finalidade de achar vértices repetidos nos caminhos do GTE, os quais representam os atratores descritos na Seção 2.2.

Aqui o estado x de uma RDD no tempo t é representado por um vetor $x^t = [x_1^t, x_2^t, \ldots, x_n^t]$ de variáveis de estado. Denotamos por S_k uma sequência de transições de estados

$$x^{t+0} \rightarrow x^{t+1} \rightarrow x^{t+2} \rightarrow \ldots \rightarrow x^{t+k} \tag{3.1}$$

com k passos de tempo (que vai do passo t até o passo $t+k$), lembrado que k é a quantidade de aplicações da função de transição F (Seção 2.2).

Uma transição de estado S_1, $x^t \rightarrow x^{t+1}$, de acordo com o método de representação simbólica de Burch et al. (1992) é dada por:

$$[x_1^{t+1} \leftrightarrow f_1(x^t)] \wedge [x_2^{t+1} \leftrightarrow f_2(x^t)] \wedge \ldots \wedge [x_n^{t+1} \leftrightarrow f_n(x^t)], \tag{3.2}$$

onde f_i, $1 \leq i \leq n$, é definida como na Seção 2.1, sendo que "$\wedge$" e "$\leftrightarrow$" representam os operadores lógicos para conjunção e bicondicional, respetivamente. Esta representação da transição S_1 é chamada de *formulação proposicional* de S_1, a qual denotamos por $\widehat{S_1}$.

Se existe uma atribuição que satisfaz as variáveis $x_1^t, \ldots, x_n^t, x_1^{t+1}, \ldots, x_n^{t+1}$ em $\widehat{S_1}$, então a transição $x^t \rightarrow x^{t+1}$ corresponde a uma aresta no GTE. Portanto, a sequência S_k de transições corresponde a um caminho válido no GTE, se existe uma atribuição satisfatível para as variáveis em $\widehat{S_k}$.

Como as transições são representadas por funções e o espaço de estados é finito, cada estado no GTE tem um único sucessor, portanto, uma vez que um caminho atinge um *loop* (atrator) ele nunca o deixa. Usando esta premissa, suponha que uma sequência $\widehat{S_k}$ é um caminho válido no GTE, então, para verificar se $\widehat{S_k}$ contém um *loop*, o algoritmo só precisa verificar se o último estado (x^{t+k}) ocorre pelo menos duas vezes em $\widehat{S_k}$.

Representamos internamente em máquina a sequência de estados correspondente a um caminho válido no GTE através de uma estrutura de dados denominada *Path*, que guarda o conjunto de valores atribuídos pelo SAT-*solver* às variáveis

$$x_1^0, \ldots, x_n^0; x_1^1, \ldots, x_n^1; \ldots; x_1^k, \ldots, x_n^k,$$

onde x_i^k denota o valor da i-ésima variável na k-ésima transição. Note que *Path* armazena o valor das n variáveis para cada uma das k transições.

Assim, temos que $Path = [Path_0, \ldots, Path_k]$, onde $Path_i$ denota a sequência de valores que as n variáveis (supondo uma RDD com n variáveis) de estado assumem na i-ésima aplicação da função de transição, sendo que estes valores são 0 (se à variável é atribuído valor lógico `falso`) ou 1 (se à variável é atribuído valor lógico `verdadeiro`). Daí que $Path$ corresponde a uma sequência de 0's e 1's.

Se $Path_i = Path_k$, com $0 \leq i \leq k-1$, significa que o último estado (x^k) já ocorreu e, portanto, existe um *loop* em $\widehat{S_k}$, que corresponde a um atrator.

Depois de identificado um atrator, ele precisa ser atribuído ao conjunto solução $\mathcal{A}$. Essa operação de atribuição é feita por meio do operador lógico $\vee$, da seguinte forma:

$$\mathcal{A} \leftarrow \mathcal{A} \vee \mathcal{B}, \tag{3.3}$$

onde $\mathcal{B}$ denota o atrator que está sendo atribuído ao conjunto solução $\mathcal{A}$.

Para que operação representada pela Eq. 3.3 seja válida necessitamos que tanto $\mathcal{A}$ como $\mathcal{B}$ estejam representados via expressões Booleanas, daí que existe um procedimento que converte o atrator representado em $Path$ (uma sequência de bits) em uma expressão Booleana. Esse procedimento pode ser assim descrito:

- 0's em $Path$ correspondem a negativos em $\mathcal{B}$;
- 1's em $Path$ correspondem a positivos em $\mathcal{B}$;
- os literais (positivos e negativos das variáveis) são conectados pelo operador $\wedge$;
- cada estado do atrator em $Path$ (sequência de bits) corresponde a uma cláusula em $\mathcal{B}$;
- as cláusulas são conectadas através do operador $\vee$.

Para ilustrar, aqui um exemplo de um atrator com dois estados numa rede de 3 variáveis:

$$Path = [010, 110]$$

$$\mathcal{B} = (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

Depois de incluir o atrator recém identificado no conjunto solução, é preciso excluí-lo do espaço de busca, o que é feito através da seguinte operação:

$$\widehat{S_k} \leftarrow \widehat{S_k} \wedge \neg \mathcal{A} \tag{3.4}$$

Os valores de $\mathcal{A}$, no momento em que são adicionados à $\widehat{S_k}$, têm que ser representados no tempo k, então a variável x_i será convertida no x_i^k, pois assim restringimos que o SAT-*solver* devolva os mesmos valores dos atratores já achados. Feito isto, o $Path$ pertencerá a outra bacia de atracão. No caso de o SAT-*solver* devolver *insatisfazível* significa que todos os atratores foram achados e o algoritmo termina.

Se em $Path$ não existem estados repetidos, o tamanho de k é dobrado, ou seja, $k \leftarrow 2k$. Assim, com o aumento de $\widehat{S_k}$, é maior a chance de achar um atrator.

A descrição geral do algoritmo é a seguinte:

1. Gerar a formulação proposicional $\widehat{S_k}$ correspondente às sequências das transições;
2. Converter $\widehat{S_k}$ à Forma Normal Conjuntiva (CNF), denotada por $\mathcal{P}$;
3. Executar o `SAT-solver` na instância $\mathcal{P}$:
   a) Se o `SAT-solver` retornar *satisfazível*, guardar a saída em $Path$ e depois verificar se $Path$ contém um *loop*, e:
      i. Se contém um *loop*, guarda-se o *loop* em o conjunto solução $\mathcal{A}$, excluindo a bacia de atração do espaço de pesquisa e volta para o passo 1;

ii. Caso contrário, duplica-se k, $(k \leftarrow 2k)$ e retorna para passo 1.

b) Se o `SAT-solver` retornar *insatisfazível*, retorna-se a lista de atratores $\mathcal{A}$ e o algoritmo acaba.

Apresentamos agora uma rede Booleana r de três variáveis para explicar o algoritmo:

$$r : \begin{cases} x_1^{t+1} = (x_1^t \vee x_2^t) \wedge (x_1^t \vee x_3^t), \\ x_2^{t+1} = x_1^t \vee x_2^t \vee \neg x_3^t, \\ x_3^{t+1} = \neg x_1^t \vee \neg x_2^t \vee \neg x_3^t. \end{cases}$$

A formulação proposicional da sequência de transições $x^0 \to x^1 \to x^2$ em r, é dada por:

$$\begin{aligned} \widehat{S_2} = (x_1^1 \leftrightarrow ((x_1^0 \vee x_2^0) \wedge (x_1^0 \vee x_3^0))) \\ \wedge (x_2^1 \leftrightarrow (x_1^0 \vee x_2^0 \vee \neg x_3^0))) \\ \wedge (x_3^1 \leftrightarrow (\neg x_1^0 \vee \neg x_2^0 \vee \neg x_3^0)) \\ \wedge (x_1^2 \leftrightarrow ((x_1^1 \vee x_2^1) \wedge (x_1^1 \vee x_3^1))) \\ \wedge (x_2^2 \leftrightarrow (x_1^1 \vee x_2^1 \vee \neg x_3^1))) \\ \wedge (x_3^2 \leftrightarrow (\neg x_1^1 \vee \neg x_2^1 \vee \neg x_3^1)). \end{aligned}$$

Se houver uma atribuição satisfazível mediante o `SAT-solver` para as variáveis

$$x_1^0, x_2^0, x_3^0, x_1^1, x_2^1, x_3^1, x_1^2, x_2^2, x_3^2$$

em $\widehat{S_2}$, então as transições $x^0 \to x^1 \to x^2$ correspondem a um caminho válido no GTE representando a dinâmica da rede r. $\widehat{S_2}$ é convertido em $\mathcal{P}$ na CNF, para depois ser avaliado pelo `SAT-solver`. No caso de termos $Path = [010, 011, 111]$, pode-se verificar que não existe atrator pois $Path_2 \neq Path_0$ e $Path_2 \neq Path_1$. Então dobra-se k e $\widehat{S_4}$ é então gerada, para posteriormente usar o `SAT-solver` em $\mathcal{P}$ novamente e ter como resposta $Path = [011, 111, 110, 101, 111]$. Então, pode-se verificar que $Path_1 = Path_4$, sendo que os estados das transições de 2 até k formam um atrator e são agregados a lista de atratores $\mathcal{A}$. Na iteração seguinte, como foi achado um atrator, k não é dobrado e utiliza-se a Equação 3.4 em $\widehat{S_4} \leftarrow \widehat{S_4} \neg \mathcal{A}$. Deste jeito remove-se o atrator encontrado e estados relacionados do espaço de busca.

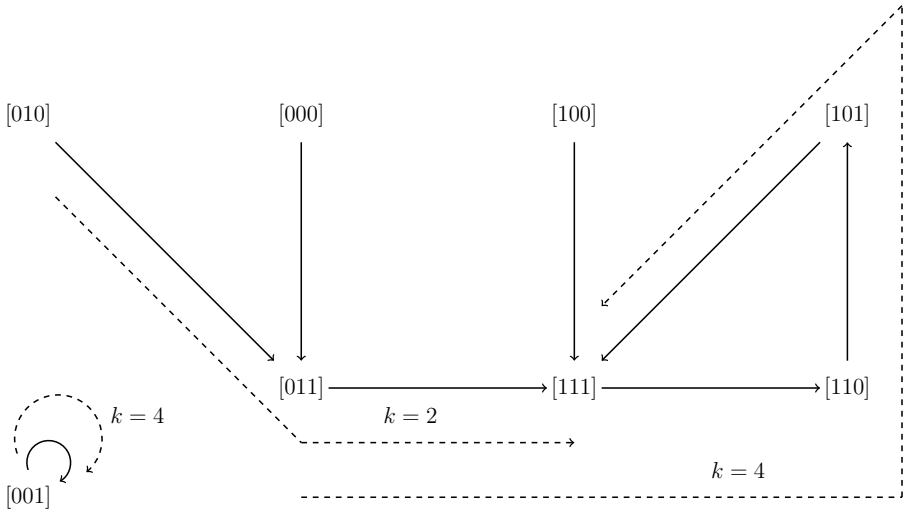

Figura 4 – Grafo de transições de estados para a rede Booleana r de 3 variáveis. São mostrados em linhas pontilhadas os caminhos que o algoritmo analisa até encontrar os atratores.

No caso em que k é muito menor do que o diâmetro máximo do GTE, pode-se cair em algum caminho da bacia de atração, mas quando k superar o diâmetro já não cairá na mesma bacia pois a Equação

3.4 restringe o `SAT-solver` para retornar caminhos com os estados salvos em $\mathcal{A}$. Na próxima iteração com $\widehat{S_4}$ o `SAT-solver` devolverá $Path = [001, 001, 001, 001, 001, 001]$ e guardará o *loop* dentro da lista $\mathcal{A}$. Finalmente, quando não houver mais caminhos, como é o caso deste exemplo, a iteração do `SAT-solver` devolverá insatisfazível mostrando todos os atratores em $\mathcal{A}$ e o algoritmo terá acabado. Pode-se ver as bacias de atração e as atribuições de $Path$ na Figura 4.

3.2 Um Método para Cálculo de Atratores Locais

A aplicação do algoritmo de Dubrova e Teslenko não é direta porque nesse tipo de modelo operam dois tipos de sinais em uma rede local:

1. Os *sinais localmente independentes* (SLI), que são aqueles não influenciados por qualquer outro sinal com origem na própria rede local; eles são os sinais externos, que correspondem aos sinais de acoplamento.

2. Os *sinais localmente regulados* (SLR), que são aqueles que dependem de ao menos um outro sinal com origem na própria rede local; eles são representados pelos valores das variáveis de estado da própria rede local.

Se excluirmos de uma rede local os seus SLIs, obteremos uma RDD "atômica" (uma rede única, não acoplada com qualquer outra). Portanto, redes locais são como RDDs às quais são acrescidos os sinais localmente independentes (os sinais de acoplamento). Daí que o uso do algoritmo de Dubrova e Teslenko implica na necessidade de desenvolvimento de um procedimento de "recorte", que permita a sua aplicação na computação dos atratores locais considerando a presença dos sinais localmente independentes. Essa adaptação (aplicada a toda rede local j) pode ser assim descrita:

1. Construímos uma extensão de j definida pelos seguintes procedimentos:

   - Incorporamos ao conjunto das variáveis de estado de j uma variável para cada um dos sinais de acoplamento que incidem sobre j;
   - Definimos para as variáveis de estado incorporadas funções de transição tipo identidade.

2. Aplicamos na rede local expandida o algoritmo de Dubrova e Teslenko.

Na Figura 5 ilustramos os tipos de sinais em uma rede local simples e o seu processo de expansão. É interessante notar que se assumirmos na entrada uma RDDA "simétrica" (padrão de conexão entre as RDDs representado por um grafo regular)e tal que todas as redes locais sejam idênticas, pode haver uma redução expressiva no esforço de computação, visto que bastaria uma única execução do algoritmo de identificação de atratores locais. Outras topologias representadas por grafos específicos com alguma regularidade poderiam ser exploradas a fim de diminuir os custos de computação.

As chamadas a um `SAT-solver` exigem que as RDDs sejam redes Booleanas; entretanto, a fim de dar alguma generalidade ao método, admitiremos entradas onde as RDDs são funções lógicas com domínio multi-valor. Por conta disso pretendemos desenvolver um procedimento de pré-processamento, baseado no trabalho de Dubrova, Teslenko e Ming (2010), que converterá funções lógicas multi-valor para Funções Booleanas. Desenvolvemos duas implementações para resolver o problema de cálculo de atratores locais: uma, baseada numa abordagem inovadora que usa um `HSP-solver` para encontrar caminhos válidos no GTE, a qual descrevemos na Seção 3.2.1 adiante; e outra, que usa um `SAT-solver` tradicional para encontrar caminhos válidos no GTE – como fez Dubrova e Teslenko – mas que inova ao considerar também a presença dos sinais de acoplamento, a qual descrevemos na seção 3.2.2.

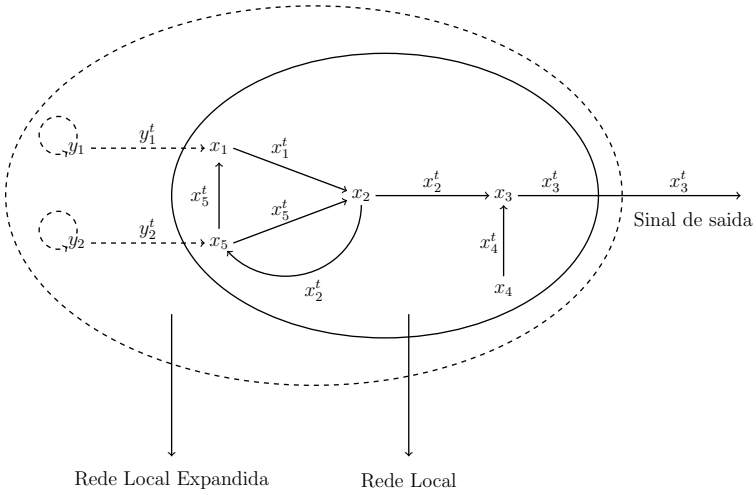

Figura 5 – Nesse exemplo de rede local, os sinais y_1^t, y_2^t são localmente independentes (os sinais de acoplamento) e os sinais x_1^t, x_2^t, x_3^t, x_4^t e x_5^t são localmente regulados. As variáveis x_1 e x_5 são as variáveis de entrada e a variável x_3 é variável de saída. Na rede local expandida é incorporada uma variável para cada sinal localmente independente e é definida uma função de transição do tipo identidade para cada uma.

3.2.1 Uma Abordagem Baseada em Hitting Set

Nesta seção nós apresentamos uma modificação do algoritmo de Dubrova e Teslenko descrito na seção 3.1, usando um `HSP-solver` no lugar do tradicional `SAT-solver` para a validação dos caminhos dentro do GTE. Esta abordagem requer a redução polinomial de SAT para HSP. Daí que desenvolvemos um procedimento de tempo linear para converter instâncias SAT em instâncias HSP, descrito na seção 3.2.1.2. Além disso, nós adaptamos o algoritmo paralelo baseado em *Graphics Processing Unit* (GPU) de Carastan-Santos et al. (2017) para trabalhar como um `SAT-solver`. Esse algoritmo paralelo é capaz de resolver instâncias do HSP com milhares (4000 ou 5000) de variáveis em tempo razoável (uma ou duas horas). É por esta razão que usamos inicialmente este algoritmo nesta investigação. Para dar um tratamento independente ao problema de detecção de atratores em redes Booleanas, que por si só é um problema importante, optamos nessa abordagem não considerar a presença dos sinais de acoplamento Y, isto é, na implementação dessa abordagem supomos que as RDDs estão "isoladas". Para considerar a presença dos sinais de acoplamento basta que realizemos uma adaptação de modo a incorporar tratamento descrito na seção 3.2.2.

3.2.1.1 Visão Geral

Dada uma rede Booleana r de n variáveis, suponha uma sequência de transições contendo $\mathcal{N}$ variáveis na sua formulação proposicional $\widehat{S_k}$, onde $\mathcal{N} = n \times (k+1)$. Nós convertemos $\widehat{S_k}$ em uma instância do HSP mediante a abordagem da seção 3.2.1.2. Se o `HSP-solver` devolve uma solução H tal que $|H| = \mathcal{N}$, significa que há uma atribuição satisfatória para as variáveis em $\widehat{S_k}$, e portanto $\widehat{S_k}$ é satisfazível e corresponde a um caminho válido no GTE (ver seção 3.1).

Se $\widehat{S_k}$ corresponde a uma sequência de transições de um caminho válido no GTE, então é verificado se $\widehat{S_k}$ contém um *loop*. Nesse caso, o *loop* é armazenado na lista de atratores $\mathcal{A}$, e a bacia de atração à qual o *loop* pertence é excluída do espaço de pesquisa; caso contrário, o tamanho k do $\widehat{S_k}$ é duplicado e o processo é repetido iterativamente. Aqui é apresentada uma descrição geral do algoritmo (mais detalhes são vistos na seção 3.2.1.3):

1. Gerar a formulação proposicional $\widehat{S_k}$ correspondente às sequências das transições;
2. Converter $\widehat{S_k}$ à Forma Normal Conjuntiva (CNF), denotada por $\mathcal{P}$;

3. Converter $\mathcal{P}$ em uma instância $\mathcal{S}$ do HSP;

4. Executar o `HSP-solver` para a instância $\mathcal{S}$ gerada no passo anterior:

   a) Se o `HSP-solver` retorna H tal que $|H| = \mathcal{N}$, onde $\mathcal{N} = n \times (k+1)$, checar se $\widehat{S_k}$ contém um *loop*, e:

      i. Se contém um *loop*, guardar o *loop* no conjunto solução $\mathcal{A}$, excluindo a bacia de atração do espaço de pesquisa e voltar ao passo 1;

      ii. caso contrário, duplicar k, $k \leftarrow k * 2$ e retornar ao passo 1.

   b) Se o `HSP-solver` retorna H de tal modo que $|H| > \mathcal{N}$, retorne a lista de atratores $\mathcal{A}$ e o algoritmo acaba.

3.2.1.2 Uma redução simples e linear do problema SAT para HSP

De acordo com o definido acima, dado um conjunto $\mathbb{X}$ e uma coleção $\mathcal{S}$ de subconjuntos de $\mathbb{X}$, o HSP consiste em encontrar um conjunto $H \subseteq \mathbb{X}$, tal que $H \cap S \neq \emptyset$ para todos $S \in \mathcal{S}$ e $|H|$ é mínimo. Assumimos que $\mathbb{X} = \cup_{S \in \mathcal{S}} S$, o que implica que um algoritmo que resolve HSP precisa apenas de $\mathcal{S}$ como argumento. Nós chamamos o algoritmo que resolve HSP de `HSP-solver`.

Seja $\mathcal{X}$ um conjunto ordenado de $\mathcal{N}$ variáveis. A seguir, denotamos os *literais* da i-ésima variável em $\mathcal{X}$ por x_i e $\neg x_i$. Uma *clausula* é uma disjunção de literais. Uma fórmula proposicional está na CNF se ela e uma conjunção de cláusulas.

Uma *interpretação* de $\mathcal{X}$ é um conjunto $L = \{L_1, \ldots, L_{\mathcal{N}}\}$, onde L_i é um literal de a i-ésima variável de $\mathcal{X}$, ou em outras palavras, L é um conjunto com exatamente um literal de cada variável em $\mathcal{X}$. A interpretação L *satisfaz* uma fórmula proposicional $\mathcal{P}$ na CNF se cada clausula em $\mathcal{P}$ tem um literal em L. O SAT consiste em decidir se existe uma interpretação L de $\mathcal{X}$ que satisfaça a fórmula proposicional $\mathcal{P}$ dada em CNF.

Nesta seção, mostramos o algoritmo `Convert-SAT-to-HSP` que transforma uma instância do problema SAT em uma instância do HSP no tempo $\Theta(N)$, onde N é o tamanho da instância do problema SAT. Dada a formula proposicional $\mathcal{P}$, `Convert-SAT-to-HSP` $(\mathcal{P})$ retorna a coleção

$$\mathcal{S} = \mathcal{C} \cup \mathcal{V},$$

onde $\mathcal{C}$ é um conjunto de literais de variáveis correspondentes às cláusulas de $\mathcal{P}$, $|\mathcal{C}| = m$, e $\mathcal{V}$ é um conjunto de todas as variáveis, por exemplo, $\mathcal{V} = \cup_i \{x_i, \neg x_i\}$, $|\mathcal{V}| = \mathcal{N}$. Assim, $\mathcal{N} + m$ conjuntos de $\mathcal{S}$ são dados como entrada para o `HSP-solver`. A Figura 6 mostra um exemplo de uma instância HSP obtida de uma instância SAT.

Em seguida, mostramos o seguinte resultado direto.

Teorema 1. *Seja $\mathcal{P}$ uma instância de SAT, `Convert-SAT-to-HSP`$(\mathcal{P}) = \mathcal{S}$, e `HSP-solver`$(\mathcal{S}) = H$. Então, $|H| = \mathcal{N}$ se e somente se existir uma interpretação de $\mathcal{X}$ que satisfaz $\mathcal{P}$. Além do `Convert-SAT-to-HSP` é um procedimento lineal.*

Demonstração. ($\Rightarrow$) Suponha que $|H| = \mathcal{N}$. Como `HSP-solver`$(\mathcal{S}) = H$, temos que $H \cap C \neq \emptyset$ para cada $C \in \mathcal{C}$ e $H \cap \{x_i, \neg x_i\} \neq \emptyset$ para cada i. Como $H \cap \{x_i, \neg x_i\} \neq \emptyset$ para cada i, $|H| = |\mathcal{X}| = \mathcal{N}$, temos que H tem exatamente um literal de cada variável em $\mathcal{X}$, o que implica que H é uma interpretação de $\mathcal{X}$. Além disso, como $H \cap C \neq \emptyset$ para cada $C \in \mathcal{C}$, temos que H satisfaz $\mathcal{P}$.

($\Leftarrow$) Por outro lado, suponha que H' é uma interpretação de $\mathcal{X}$ que satisfaz $\mathcal{P}$. Isto implica que $H' \cap \{x_i, \neg x_i\} \neq \emptyset$ para cada i e $H' \cap C \neq \emptyset$ para cada $C \in \mathcal{C}$. Além disso, o fato de H' ser uma interpretação que também implica que $|H'| = \mathcal{N}$. Portanto, como $|H|$ é mínimo, $|H| \leq |H'| = \mathcal{N}$. Por outro lado, como o `HSP-solver`$(\mathcal{S}) = H$ e existem $\mathcal{N}$ conjuntos disjuntos $\{x_i, \neg x_i\}$ para cada i, temos

que $|H| \geq \mathcal{N}$. como $|H| \leq \mathcal{N}$ e $|H| \geq \mathcal{N}$ de acordo com as últimas sentenças, concluímos que $|H| = \mathcal{N}$. Além disso, claramente `Convert-SAT-to-HSP` é um procedimento linear. □

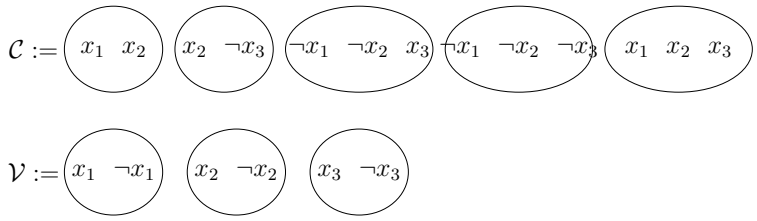

Figura 6 – Exemplo de conjuntos $\mathcal{C}$ e $\mathcal{V}$ cuja união é a instância do HSP de $\mathcal{P} = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$. Neste caso, observe que existe uma interpretação $(\neg x_1, x_2, x_3)$ satisfazendo $\mathcal{P}$ pois cada conjunto em $\mathcal{C} \cup \mathcal{V}$ tem pelo menos um literal de $\{\neg x_1, x_2, x_3\}$.

3.2.1.3 Implementação

Nesta seção fornecemos alguns detalhes adicionais do nosso método, que é descrito com mais precisão através do Algoritmo 1. Para testar esta abordagem usamos duas redes de exemplo de 3 e 4 variáveis, respetivamente . O exemplo de 3 variáveis foi descrito na seção 3.1 e descrevemos o exemplo de 4 variáveis aqui:

$$\begin{cases} x_1^{t+1} = (x_2^t \vee \neg x_3^t) \wedge (x_2^t \vee x_4^t), \\ x_2^{t+1} = (x_1^t \vee \neg x_2^t \vee \neg x_4^t) \wedge (\neg x_1^t \vee x_4^t), \\ x_3^{t+1} = (\neg x_1^t \vee \neg x_4^t), \\ x_4^{t+1} = (x_1^t \vee x_3^t \vee \neg - x_4^t) \wedge (x_1^t \vee x_4^t). \end{cases}$$

Como foi explicado anteriormente, é criada uma formulação proposicional $\widehat{S_2}$, que é depois convertida em $\mathcal{S}$ mediante o procedimento `Convert-SAT-to-HSP` da seção 3.2.1.2, que é por sua vez avaliada pelo `HSP-solver`. O Algoritmo de Carastan-Santos et al. (2017) foi modificado para devolver somente resultados H de tamanho $\mathcal{N}$, então se o `HSP-solver` retornar $H = NULL$, quer dizer que a formulação proposicional é *insatisfazível*. No caso que $|H| = \mathcal{N}$, quer dizer que temos um caminho válido e o H armazenado na estrutura $Path$ na forma de 0 e 1. É utilizado 0 se a variável tem o valor negativo e 1 no caso positivo. Uma atribuição inicial válida H seria

$$H = \{\neg x_1^0, x_2^0, x_3^0, \neg x_4^0, x_1^1, x_2^1, x_3^1, \neg x_4^1, x_1^2, \neg x_2^2, x_3^2, \neg x_4^2\}$$

e sua representação na estrutura de dados seria

$$Path = [0110, 1110, 1011].$$

Pode-se verificar que $Path$ não contém um atrator; sendo assim, dobra-se o tamanho de k e gera-se a formulação proposicional $\widehat{S_4}$. Depois reexecutamos o procedimento `Convert-SAT-to-HSP` e geramos uma nova atribuição $Path = [1111, 1101, 1101, 1101, 1101]$, a qual contém o atractor $[1101]$, que é representado em variáveis $(x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)$, e adicionado à lista de atratores $\mathcal{A}$. O processo é repetido iterativamente como é descrito na seção 3.1 até achar todos os atratores.

Para que as cláusulas Booleanas (instâncias SAT) possam ser convertidas em subconjuntos (instâncias HSP), deve-se usar o método `Convert-SAT-to-HSP` descrito na seção 3.2.1.2, mas também uma transformação adicional mediante um dicionário tem que ser feita, pois os programas `HSP-solver` (incluído o programa do Carastan-Santos et al. (2017)) encontrados só permitem o uso de números inteiros positivos

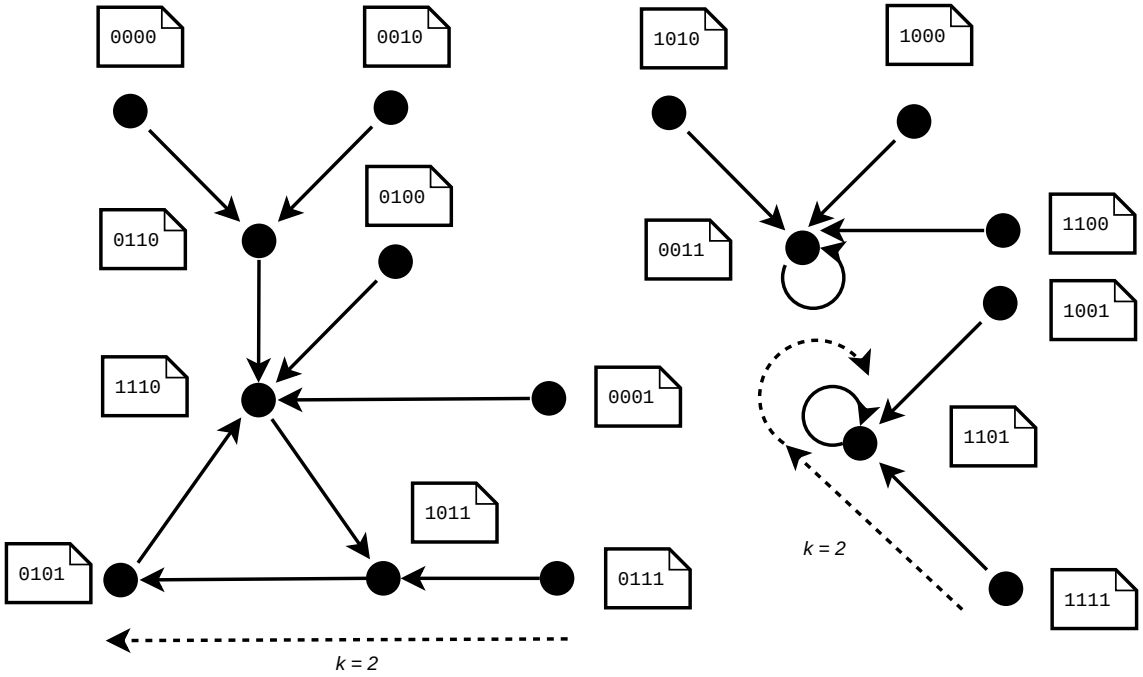

Figura 7 – Grafo de transição de estados para uma rede Booleana de 4 variáveis.

Z, então as variáveis x_i^t, junto com seus negativos $\neg x_i^t$ são convertidos em números positivos consecutivos. Quando é encontrada a resposta H, esta é convertida de volta a variáveis e analisada para encontrar o *loop* (atrator), que constitui o *conjunto solução* descrito na seção 3.1. O *conjunto solução* depois será transformado á formulação Booleana $\mathcal{B}$ e adicionado a $\mathcal{A}$ como foi descrito na Equação 3.3.

Um detalhe importante a respeito ao tamanho inicial de k na $\widehat{S_k}$, é que seu comprimento ideal, seria o valor do diâmetro do GTE, porque só podemos encontrar caminhos menores que seu diâmetro. Embora o cálculo do diâmetro de um grafo direcionado seja em geral um procedimento caro (PENNYCUFF; WENINGER, 2015), GTEs são grafos restritos porque o grau de saída de cada vértice é exatamente um, o que implica que, para esses grafos, é possível encontrar o diâmetro do gráfico em tempo linear usando a Busca em largura de grafos. No entanto, o número de vértices no GTE é exponencial ao número de variáveis, o que inviabiliza o uso desse procedimento.

Considerando as razões acima, nós seguimos Dubrova e Teslenko (2011), que adotam $k = n$ (linha 2 do Algoritmo 1) baseado em dados empíricos, onde n é o número de variáveis da rede Booleana. Em relação à maneira como aumentamos o tamanho das sequências de transições, fizemos isso para que, para cada iteração do algoritmo, seu tamanho fosse duplicado (linha 20 do Algoritmo 1). O tamanho das sequências de transição cresce para aproximadamente o diâmetro do gráfico, porque é esperado que quando k for maior que o diâmetro do gráfico, os atratores serão encontrados rapidamente.

Uma vantagem da pesquisa de atratores baseada em HSP proposta é que ela permite usar os algoritmos que são adaptados para Computação de Alto Desempenho (HPC em inglês) e computação CPU-GPU híbrida. Nesse sentido, adotamos o algoritmo HSP proposto por Carastan-Santos et al. (2017). Este algoritmo segue uma busca exaustiva de soluções H, na qual as soluções candidatas, que são codificadas como combinações de variáveis de $\mathbb{X}$ (Veja Seção 3.2.1.2).Então são enumeradas e avaliadas em aumento de ordem de cardinalidade $i, 1 \leq i \leq k$ e para quando uma solução é encontrada. O processo de avaliação é uma verificação de disjunção com as cláusulas em uma coleção classificada SortS, que é a coleção $\mathcal{S}$ cujas cláusulas são ordenadas em ordem crescente de seus tamanhos. A avaliação da solução candidata com SortS permite um descarte eficiente de candidatos que não sejam soluções. Esse processo pode ser eficientemente feito paralelamente por múltiplas unidades de processamento (CPUs e GPUs) por uma abordagem

1: r = Read from Boolean network description
2: $k = n$ /*n == Number of variables*/
3: $\widehat{S_k}$ = generatePropositionalFormulation (r,k)
4: H = `HSP-solver`($\mathcal{S}$)
5: *loop*Found = **false**
6: **while** $|H| == \mathcal{N}$ **do**
7: $Path$ = captureAssignment(H)
8: **for** i=0 **to** $k-1$ **do**
9: **if** $Path$[i] == $Path[k]$ **then**
10: B = transformToBooleanFormulation($Path[i+1,k]$)
11: $\mathcal{A} = \mathcal{A} \vee \mathcal{B}$
12: *loop*Found = **true**
13: **breakfor**
14: **end if**
15: **end for**
16: **if** *loop*Found **then**
17: $\widehat{S_k}$ = generatePropositionalFormulation (r,k)
18: $\widehat{S_k} = \widehat{S_k} \wedge \neg\mathcal{A}$ /*exclude from the search space the last computed attractor*/
19: **else**
20: $k = k * 2$ /*double the length (k) of the transitions sequence $\widehat{S_k}$ */
21: $\widehat{S_k}$ = generatePropositionalFormulation (r,k)
22: **end if**
23: H = `HSP-solver`($\mathcal{S}$)
24: *loop*Found = **false**
25: **end while**
26: **print** $\mathcal{A}$

Algorithm 1: Encontrar atratores usando *HSP-solver*

mestre-escravo, onde o mestre designa soluções candidatas aos escravos e busca independentemente por soluções Hitting Set entre seus candidatos designados.

Pode-se observar que nosso processo de redução SAT-HSP linear cria instâncias de HSP com a propriedade que suas soluções (se existirem) devem ter um tamanho k de pelo menos $\mathcal{N}$. Além disso, para encontrar atratores, é necessário apenas encontrar uma solução de uma dada instância, como contraposto a todas as soluções. Assim, exploramos essas características modificando o algoritmo Hitting Set de Carastan-Santos et al. (2017) de tal forma que:

- O algoritmo de busca começa com soluções candidatas com tamanho $k = \mathcal{N}$.
- Se um escravo encontra uma solução, ele sinaliza ao mestre que uma solução foi encontrada e retorna a solução para o mestre.
- Finalmente o mestre sinaliza a todos os outros escravos para finalizar seu processamento para que o algoritmo possa terminar.

3.2.2 Tratando Sinais de acoplamento em RDDs

Nesta seção nós apresentaremos uma abordagem baseada no Algoritmo de Dubrova e Teslenko para achar os atratores locais em RDDs que são parte de uma RDDA. Essas RDDs tem como caraterística o uso dos sinais de acoplamento que relacionam uma RDD com outra. Assumimos os sinais de acoplamento com valores fixos 0 ou 1, e para representar esta caraterística criamos uma RDD para cada possível combinação de valores dos sinais de acoplamento, o que gera o total de 2^2m RDDs. Também consideramos que cada sinal de acoplamento representa somente uma variável externa, de tal forma que estes sinais de acoplamento podem pertencer a distintas RDDs, podendo também, pertencer à mesma RDD. Adicionalmente, consideramos que as funções de acoplamento I estão inclusas dentro da descrição das funções de transição F das RDDs

na CNF, deixando a declaração explicita de I para tarefas de pre-processamento posteriores. Um ponto importante a considerar é que uma RDD com sinais de acoplamento fixos é equivalente a uma rede Booleana com variáveis de estado fixo, assim, um algoritmo que procura atratores em uma rede Booleana pode procurar atratores locais neste tipo de RDD. Além disso, algumas modificações tem que ser feitas dentro do algoritmo para trabalhar com RDDs que são partes de uma RDDA. A procura de atratores em RDDs é parte fundamental para poder achar os *Campos de atratores* da RDDA.

3.2.2.1 Visão Geral

O algoritmo para achar os atratores Locais em uma RDD utiliza como entrada as funções F e o número de sinais de acoplamento. Inicia-se criando uma RDD para cada combinação de valores dos sinais de acoplamento, os quais são tratados como variáveis de valor fixo dentro da RDD, e o algoritmo procura os atratores locais para cada uma delas. Estas variáveis farão parte da $\widehat{S_k}$ e tem que ser criadas para cada transição k; no caso da variável x_i com valor fixo 1, se criam as cláusulas com atribuição positiva ($x_i^{0..n}$), se o valor fixo for 0 se criam as cláusulas com atribuição negativa ($\neg x_i^{0..n}$). As cláusulas são adicionadas a $\widehat{S_k}$ com o operador lógico "$\wedge$". Isto faz que quando o `SAT-solver` esteja atribuindo valores para as variáveis, sempre escolha o valor atribuído para poder satisfazer a $\widehat{S_k}$, posteriormente seguir o processo descrito na seção 3.1, criando, novamente as cláusulas que representam as variáveis de valores fixos em cada iteração do algoritmo. O algoritmo pode ser descrito da seguinte forma (mais detalhes podem ser vistos na seção 3.2.2.2):

1. Gerar as combinações para o conjunto de sinais de acoplamento Y e repetir até processar todas as combinações;
   a) Gerar a formulação proposicional $\widehat{S_k}$ correspondente as sequências das transições, incluindo os sinais de acoplamento como variáveis fixas;
   b) Converter $\widehat{S_k}$ à Forma Normal Conjuntiva (CNF), representando como $\mathcal{P}$;
   c) Executar o `SAT-solver` em instância $\mathcal{P}$:
   d) Se o `SAT-solver` retorna a expressão como *satisfazível* os estados são armazenados no $Path$, e verficar se $\widehat{S_k}$ conter um *loop*:
      i. Se contém um *loop*, guardar o *loop* no conjunto $\mathcal{A}$, excluindo a bacia de atração do espaço de pesquisa, e voltar ao passo a);
      ii. caso contrário, duplicar $k \rightarrow 2k$ e voltar ao passo a).
   e) Se o `SAT-solver` retorna a expressão como *insatisfazível*, retornar-se a lista de atratores $\mathcal{A}$ e o algoritmo acaba.

3.2.2.2 Implementação

Para a implementação do Programa nós utilizamos a linguagem Python, uma biblioteca chamada Satispy, encarregada de transformar a formulação proposicional à CNF e chamar ao `SAT-solver`, neste caso nós usamos o MiniSAT de Eén e Sörensson (2003), o qual é muito usado para resolver Satisfazibilidade em Fórmulas Booleanas.

Para explicar a abordagem criamos 3 RDDs $r1, r2, r3$ como parte de uma mesma RDDA, cada uma delas com 5 variáveis locais as quais definimos: $\mathcal{X}_1 = \{x_1, x_2, x_3, x_4, x_5\}, \mathcal{X}_2 = \{x_6, x_7, x_8, x_9, x_{10}\}, \mathcal{X}_3 = \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$. E cada RDD com dois sinais de acoplamento y_1 e y_2, isto vai gerar 2^2 possíveis estados: $P(0,1) = \{00, 01, 10, 11\}$ em cada uma das três RDDs. Como o cálculo de atratores em todas as RDDs segue o mesmo procedimento, pegaremos como exemplo somente a rede $r1$ com os sinais de acoplamento de $y_1 = 0, y_2 = 1$ correspondentes as variáveis x_7, x_{11} da $r2, r3$ respetivamente. A continuação

descrevemos a continuação as funções F da $r1$ incluindo os sinais de acoplamento em forma de variáveis com valor fixo.

$$F^j \begin{cases} x_1^{t+1} = (x_{11}^t \vee \neg x_2^t) \wedge (\neg x_5^t \vee x_3^t \vee x_1^t \vee x_2^t), \wedge (\neg x_7^t \vee x_2^t \vee x_1^t) \\ x_2^{t+1} = (x_{11}^t \vee \neg x_1^t) \wedge (\neg x_7^t \vee \neg x_4^t) \\ x_3^{t+1} = (x_{11}^t \vee \neg x_1^t) \\ x_4^{t+1} = (\neg x_3^t \vee \neg x_5^t) \wedge (\neg x_5^t \vee \neg x_2^t \vee \neg x_{11}^t \vee \neg x_7^t) \wedge (\neg x_5^t \vee \neg x_3^t \vee \neg x_{11}^t) \\ x_5^{t+1} = (\neg x_3^t \vee \neg x_5^t) \wedge (x_5^t \vee \neg x_4^t) \end{cases}$$

De acordo com Algoritmo 2 começamos criando a $\widehat{S_2}$ e adicionamos as variáveis de valor fixo (sinais de acoplamento) para cada tempo t, o desenvolvimento da formulação proposicional ficaria assim:

$$\widehat{S_2} = \widehat{S_2} \wedge (\neg x_{11}^0) \wedge (x_6^0) \wedge (\neg x_{11}^1) \wedge (x_6^1) \wedge (\neg x_{11}^2) \wedge (x_6^2)$$

Em seguida a $\widehat{S_2}$ é transformada á CNF e depois $\mathcal{P}$ é avaliada pelo `SAT-solver`. Na primeira iteração do algoritmo para $\widehat{S_2}$ tem-se os valores: $Path = [0001001, 1111001, 0011001]$. Ao avaliar $Path$, vê-se que não tem um *loop*, então o valor do k é duplicado para $k = 4$. Nota-se que os dois últimos estados representam os sinais de acoplamento na forma de variáveis de estado fixo 01. Geramos a $\widehat{S_4}$ e adicionamos os valores das variáveis fixas, para depois converter na CNF como $\mathcal{P}$. Finalmente avalia-se com o `SAT-solver`, encontrando o novo $Path = [0001001, 1111001, 0011001, 1111001, 0011001]$, fazendo as análises de estados repetidos pode-se verificar que $Path_5 = Path_3$, então os estados $[0011001, 1111001]$ representam um atrator, o qual depois é adicionado á lista de atratores $\mathcal{A}$, na próxima iteração a lista de atratores é adicionada á $\widehat{S_6}$ seguindo a Equação 3.4, depois é avaliada novamente pelo `SAT-solver`. Este processo se repete até não achar todos os atratores, de acordo a seção 3.1.

Dado que o(s) estado(s) que compõe(m) um atrator é(são) devolvido(s) por o Programa Satispy no formato de 0's e 1's, precisamos – para efeitos de implementação do algoritmo – representá-los na forma de variáveis Booleanas e guardar-os em uma estrutura de dados $\mathcal{B}'$, se for 0 se agrega $\neg x_i$, se for 1 se agrega x_i, finalmente transforma-se o *conjunto solução* neste caso definido como $\mathcal{B}'$ em $\mathcal{B}$, de acordo a seção 3.1.

3.3 Um Método para Cálculo de Campos Atratores

Dada a coleção de conjuntos de atratores $\mathsf{A}^1, \ldots, \mathsf{A}^m$ (correspondentes às respectivas redes locais $1, \ldots, m$) calculada conforme descrito na Seção 3.2 e em suas subseções, o passo seguinte (o "passo 2") corresponde à computação dos campos atratores estáveis como combinações de atratores locais.

Para isso, um método enumerativo exigiria a computação de todas as possíveis atribuições a fim de identificar aquelas que satisfazem as condições de estabilidade (ver Seção 2.5), o que poderia ser computacionalmente muito custoso, dependendo do número m de redes locais e da cardinalidade dos conjuntos de atratores $\mathsf{A}^1, \ldots, \mathsf{A}^m$.

A fim de promover ganhos de desempenho, vislumbramos duas alternativas de investigação:

- explorar as propriedades do espectro das matrizes de adjacência e/ou incidência do grafo que representa o padrão de conexão entre as RDDs;
- explorar as propriedades da função de acoplamento a fim de promover "podas" que permitam alguma economia de computação.

O desenvolvimento desse "passo 2" é a próxima etapa do trabalho que pretendemos realizar. Ver cronograma na Seção 5.4.

1: r = Read from Boolean network description
2: $numberSignals$ = Read from Boolean network description
3: $listCombinations$ = generateCombinations($numberSignals$)
4: $listAtratorsCombinations$ = []
5: **while** y in $listCombinations$ **do**
6: $k = 2$
7: $\widehat{S_k}$ = generatePropositionalFormulation (r,k,y)
8: $Path$ = `SAT-solver`($\widehat{S_k}$)
9: $\mathcal{A}$ = [ ]
10: **while** $Path$!= $NULL$ **do**
11: **for** i=0 **to** $k-1$ **do**
12: **if** $Path$[i] == $Path[k]$ **then**
13: B = transformToBooleanFormulation($Path[i+1,k]$)
14: $\mathcal{A} = \mathcal{A} \vee \mathcal{B}$
15: *loop*Found = **true**
16: **breakfor**
17: **end if**
18: **end for**
19: **if** *loop*Found **then**
20: $\widehat{S_k}$ = generatePropositionalFormulation (r,k,y)
21: $\widehat{S_k} = \widehat{S_k} \wedge \neg\mathcal{A}$ /*exclude from the search space the last computed attractor*/
22: **else**
23: $k = k * 2$ /*double the length (k) of the transitions sequence $\widehat{S_k}$ */
24: $\widehat{S_k}$ = generatePropositionalFormulation (r,k,y)
25: **end if**
26: $Path$ = `SAT-solver`($\widehat{S_k}$)
27: **end while**
28: add($listAtratorsCombinations$,(y,$\mathcal{A}$))
29: **end while**

Algorithm 2: Encontrar atractores em RDD usando o *SAT-solver*

4 Resultados Preliminares

Os experimentos foram executados usando sistema operacional Ubuntu 18.04 em uma área de trabalho com processador Intel Core(TM) i7-3930K, processador de 3.17GHz, 12 CPUs, 32 GB de RAM e dois Nvidia GK 104 GeForce GTX boards.

4.1 Abordagem Baseada em HSP

A implementação desta abordagem foi feita no Python 2.7, utilizando a biblioteca Satispy a qual se encarga de deixar a formulação proposicional no CNF para posteriormente chamar o programa do Carastan-Santos et al. (2017) escrito em C++ que faz uso da GPU, este `HSP-solver` é usado para determinar para a avaliação dos caminhos determinando a satisfazibilidade, mediante o método descrito na seção 3.2.1.2. Foram feitos dois experimentos um 3 e outro de 4 variáveis descritos na seção 3.1 e 3.2.1.3 respetivamente, para o experimento com 3 variáveis acima, o tempo de execução foi de 3,55 segundos, mas para o experimento de rede de 4 variáveis, o tempo de execução foi de 9,15 horas. Esse crescimento no tempo de execução depende do `HSP-solver` (CARASTAN-SANTOS et al., 2017), levando em conta algumas experiências anteriores com a mesma entrada e hardware adotados aqui, o `HSP-solver` pode ser otimizado em versões futuras, adaptando-o para resolver eficientemente o problema de Satisfazibilidade Booleana.

4.2 Abordagem Baseada em SAT

A implementação desta abordagem foi feita no Python 2.7, fazendo uso da biblioteca Satispy a qual se encarga de deixar a formulação proposicional na CNF, e depois internamente chama ao Minisat de Eén e Sörensson (2003) feito em C++ como *SAT-solver* para a avaliação dos caminhos determinando a *Satisfazibilidade* mediante o abordagem da seção 3.2. Para criar as RDDs como parte de uma RDDA, foi feito um gerador de redes Booleanas que tem como parâmetros: quantidade de redes, número de variáveis por rede e número de sinais de acoplamento. O gerador cria as RDDs em arquivos de texto, que tem como valores: número de rede, variáveis locais, sinais de acoplamento e as funções F que determinam o estado de cada variável. O gerador foi configurado para que as RDDs tenham número aleatório de cláusulas nas funções F, este número está no intervalo de 1 a 4, também foi usando um número aleatório para o número de variáveis por clausula, o qual esta no intervalo de 2 a 4. Primeiro testamos o programa com 10 RDDAs com número de RDDs variando de 50 até 500 RDDs, com o número de variáveis igual a 5 e um sinal de acoplamento ambos valores iguais para todas as RDDs. Dá para verificar que o tempo de execução cresceu de forma linear de acordo ao número de redes quando estas tem o mesmo número de variáveis e sinais de acoplamento. Os dados deste teste se encontram na tabela 1

Depois testamos 10 RDDAs de 50 RDDs cada uma, variando o número de variáveis de 3 a 30 e com um sinal de acoplamento. Pode-se observar um crescimento exponencial, que é esperado de acordo a complexidade descrita no artigo de Dubrova, Teslenko e Ming (2010). Os dados deste teste se encontram tabela 2.

Finalmente fizemos um teste mexendo o número de sinais de acoplamento de 1 à 10, com 10 RDDAs de 2 RDDs cada uma e com o número de variáveis de tamanho 15 para as duas RDDs. Neste teste pode-se verificar também um crescimento exponencial, pois os sinais de acoplamento dentro do algoritmo são convertidos em variáveis de valor fixo como foi explicado na seção 3.2.2. Em termos de complexidade, ao

adicionar sinais de acoplamento somente esta se adicionando quantidade de variáveis á RDD. Os dados deste teste se encontram tabela 3.

número de redes	**Variáveis locais**	**Sinais de acoplamento**	**Chamadas ao Programa**	**Variáveis Totais**	**Tempo (seg)**
50	5	1	100	6	5,23
100	5	1	200	6	10,45
150	5	1	300	6	15,75
200	5	1	400	6	19,82
250	5	1	500	6	25,39
300	5	1	600	6	31,83
350	5	1	700	6	37,13
400	5	1	800	6	43,66
450	5	1	900	6	46,24
500	5	1	1000	6	52,93

Tabela 1 – Dados experimentais com redes entre 50 e 500, com número de variáveis fixo em 5 e número de sinais de acoplamento em 1.

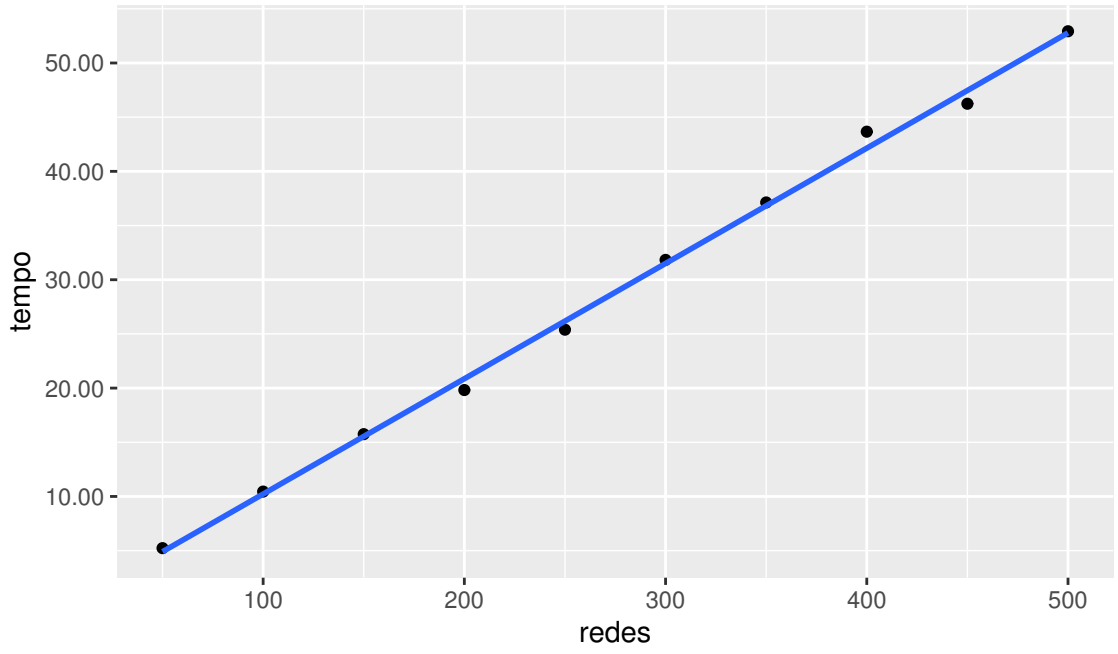

Figura 8 – Gráfico da tabela 1

número de redes	Variáveis locais	Sinais de acoplamento	Chamadas ao Programa	Variáveis Totais	Tempo (seg)
50	3	1	100	4	2,72
50	5	1	100	6	5,66
50	8	1	100	9	12,57
50	10	1	100	11	17,88
50	13	1	100	14	33,37
50	15	1	100	16	46,43
50	18	1	100	19	95,21
50	20	1	100	21	116,60
50	23	1	100	24	282,91
50	25	1	100	26	436,70

Tabela 2 – Dados experimentais com 50 redes, com número de variáveis entre 3 e 25 e número de sinais de acoplamento em 1.

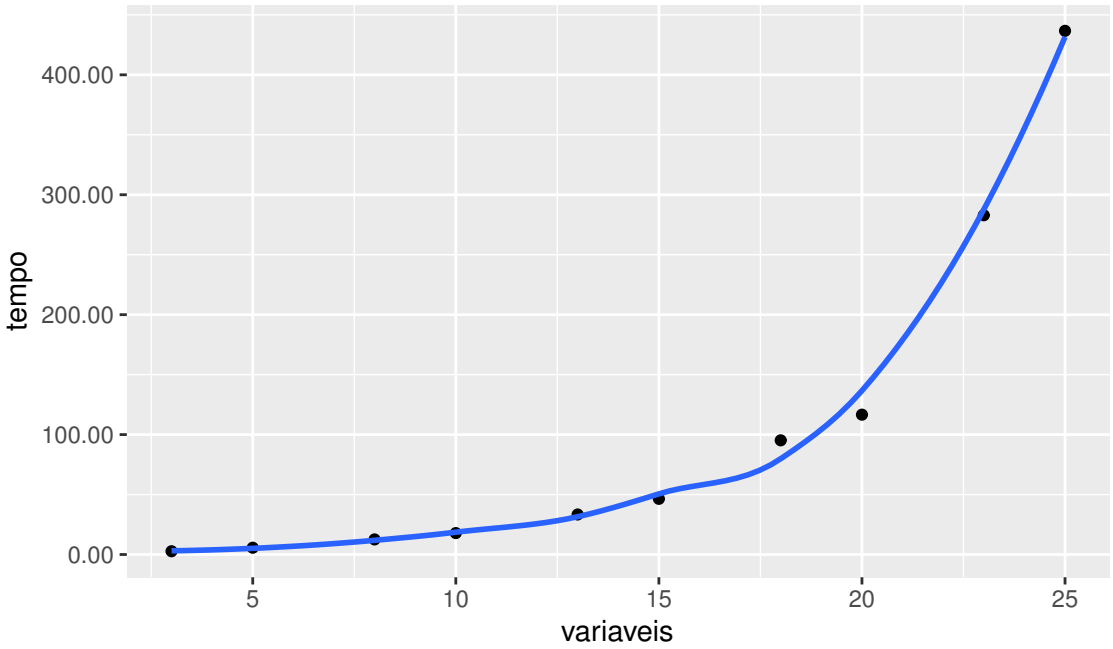

Figura 9 – Gráfico da tabela 2

número de redes	Variáveis locais	Sinais de acoplamento	Chamadas ao Programa	Variáveis Totais	Tempo (seg)
2	15	1	4	16	3,40
2	15	2	8	17	7,35
2	15	3	16	18	7,89
2	15	4	32	19	11,70
2	15	5	64	20	24,85
2	15	6	128	21	56,52
2	15	7	256	22	60,94
2	15	8	512	23	160,20
2	15	9	1024	24	227,31
2	15	10	2048	25	421,96

Tabela 3 – Dados experimentais com 2 redes, com número de variáveis fixo em 15 e o número de sinais de acoplamento fixo entre 1 e 10.

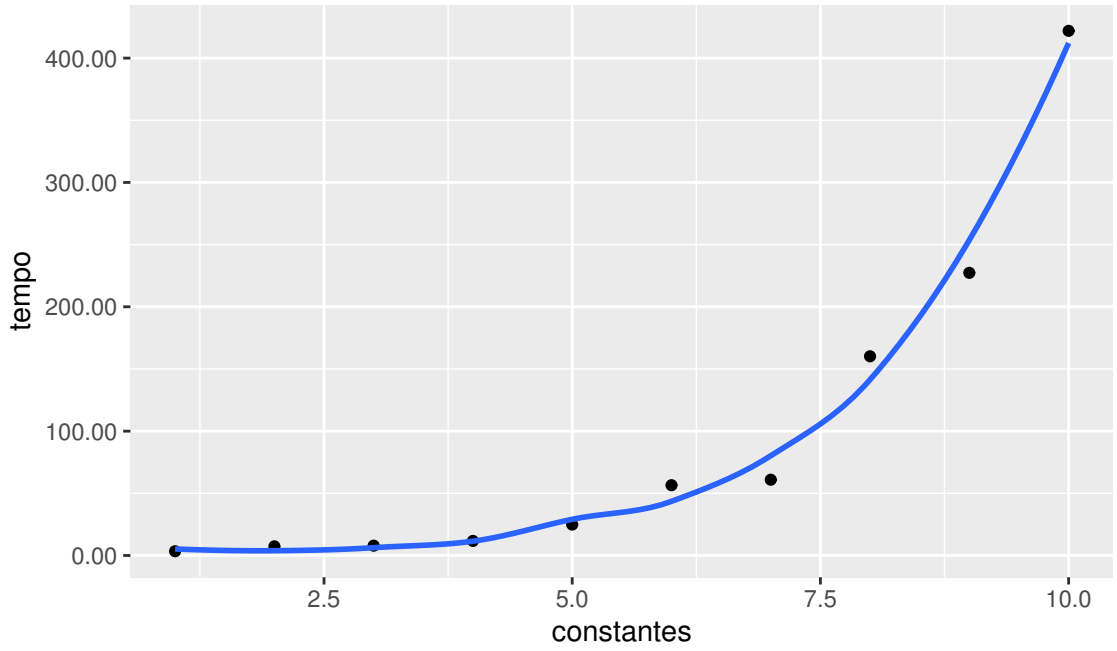

Figura 10 – Gráfico da tabela 3

5 Conclusões Parciais

5.1 Contribuições Preliminares

Conforme mencionamos na Introdução, as contribuições preliminares obtidas nesse trabalho são as seguintes:

1. Desenvolvemos uma modelagem de sistemas de entidades interagentes por redes dinâmicas discretas acopladas (RDDAs).

2. Desenvolvemos e implementamos uma abordagem inovadora para o problema de detecção de atratores em redes Booleanas, sendo que para isso tivemos que:

   - adaptar o algoritmo de Dubrova e Teslenko, de modo a combiná-lo com o algoritmo de Carastan-Santos *et al*, que é um algoritmo de alto-desempenho baseado na arquitetura GPU/CUDA e originalmente desenvolvido para o problema *Hitting Set*;
   - desenvolver uma redução [SAT —> *Hitting Set*], que é uma contribuição de natureza mais teórica original, a fim de podermos usar o algoritmo de Carastan-Santos *et al* como um SAT-*solver*.

3. Implementamos a abordagem original (baseada em SAT "clássico") proposta por Dubrova e Teslenko; porém, adaptada de modo a considerar a presença de sinais externos, que é uma característica necessária à aplicação no contexto de RDDAs.

5.2 Limitações e Desafios

Quando concebemos a estratégia de usar o algoritmo de Carastan-Santos *et al* como um SAT-*solver*, a ideia era que pudéssemos explorar as suas características de alto-desempenho de modo a resolver eficientemente o problema de detecção de atratores locais nas RDDs. Entretanto, apesar dessa característica, os resultados experimentais – do ponto de vista de desempenho – não foram como esperado devido ao fato de a redução [SAT —> *Hitting Set*] levar a instâncias do *Hitting Set* de pior caso para o algoritmo de Carastan-Santos *et al*. Essa é a principal limitação da opção de cálculo de atratores locais ("passo 1" do nosso método) via *Hitting Set*, e sua superação corresponde ao principal desafio dessa abordagem.

5.3 Próximos Passos

Para romper a limitação a que nos referimos na Seção anterior, vislumbramos como possível caminho a remodelagem da estratégia para o passo de identificação de caminhos válidos no GTE, de modo a "enquadrá-lo" em instâncias do *Hitting Set* que não são de pior caso para algoritmo de Carastan-Santos *et al*, o que permitiria usá-lo sem a necessidade de promover maiores modificações estruturais.

De toda forma, devido a restrições de prazo, optamos adotar como prioridade o desenvolvimento do "passo 2" (computação dos campos atratores estáveis como combinações de atratores locais), a fim concluir o desenvolvimento de uma primeira versão "inteira" do método a que nos propusemos, isto é, um método que, dada uma RDDA como entrada, seja capaz de responder se ela contém ou não campos atratores estáveis e, em caso positivo, seja capaz de identificá-los.

Com o propósito de buscar melhorias de desempenho, o desenvolvimento dessa versão "inteira" inclui a implementação de uma versão distribuída do passo para computação dos atratores locais.

5.4 Cronograma de Trabalho

A fim de concluir os próximos passos mencionados na Seção anterior, o desenvolvimento do trabalho a partir daqui dar-se-á através da execução das seguintes etapas:

Etapa 1: Projeto do algoritmo para computação dos campos atratores estáveis.

Etapa 2: Implementação do algoritmo para computação dos campos atratores estáveis como combinações de atratores locais.

Etapa 3: Implementação distribuída do algoritmo para detecção dos atratores locais.

Etapa 4: Execução dos experimentos com o código integrado ("passo 1" e "passo 2") e análise dos resultados.

Etapa 5: Redação e submissão de artigo.

Etapa 6: Redação da dissertação e defesa.

Cronograma de Execução

Atividades/Mês	Etapa 1	Etapa 2	Etapa 3	Etapa 4	Etapa 5	Etapa 6
07/2019	X	X	X			
08/2019		X	X	X		
09/2019				X	X	
10/2019					X	X
11/2019						X

Referências

ARENAS, A. et al. Synchronization in complex networks. *Physics Reports*, v. 469, p. 93–153, 2008.

BENíTEZ, M. et al. Interlinked nonlinear subnetworks underlie the formation of robust cellular patterns in arabidopsis epidermis: a dynamic spatial model. *BMC Systems Biology*, v. 2, n. 98, p. 1752–0509, 2008.

BERNTENIS, N.; EBELING, M. Detection of attractors of large boolean networks via exhaustive enumeration of appropriate subspaces of the state space. *BMC bioinformatics*, BioMed Central, v. 14, n. 1, p. 361, 2013.

BOCCALETTI, S. et al. Detecting complex network modularity by dynamical clustering. *Physical Review E*, n. 75, p. 045102, 2007.

BRYANT, R. E. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, IEEE, v. 100, n. 8, p. 677–691, 1986.

BURCH, J. R. et al. Symbolic model checking: 1020 states and beyond. *Information and computation*, Elsevier, v. 98, n. 2, p. 142–170, 1992.

CARA, A. D. et al. Dynamic simulation of regulatory networks using squad. *BMC bioinformatics*, BioMed Central, v. 8, n. 1, p. 462, 2007.

CARASTAN-SANTOS, D. et al. Finding exact hitting set solutions for systems biology applications using heterogeneous gpu clusters. *Future Generation Computer Systems*, Elsevier, v. 67, p. 418–429, 2017.

CHEN, H.; LIANG, J.; LU, J. Partial synchronization of interconnected boolean networks. *IEEE Transactions on Cybernetics*, v. 47, n. 1, p. 258–266, 2017.

CHENG, D. Semi-tensor product of matrices and its applications-A survey. In: *Proceeding of ICCM*. [S.l.: s.n.], 2007. v. 3, p. 641–668.

CHENG, D.; QI, H. A linear representation of dynamics of boolean networks. *IEEE Transactions on Automatic Control*, IEEE, v. 55, n. 10, p. 2251–2258, 2010.

COOK, S. A. The complexity of theorem-proving procedures. In: ACM. *Proceedings of the third annual ACM symposium on Theory of computing*. [S.l.], 1971. p. 151–158.

DINUR, I.; SAFRA, S. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, JSTOR, p. 439–485, 2005.

DUBROVA, E.; TESLENKO, M. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 8, n. 5, p. 1393–1399, 2011.

DUBROVA, E.; TESLENKO, M.; MING, L. Finding attractors in synchronous multiple-valued networks using sat-based bounded model checking. In: *40th IEEE International Symposium on Multiple-Valued Logic*. [S.l.: s.n.], 2010. p. 144–149.

EÉN, N.; SÖRENSSON, N. An extensible sat-solver. In: SPRINGER. *International conference on theory and applications of satisfiability testing*. [S.l.], 2003. p. 502–518.

GUO, W. et al. A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks. *PLoS ONE*, v. 9, n. e94258, 2014.

HONG, Y.; SCAGLIONE, A. Distributed change detection in large scale sensor networks through the synchronization of the pulse-coupled oscillators. In: *IEEE ICASSP 2004*. [S.l.: s.n.], 2004. p. 869–872.

HUANG, S.; ERNBERG, I.; KAUFFMAN, S. Cancer attractors: A systems view of tumors from a gene network dynamics and developmental perspective. *Seminars in Cell & Developmental Biology*, v. 20, p. 869–876, 2009.

JALAN, S.; AMRITKAR, R. E. Self-organized and driven phase synchronization in coupled maps. *Physical Review Letters*, v. 90, p. 014101, 2003.

JOHNSON, D. B. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, SIAM, v. 4, n. 1, p. 77–84, 1975.

JONG, H. D.; GEISELMANN, J.; HERNANDEZ, C. Genetic network analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics*, Oxford University Press, v. 19, n. 3, p. 336–344, 2003.

KARL, S.; DANDEKAR, T. Jimena: efficient computing and system state identification for genetic regulatory networks. *BMC bioinformatics*, BioMed Central, v. 14, n. 1, p. 306, 2013.

KAUFFMAN, S. A. *The Origins of Order*. New York: Oxford University Press, 1993.

KLAMT, S.; SAEZ-RODRIGUEZ, J.; GILLES, E. E. Structural and functional analysis of cellular networks with cellnetanalyzer. *BMC systems biology*, BioMed Central, v. 1, n. 1, p. 2, 2007.

KURAMOTO, Y. Self-entrainment of a population of coupled non-linear oscillators. In: *International Symposium on Mathematical Problems in Theoretical Physics*. [S.l.: s.n.], 1975. v. 39, p. 420.

LI, F.; LU, X. Complete synchronization of temporal boolean networks. *Neural Networks*, v. 44, p. 72–77, 2013.

LI, R.; CHU, T. Synchronization in an array of coupled boolean networks. *Physics Letters A*, v. 376, p. 3071–3075, 2012.

LäMMER, S. et al. Decentralised control of material or traffic flows in networks using phase-synchronisation. *Physica A*, v. 363, n. 1, p. 39–47, 2006.

MIYANO, T.; TSUTSUI, T. Data synchronization in a network of coupled phase oscillators. *Physical Review Letters*, v. 98, p. 024102, 2007.

MOTTER, A. E. Bounding network spectra for network design. *New Journal of Physics*, v. 9, n. 6, p. 182, 2007.

MOTTER, A. E.; ZHOU, C. S.; KURTHS, J. Enhancing complex-network synchronization. *Europhysics Letters*, v. 69, n. 3, p. 334–340, 2005.

NISHIKAWA, T.; MOLNAR, F.; MOTTER, A. E. Stability landscape of power-grid synchronization. In: *4th IFAC Conference on Analysis and Control of Chaotic Systems*. [S.l.: s.n.], 2015. v. 48, p. 1–6.

NISHIKAWA, T. et al. Heterogeneity in oscillator networks: Are smaller worlds easier to synchronize? *Physical Review Letters*, v. 91, p. 014101, 2003.

PECORA, L. M.; CARROLL, T. L. Master stability functions for synchronized coupled systems. *Physical Review Letters*, v. 80, p. 2109, 1998.

PENNYCUFF, C.; WENINGER, T. Fast, exact graph diameter computation with vertex programming. In: BARCELONA SUPERCOMPUTING CENTER. *1st High Performance Graph Mining workshop, Sydney, 10 August 2015*. [S.l.], 2015.

PLUCHINO, A.; LATORA, V.; RAPISARDA, A. Changing opinions in a changing world: A new perspective in sociophysics. *International Journal of Modern Physics C*, World Scientific, v. 16, n. 04, p. 515–531, 2005.

SCHAUB, M. T. et al. Graph partitions and cluster synchronization in networks of oscillators. *Chaos*, v. 26, p. 094821, 2016.

SEYBOTH, G. S. et al. On robust synchronization of heterogeneous linear multi-agent systems with static couplings. *Automatica*, v. 53, p. 392–399, 2015.

WU, Y. et al. Adaptive output synchronization of heterogeneous network with an uncertain leader. *Automatica*, v. 76, p. 183–192, 2017.

WUENSCHE, A. Genome regulation modeled as a network with basins of attraction. *Complexity*, v. 4, n. 3, p. 47–66, 1998.

WUENSCHE, A. Basins of attraction in network dynamics: A conceptual framework for biomolecular networks. In: MODULARITY IN DEVELOPMENT AND EVOLUTION. Chicago: G.Schlosser and G.P.Wagner, 2004. p. 288–311.

XIANG, J.; LI, Y.; HILL, D. J. Cooperative output regulation of linear multi-agent network systems with dynamic edges. *Automatica*, v. 77, p. 1–13, 2017.

YANG, Y.; NISHIKAWA, T.; MOTTER, A. E. Small vulnerable sets determine large network cascades in power grids. *Science*, v. 358, n. 6365, p. eaan3184, 2017.

ZHAO, Y.; KIM, J.; FILIPPONE, M. Aggregation algorithm towards large-scale boolean network analysis. *IEEE Transactions on Automatic Control*, IEEE, v. 58, n. 8, p. 1976–1985, 2013.