



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

Carlos Reynaldo Portocarrero Tovar

**Computação de Campos Atratores em Redes Dinâmicas Discretas
Acopladas**

Santo André - SP
Maio de 2020

Carlos Reynaldo Portocarrero Tovar

Computação de Campos Atratores em Redes Dinâmicas Discretas Acopladas

Texto de Defesa 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
Maio de 2020

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Portocarrero Tovar, Carlos Reynaldo
Computação de Campos Atratores em Redes Dinâmicas Discretas
Acopladas / Carlos Reynaldo Portocarrero Tovar. — 2020.

84 fls.

Orientador: Luiz Carlos da Silva Rozante

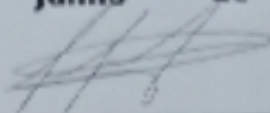
Tese (Doutorado) — Universidade Federal do ABC, Programa de Pós-Graduação em Ciência da Computação, Santo André, 2020.

1. Redes Dinâmicas Discretas Acopladas. 2. Estabilidade. 3. Campos Atratores. 4. Satisfatibilidade. 5. Problema da Transversal Mínima. I. Silva Rozante, Luiz Carlos da. II. Programa de Pós-Graduação em Ciência da Computação, 2020. III. Título.

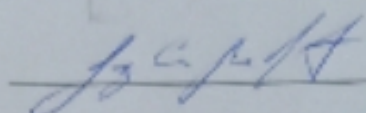
Este exemplar foi revisado e alterado em relação à versão original, de acordo com as observações levantadas pela banca no dia da defesa, sob responsabilidade única do(a) autor(a) e com a anuência do(a) orientador(a).

15 de Junho de 2020

Assinatura do(a) autor(a):



Assinatura do(a) orientador(a):



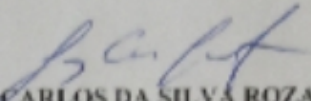


SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas
UFABC - Fundação Universidade Federal do ABC
Programa de Pós-Graduação em Ciência da Computação
CNPJ nº 07.722.779/0001-06
Av. dos Estados, 5001 - Bairro Santa Teresinha - Santo André - SP - Brasil
poscomp@ufabc.edu.br

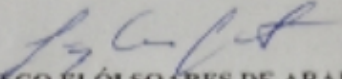


No dia 27 de Maio de 2020 às 14:00, por participação remota, realizou-se a Defesa da Dissertação de Mestrado, que constou da apresentação do trabalho intitulado "Computação de Campos Atratores em Redes Dinâmicas Discretas Acopladas" de autoria do candidato, CARLOS REYNALDO PORTOCARRERO TOVAR, RA nº 21201810221, discente do Programa de Pós-Graduação em CIÊNCIA DA COMPUTAÇÃO da UFABC. Concluídos os trabalhos de apresentação e arguição, o candidato foi considerado APROVADO pela Banca Examinadora.

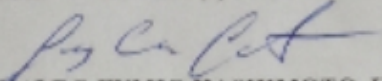
E, para constar, foi lavrada a presente ata, que vai assinada pelos membros da Banca.


Dr. LUIZ CARLOS DA SILVA ROZANTE, UFABC

Presidente - Interno ao Programa


Dr. FRANCISCO ELÓI SOARES DE ARAÚJO, UFMS

Membro Titular - Examinador(a) Externo à Instituição


Dr. RONALDO FUMIO HASHIMOTO, USP

Membro Titular - Examinador(a) Externo à Instituição

Dr. DAVID CORREA MARTINS JUNIOR, UFABC

Membro Suplente - Examinador(a) Interno ao Programa

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

Agradecimentos

Primeiramente agradeço ao meus pais Gladys Vilma Tovar Acuna e Reinaldo Crisólogo Portocarrero Rossel – os quais também dedico este trabalho – por me ajudar ao longo da minha vida e realizar meus sonhos. Ao meu orientador, Professor Doutor Luiz Carlos da Silva Rozante, pela atenção dada a mim e pelas excelentes observações referentes aos meus erros e acertos durante o trabalho. Também agradeço a: Lucas Lima Rodrigues, Fernando Montoya Cubas, Rodrigo Cesar Bonini, Luis Felipe Mello e muitos outros amigos por sua ajuda e apoio durante o mestrado. Por fim, à Universidade Federal do ABC e à CAPES, por prover suporte financeiro durante o desenvolvimento deste trabalho.

"A gota perfurou a pedra, não por sua força, mas por sua constância."

Ovidio

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 envolvendo essa classe específica de modelos, em particular estudos baseados em abordagens computacionais. Campos atratores em RDDAs consistem numa classe restrita de estados globalmente estáveis do sistema, nos quais a dinâmica de toda entidade interagente permanece “localmente confinada” – para todo tempo – num mesmo atrator local. O objetivo desse projeto consiste em desenvolver um método computacionalmente eficiente que, dada uma RDDA como entrada, seja capaz de responder se ela contém ou não campos atratores, bem como identificá-los.

Palavras-chave: Redes Dinâmicas Discretas Acopladas, Estabilidade, Campos Atratores, Satisfatibilidade, 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 involving this particular class of models, in particular studies based on computational approaches. Attractor fields in RDDAs consist of a restricted class of globally stable system states, in which the dynamics of every interacting entity remains “locally confined ” - for every time - in the same local attractor. 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 attractor fields, as well as to identify them.

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

Sumário

1	INTRODUÇÃO	25
1.1	Objetivos	25
1.2	Metodologia	26
1.3	Justificativa	26
1.4	Contribuições	27
1.5	Organização do Trabalho	28
2	DEFINIÇÕES PRELIMINARES, NOTAÇÃO E EXEMPLOS DE APLICAÇÕES	29
2.1	Redes Dinâmicas Discretas (RDDs)	29
2.2	Grafo de Transições de Estados (GTE)	29
2.3	Computação de Atratores em Redes Dinâmicas Discretas	30
2.4	Redes Dinâmicas Discretas Acopladas (RDDAs)	32
2.5	Sincronização e Estabilidade em RDDAs	35
2.6	Problema de Satisfatibilidade Booleana	36
2.7	Forma Normal Conjuntiva	36
2.8	<i>Hitting Set Problem</i>	37
2.9	Problema da Cobertura de Vértices (PCV)	37
2.10	Aplicações	37
3	CÁLCULO DE ATRADORES EM RDDS	39
3.1	O Algoritmo de Dubrova e Teslenko	39
3.2	Uma Nova Abordagem Baseada no HSP	42
3.2.1	Visão Geral	42
3.2.2	Uma Redução Simples e Linear do Problema SAT para HSP	44
3.2.3	Implementação	45
3.2.4	Experimentos da Abordagem Baseada em HSP	47
3.3	Um Método para Cálculo de Atratores em Redes Locais	47
3.3.1	Tratando Sinais de Acoplamento em RDDs	49
3.3.1.1	Visão Geral	49
3.3.1.2	Implementação	50
3.3.1.3	Experimentos	51
4	CONSIDERAÇÕES PARA O CÁLCULO DE CAMPOS ATRADORES EM RDDAS	57
4.1	Método Enumerativo	57
4.2	Método com Podas	58
4.2.1	Visão Geral	58
4.2.2	Cálculo dos Pares Estáveis de Atratores	60
4.2.3	Implementação do Cálculo dos Pares Estáveis de Atratores	65
4.3	Gerador de RDDAs	69
5	CONCLUSÕES FINAIS	73
5.1	Principais Contribuições	73
5.2	Limitações e Desafios	73
5.3	Trabalhos Futuros	73

5.4	Artigos Publicados	74
	ANEXO A – ARTIGO BIBE2019	75
	REFERÊNCIAS	81

Lista de ilustrações

Figura 1	– Exemplo de um gráfico de transição de estado de uma rede booleana de 3 variáveis. Neste gráfico são observados dois atratores, um atrator de ponto fixo formado pelo estado $\{001\}$ e outro atrator periódico de três estados formado por $\{111, 101, 110\}$	30
Figura 2	– (a): Parte de uma órbita $O(x)$ em um espaço de estados X , onde $(F^j)^2(x)$ é o <i>sucessor</i> de $(F^j)^1(x)$ e x é a <i>pré-imagem</i> (ou <i>predecessor</i>) 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 z é um exemplo de estado do tipo <i>jardim do Éden</i> . (e): Exemplo de <i>bacia de atração</i>	31
Figura 3	– Exemplo de uma RDDA de 6 RDDs diferentes. Nas RDDs os vértices internos representam as variáveis e as arestas internas representam as relações entre elas. Por sua vez, se cada RDD é tomada como um vértice, elas formam uma RDDA e as setas entre as RDDs representam o sinal de acoplamento que se vincula de uma RDD para outra.	33
Figura 4	– Uma representação gráfica da forma pela qual as RDDs u, \dots, r (que são convizinhas de j) exercem influência sobre a RDD j	34
Figura 5	– 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 $A_1 = [(1, a_1), (2, a_3), (3, a_{10})]$ contém o atrator a_1 , oriundo da RDD 1; o atrator a_3 oriundo da RDD 2 e o atrator a_{10} oriundo da RDD 3. A atribuição $A_2 = [(1, a_4), (2, a_6), (3, a_{11})]$ contém o atrator a_4 , oriundo da RDD 1; o atrator a_6 oriundo da RDD 2 e o atrator a_{11} oriundo da RDD 3.	35
Figura 6	– Grafo de transições de estados para uma rede Booleana r de 3 variáveis. São mostrados em linhas pontilhadas os caminhos que o algoritmo analisa até encontrar os atratores.	41
Figura 7	– Fluxograma representando as etapas do algoritmo para encontrar atratores com base no algoritmo Dubrova e Teslenko usando HSP para resolver a satisfazibilidade.	43
Figura 8	– 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 $\{\neg x_1, x_2, x_3\}$	44
Figura 9	– GTE para a rede Booleana de 4 variáveis descrita na Seção 3.2.3, o GTE possui três atratores, dois deles são atratores pontuais e o outro é um atrator periódico composto por três estados.	47
Figura 10	– 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.	48
Figura 11	– Gráfico correspondente ao primeiro experimento em que a relação entre o número de redes e o tempo de execução do algoritmo é vista, quanto mais redes, maior o tempo de execução, mas esse crescimento é apenas linear, os dados com os quais esse gráfico gerou eles estão na Tabela 1	53
Figura 12	– Gráfico correspondente ao segundo experimento em que a relação entre o número de variáveis e o tempo de execução do algoritmo é observada, com maior número de variáveis na RDD maior tempo de execução, mas esse crescimento é exponencial, o que concorda com a complexidade apresentada em artigo de Dubrova e Teslenko (DUBROVA; TESLENKO, 2011), os dados com os quais foi gerado este gráfico estão na tabela 2.	54

Figura 13 – Gráfico correspondente ao terceiro experimento em que a relação entre o número de sinais de acoplamento e o tempo de execução do algoritmo é vista, quanto maior o número de sinais de acoplamento por rede, maior o tempo de execução, mas esse crescimento é exponencial, o que concorda com a complexidade apresentada no artigo de Dubrova e Teslenko (DUBROVA; TESLENKO, 2011), como os sinais de acoplamento aumentam o número de variáveis na RDD, os dados com os quais eu gero esse gráfico estão na tabela 3.	55
Figura 14 – (a): à esquerda, um exemplo de RDDA com três RDDs, sendo que nessa representação cada uma delas está associada a uma cor; (b): ao centro, uma representação da coleção de conjuntos de atratores locais obtida no “passo um”: o conjunto \mathcal{A}^1 contendo os atratores locais associados à RDD 1 (em vermelho), o conjunto \mathcal{A}^2 contendo os atratores locais associados à RDD 2 (em verde) e o conjunto \mathcal{A}^3 contendo os atratores locais associados à RDD 3 (em azul); (c): à direita, uma representação do grafo G' , sendo que cada vértice desse grafo corresponde a um atrator local e cada aresta representa um par estável de atratores, isto é, se existe uma aresta direcionada $a_2^1 \rightarrow a_3^2$ em G' , significa que a_2^1 e a_3^2 satisfazem a condição de estabilidade.	59
Figura 15 – (a): à esquerda, um grafo G' na entrada da etapa de montagem dos campos atratores; (b): à direita, uma representação de um conjunto solução S , com destaque para dois campos atratores: um, acima, contendo os atratores a_2^1, a_4^2 e a_7^3 ; outro, abaixo, contendo os atratores a_2^1, a_4^2 e a_6^3	60
Figura 16 – Gráfico mostrando a relação da informação na condição de estabilidade, a) a_z^j com o e_1 que o compõe e os valores das variáveis, b) definição do valor do y_u^j quando o a_z^j é calculado $\phi(y_u^j, a_z^j) = 1$, c) tabela verdade relacionada ao y_u^j , primeiro selecionando as linhas com valor 1 e depois os valores das variáveis de saída, d) a_w^u do qual os valores pertencentes às S_u^j são selecionados; se esses valores forem iguais a alguma combinação de valores selecionados na tabela verdade, a condição de compatibilidade será atendida.	61
Figura 17 – Fluxograma do método do Cálculo dos pares estáveis de atratores.	62
Figura 18 – Grafo G' de uma RDDA composta por 3 RDDs j, u, m . Os vértices foram coloridos de acordo com a RDD à qual o atrator representado pertence: vermelho para u , verde para j e azul para m . Um subgrafo representando um campo de atrator estável é composto de linhas tracejadas.	65
Figura 19 – Representação topológica por meio de um gráfico do teste feito com RDD formada por 3 RDDs, incluindo conexões entre as RDDs.	69
Figura 20 – Grafo de conexões compatíveis pertencente ao teste da RDDA de 3 RDDs e 5 variáveis, os nomes dos vértices foram trocados por números para melhorar a visualização.	69
Figura 21 – Representação topológica do exemplo de uma RDDA de 3 RDDs e 5 variáveis cada, incluindo conexões entre as RDDs.	71

Lista de tabelas

Tabela 1	– Dados experimentais de uma RDDA com um intervalo de 50 a 500 RDDs, 5 variáveis e um sinal de acoplamento por RDD.	53
Tabela 2	– Dados experimentais de uma RDDA com 50 RDDs cada RDDA, com um número de variáveis na faixa de 3 a 25 e um sinal de acoplamento por RDD.	53
Tabela 3	– Dados experimentais de uma RDDA com 2 RDDs cada RDDA, com um número de variáveis de 15 e uma quantidade de sinais de acoplamento no intervalo de 1 ate 10 por RDD.	54
Tabela 4	– Tabela verdade que tem como entradas os valores das variáveis de saída \mathcal{S}^j e como fórmula a função de acoplamento h_u^j	63
Tabela 5	– Tabela verdade com os valores das variáveis de saída \mathcal{S}^u correspondentes com uma sinal de acoplamento y_u^j com valor "1".	63
Tabela 6	– Tabela verdade que tem como termos os valores das variáveis de saída \mathcal{S}_2^1	66

Lista de abreviaturas e siglas

SP	São Paulo
ABC	Região de: Santo André, São Bernardo Do Campo, São Caetano Do Sul
UFABC	Universidade Federal do ABC
CAPES	Coordenação De Aperfeiçoamento De Pessoal De Nível Superior
RDDA	Rede Dinâmica Discreta Acoplada
RDDAs	Redes Dinâmicas Discretas Acopladas
RDD	Rede Dinâmica Discreta
RDDs	Redes Dinâmicas Discretas
STP	<i>Semi-Tensor Product</i>
SAT	Problema De Satisfatibilidade Booleana
HSP	Hitting Set Problem
SDD	Sistema Dinâmico Discreto
SDF	Sistema Dinâmico Discreto Finito
RB	Rede Booleana
GTE	Grafo de Transição de Estados
BDD	<i>Binary Decision Diagram</i>
NP	<i>Nondeterministic Polynomial Time</i>
CNF	<i>Conjunctive Normal Form</i>
3CNF	<i>CNF</i> conformado por 3 literais por cláusula
SAT- <i>solver</i>	Programa para resolver a Satisfatibilidade Booleana
SAT- solver	SAT- <i>solver</i> de Eén e Sörensson (2003) chamado MiniSAT.
SLI	Sinais Localmente Independentes
SLR	Sinais Localmente Regulados
HSP- <i>solver</i>	Programa Para Resolver o HSP
HSP- solver	HSP- <i>solver</i> de Carastan-Santos et al. (2017)
POO	Programação Orientada a Objetos
PCV	Problema da Cobertura de Vértices Mínima

Lista de símbolos

A	Atribuição, Campo atrator estável
\mathcal{A}	Lista de atratores de uma RDD
a	Atrator
B	Atrator na forma de expressão booleana
C	Conjunto de cláusulas de \mathcal{P}
\mathcal{C}	Conjunto de literais de variáveis
\mathcal{E}	Conjunto de variáveis de entrada
E	Arestas
F	Conjunto de Funções
f	Função, Mapeamento
G	Grafo
\mathcal{H}	Coleção de conjuntos resposta no HSP
h	Função de acoplamento
H	Conjunto resposta do HSP
H'	Interpretação de \mathcal{X}
I	Conjunto de Funções de Acoplamento
k	Quantidade de transições
L_i	Literal dai-ésima variável
L	Conjunto de Literais
LC	Número máximo de literais em C
m	Quantidade de RDDs por RDDA
n	Quantidade de variáveis por RDD
\mathcal{N}	Numero de elemento que deve ter a resposta do HSP-solver
$O()$	Complexidade Computacional
\mathcal{P}	Formulação proposicional na CNF
$Path$	Estrutura de dados que contem os estados de uma atribuição de S_k
P	Conjunto de estados possíveis
r	Rede
\mathcal{S}	Coleção de Subconjuntos de \mathbb{X} , entrada do HSP

S	Elemento de \mathbb{S}
S	Estado n -dimensional na sincronização de saída
\mathcal{S}	Conjunto de Variáveis de Saída
\widehat{S}_k	Formulação proposicional
S_k	Sequência de transições de estados
t	Valor do tempo
V	Vértices
\mathcal{V}	Conjunto de todas as variáveis
$\mathcal{X}^{\text{RDDA}}$	Conjunto de variáveis de toda a RDDA
X	Estado n -dimensional na sincronização completa
x^t	Vector de variáveis de estado no tempo t
\mathbb{X}	Conjunto Finito no HSP
x_i^t	Valor da variável x_i no tempo t
x_i	Variável da RDD
x_i	Variável
x	Estado, Conjunto de valores das variáveis
\times	Estado do atrator, Conjunto de valores das variáveis
\mathcal{X}	Conjunto de variáveis
X	Espaço de estados
Y	Conjunto de Sinais de Acoplamento
y	Sinal de acoplamento
\rightarrow	Transição
$\ $	Cardinal de um conjunto
$\Theta(N)$	Complexidade Computacional Media

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 (RDDs) – denominados *redes dinâmicas discretas acopladas* (RDDAs) –, em particular as redes Booleanas acopladas, apresentam um amplo leque de potenciais aplicações em áreas que vão da biologia à física. Apesar de sua reconhecida aplicabilidade, existem relativamente poucos estudos envolvendo essa classe específica de modelos. Apenas recentemente foram desenvolvidos estudos envolvendo estabilidade e sincronização em redes Booleanas acopladas, enfatizando trabalhos baseados em produto semi-tensorial (STP, do Inglês *semi-tensor product*) de matrizes (CHENG, 2007).

Campos atratores em RDDAs consistem numa classe restrita de estados globalmente estáveis do sistema, nos quais a dinâmica de toda entidade interagente permanece “localmente confinada” – para todo tempo – num mesmo atrator local. Em outras palavras, um campo atrator é uma configuração globalmente estável de estados do sistema tal de que cada entidade interagente entra num atrator local e nele permanece indefinidamente, embora os estados das entidades interagentes possam variar localmente porque admitem-se atratores locais periódicos.

No entanto, para fins de definição de escopo desse trabalho, restringimos a computação apenas aos campos atratores que proporcionam sinais de acoplamento (sinais inter-RDDs) do tipo fixo, isto é, assumimos uma restrição no conjunto solução de modo a descartar aqueles campos atratores que geram sinais de acoplamento periódicos (não fixos), reduzindo assim o espaço de busca.

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 (ver definição na Seção 2.5) e, em caso positivo, seja capaz de identificá-los.

Grosso modo, um campo atrator 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 tempo – no atrator definido pela atribuição.

Quando rotulamos um método como “eficiente”, nos referimos 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.

Outro objetivo implícito nesse trabalho é o desenvolvimento de implementações e experimentos a fim de realizar provas de conceitos e avaliar-se o desempenho dessa abordagem.

1.2 Metodologia

A fim de alcançar esses objetivos foi adotada uma estratégia em dois passos: num primeiro momento, o método calcula os atratores de todas as redes locais; num segundo momento, com os atratores de cada RDD em mãos, construímos atribuições estáveis (campos atratores) como combinações de atratores locais que satisfazem algumas propriedades relacionadas à estabilidade do sistema.

Para o primeiro passo, foram desenvolvidas duas abordagens:

- Primeiramente, foi realizada uma adaptação do algoritmo de [Dubrova e Teslenko \(2011\)](#) baseado no Problema de Satisfatibilidade Booleana (SAT), 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 *Hitting Set Problem* (HSP). Isso implicou num desafio subjacente, cuja superação por si só representa um objetivo implícito: como um dos requerimentos do algoritmo de [Dubrova e Teslenko \(2011\)](#) é avaliação de sentenças lógicas, teve-se que fazer a redução [*SAT* \rightarrow *HSP*] a fim de poder-se usar o algoritmo de [Carastan-Santos et al. \(2017\)](#) como um SAT-*solver*.
- Em seguida, para fins de comparação e avaliação com a abordagem acima, foi implementada a abordagem original (baseada em SAT “classico”) proposta por [Dubrova e Teslenko \(2011\)](#); porém, foi adaptada de modo a considerar a presença de sinais externos, que é uma característica necessária à aplicação no contexto de RDDAs.

Relativamente ao segundo passo, foram desenvolvidos:

- Um método para encontrar pares estáveis de atratores que pertencem a diferentes RDDs. Esse método representa uma “etapa 1” no cálculo de campos atratores.
- Uma proposta de possível solução para computar os campos atratores a partir dos pares estáveis obtidos na etapa anterior. Aqui modelamos o problema como um problema em grafos.
- Um gerador de RDDAs, que atende as características das RDDAs descritas na Seção 2.4 e recebe como parâmetros: número de redes, número de variáveis por rede, número de conexões máximas entre redes, número máximo de variáveis de saída e número máximo de cláusulas por função. O objetivo do gerador é dar agilidade e suporte aos experimentos.

1.3 Justificativa

Devido à 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-tensoresial (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 de campos atratores, como a que propomos no presente trabalho.

1.4 Contribuições

As contribuições deste trabalho são:

1. Foi desenvolvida uma modelagem de sistemas de entidades interagentes por redes dinâmicas discretas acopladas (RDDAs).
2. Foi desenvolvida e implementada uma abordagem inovadora para o problema de detecção de atratores em redes Booleanas baseada no HSP, sendo que para isso teve-se 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 HSP;
 - desenvolver uma redução [$SAT \rightarrow HSP$] a fim de poder-se usar o algoritmo de Carastan-Santos *et al* como um SAT-*solver*.
3. Foi implementada 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.
4. Foi criado e implementado um gerador de RDDAs. Isso foi necessário porque atualmente não há nenhum disponível para manipulação de RDDAs.
5. Foi desenvolvido e implementado um método para encontrar computar pares estáveis de atratores com origem em diferentes RDDs. Esses pares são usados para gerar um grafo nos quais os campos atratores estão representados na forma de subgrafos.

6. Foi descrita uma possível solução para computar os campos atratores (subgrafos) a partir do grafo gerado na etapa anterior (pares de atratores estáveis).

1.5 Organização do Trabalho

Os seguintes capítulos estão organizados da seguinte forma: no Capítulo 2 e em algumas de suas seções são introduzidas as principais definições, notação e uma formulação mais precisa do problema em questão, conceitos necessários para entender a pesquisa, bem como alguns exemplos de aplicações de RDDAs; no Capítulo 3 e em suas seções descrevemos um método para cálculo de atratores em RDDs, aqui também descrevemos o Algoritmo de [Dubrova e Teslenko \(2011\)](#); no Capítulo 4 é descrito um método para computar pares estáveis de atratores das RDDs que compõem a RDDA, também descrevemos uma proposta para computação de campos atratores a partir dos pares estáveis de atratores ; no Capítulo 5 e em suas seções tecemos as considerações finais.

2 Definições Preliminares, Notação e Exemplos de Aplicações

Aqui são introduzidos os conceitos e definições fundamentais à descrição do trabalho, bem como nomenclaturas e notações. Além disso, ao final do capítulo, são mencionadas algumas potenciais aplicações.

2.1 Redes Dinâmicas Discretas (RDDs)

Um *sistema dinâmico discreto* (SDD) consiste em um espaço de estados X e um mapeamento $f : X \rightarrow X$ 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 $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ que assumem valores num espaço de estados finito X^n e sejam as funções f_1, f_2, \dots, f_n tal que a função $f_i : X^n \rightarrow X$ determina o próximo estado da variável x_i , onde $0 \leq i \leq n$.

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

Denomina-se um sistema j desse tipo de **Rede Dinâmica Discreta** (RDD) (WUENSCHKE, 1998), o qual é denotado por (\mathcal{X}, X^n, F^j) .

Se as funções em F^j são aplicadas simultaneamente, diz-se 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-se 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:

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

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 $F^j(x) = y$, onde $x, y \in V$. Assim, cada aplicação de F^j , que corresponde a uma *transição* de estado da RDD, é representada por uma aresta no GTE. Um exemplo da aplicação sucessiva de F é apresentada no item a) da Figura 2.

Dado um espaço de estados X^n e um conjunto estados X , com $X \subseteq X^n$, se existe algum $k \geq 1$, tal que $(F^j)^k(x) = x$ para algum $x \in X$, então dizemos que X é um *atrator*, sendo que:

- se $k = 1$, isto é, se $(F^j)^1(x) = x$, para $x \in X$, então diz-se que $X = \{x\}$ é um *atrator pontual* (ou *ponto fixo* ou *estado estacionário*);

- se $k > 1$, isto é, se existe algum $k \geq 2$, tal que $(F^j)^k(x) = x$, para $x \in X$, então diz-se que $X = \{x, (F^j)^1(x), (F^j)^2(x), \dots, (F^j)^k(x)\}$ é *atrator periódico* (ou *período/ciclo atrator*), onde $(F^j)^i(x)$, $1 \leq i \leq k$, indica a i -ésima aplicação de F^j a partir do estado x .

Mais intuitivamente falando, um atrator corresponde a um ciclo no GTE, isto é, um conjunto de estados que repetem-se quando aplicamos as funções de transição indefinidamente. Um exemplo de GTE com dois atratores é apresentado na Figura 1. O item c) da Figura 2 ilustra um atrator pontual e outro atrator periódico de 4 estados.

O conjunto de todos os atratores $\{a_1, a_2, a_3, \dots, a_n\}$ associados à uma RDD j é denotado por \mathcal{A}^j , sejam eles atratores pontuais ou periódicos.

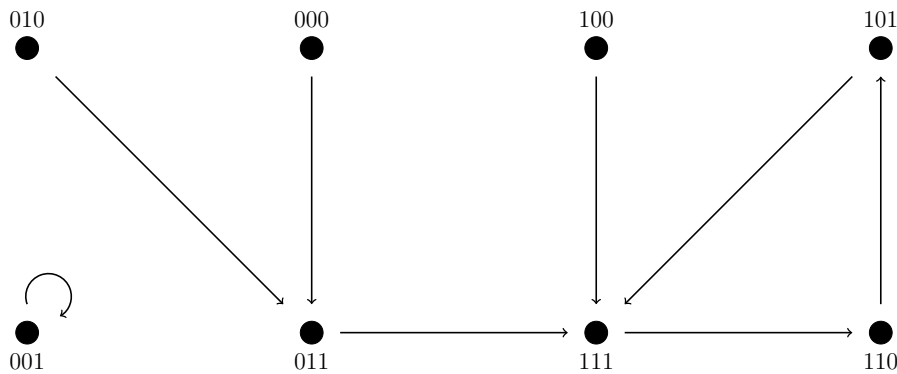


Figura 1 – Exemplo de um gráfico de transição de estado de uma rede booleana de 3 variáveis. Neste gráfico são observados dois atratores, um atrator de ponto fixo formado pelo estado $\{001\}$ e outro atrator periódico de três estados formado por $\{111, 101, 110\}$.

Dado um estado $x \in X^n$, considera-se 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 que o número total de pré-imagens de um estado é o seu *grau de entrada*, no item b) da Figura 2 mostra que o estado y tem 3 pré-imagens. Os estados que são pré-imagens podem ter suas próprias pré-imagens e assim por diante. Um exemplo disso é apresentado no item a) onde $F(x)$ é pré-imagem de $F^2(x)$ e, por sua vez, x é pré-imagem de $F(x)$. Se um estado não tem pré-imagens ele é dito ser um estado *jardim do Éden*. Um exemplo de um estado do tipo *jardim do Éden* é apresentado no item d) da Figura 2 no estado z .

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*. O conjunto de partes transientes que atingem um atrator são chamadas de *árvores transientes*.

Considerando-se 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 *bacia de atração* sem as árvores transientes, ou seja, apenas com os estados do atrator. Um exemplo de uma bacia de atração é apresentado no item e) da Figura 2, um exemplo de sem árvores transitórias é mostrado na Figura 1 no atrator de ponto formado pelo estado 001. Agora considerando-se 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.

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 prever estruturas (atratores, bacias de atração, etc) de sua dinâmica?

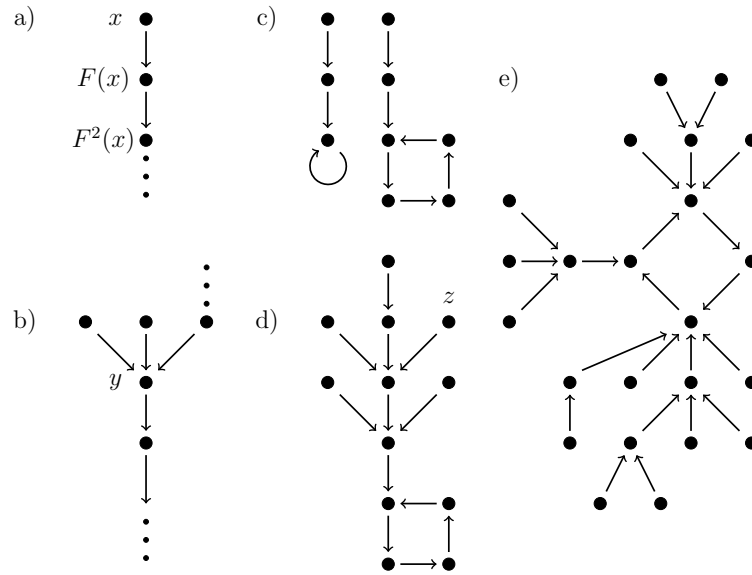


Figura 2 – (a): Parte de uma órbita $O(x)$ em um espaço de estados X , onde $(F^j)^2(x)$ é o *sucessor* de $(F^j)^1(x)$ e x é a *pré-imagem* (ou *predecessor*) de $(F^j)^1(x)$. (b): O estado y tem grau de entrada 3. (c): Exemplo de atrator pontual (à esquerda) e periódico (à direita); (d): Exemplo de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

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 heurísticamente 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 é possível identificar estruturas (por exemplo atratores) do GTE.

vi) Métodos baseados na formulação do problema em termos do 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 GTE, 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 GTE. Esse processo é repetido iterativamente para valores cada vez maiores de p até que todos os atratores sejam identificados.

O presente 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 3.3).

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 a qual chamamos de *rede local*. Este sistema se assemelha a uma “rede de redes”, onde podemos distinguir três elementos constituintes:

1. O padrão de conexão – a estrutura da rede – entre as entidades dinâmicas, representado por um grafo dirigido G (Ver Figura 3);
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 deste trabalho assumiremos que as RDDAs em estudo são tal que:

- i) Os grafos que definem a estrutura da RDDA (o padrão de conexão) são arbitrários.
- ii) As RDDs podem ser não idênticas². Porém, assume-se que o número de variáveis de estado em todas elas é o mesmo e, por facilidade de notação, considera-se que todas têm o mesmo padrão de nomeação, ou seja, todas as RDDs têm n variáveis de estado nomeadas x_1, \dots, x_n . Vale salientar que embora diferentes RDDs tenham o mesmo número de variáveis e estas sejam igualmente nomeadas, suas

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. Contudo, 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 2.10)

² Diz-se que as redes são idênticas se possuírem o mesmo número de variáveis e as funções de transição foram as mesmas. No caso de serem idênticas, há impacto no custo de computação, visto que bastaria calcular os atratores locais de uma única RDD.

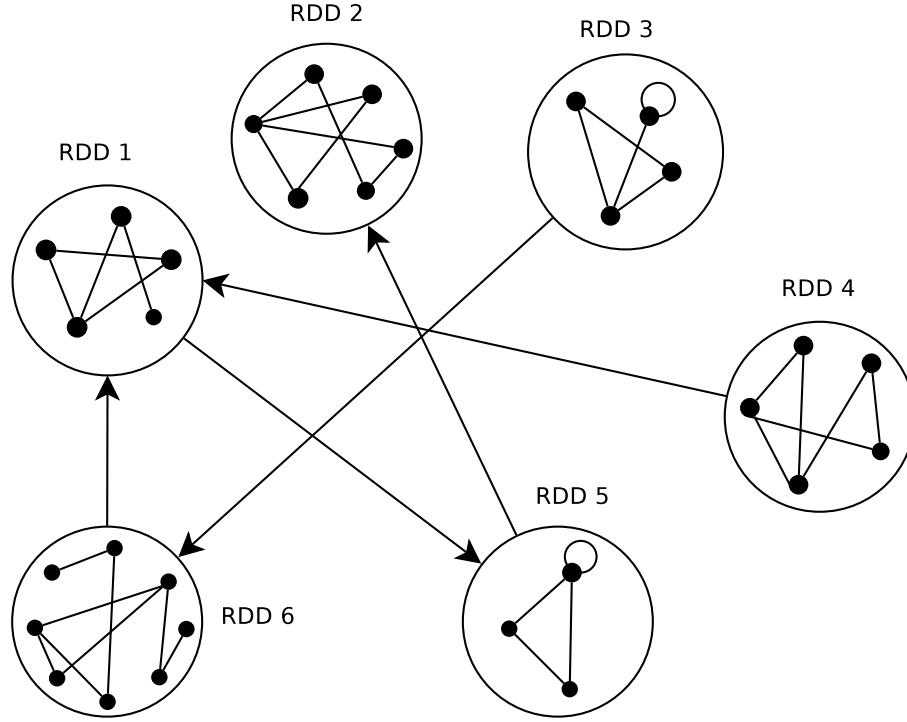


Figura 3 – Exemplo de uma RDDA de 6 RDDs diferentes. Nas RDDs os vértices internos representam as variáveis e as arestas internas representam as relações entre elas. Por sua vez, se cada RDD é tomada como um vértice, elas formam uma RDDA e as setas entre as RDDs representam o sinal de acoplamento que se vincula de uma RDD para outra.

funções de transição F^j , $1 \leq j \leq m$, são diferentes. Além disso, assume-se também que as RDDs são todas redes Booleanas³ 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 ou algumas RDDs vizinhas, então existe um subconjunto $\mathcal{S}_u \subseteq \mathcal{X}^u$ de variáveis de estado cujos valores (estados) são utilizados indiretamente pelas RDDs vizinhas. Denominam-se as variáveis em \mathcal{S}_u de *variáveis de saída* de u .

Se u é convizinha de j , então existe um subconjunto $\mathcal{S}_u^j \subseteq \mathcal{S}_u$, pois assume-se que u emite (a partir de \mathcal{S}_u^j) um sinal y_u^j que serve ao acoplamento com j , através de uma função $h_u^j : \mathcal{X}^u \rightarrow \mathcal{X}^j$. Denomina-se h_u^j de *função de acoplamento* e y_u^j de *sinal de acoplamento*. Denotamos por $I^j = [h_1^j, \dots, h_m^j]$ o *conjunto de funções de acoplamento* e $Y^j = [y_1^j, \dots, y_m^j]$ como o *conjunto de sinais de acoplamento* que influem na RDD j .

Se a RDD 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). As variáveis contidas em \mathcal{E}^j são chamadas de *variáveis de entrada* de j . A Fig. 4 contém um diagrama ilustrativo dessa modelagem.

³ Assumimos que as RDD são redes Booleanas por facilidade de análise. No entanto, para fins de implementação e simulação, seria relativamente fácil admitir-se entradas do problema que sejam RDDs com variáveis e funções multi-valor, visto que para isso bastaria desenvolver um procedimento de pré-processamento que converta funções lógicas multi-valor em funções Booleanas, o que pode ser adaptado do artigo de [Dubrova e Teslenko \(2011\)](#). Este recurso pode ser desenvolvido em trabalhos futuros.

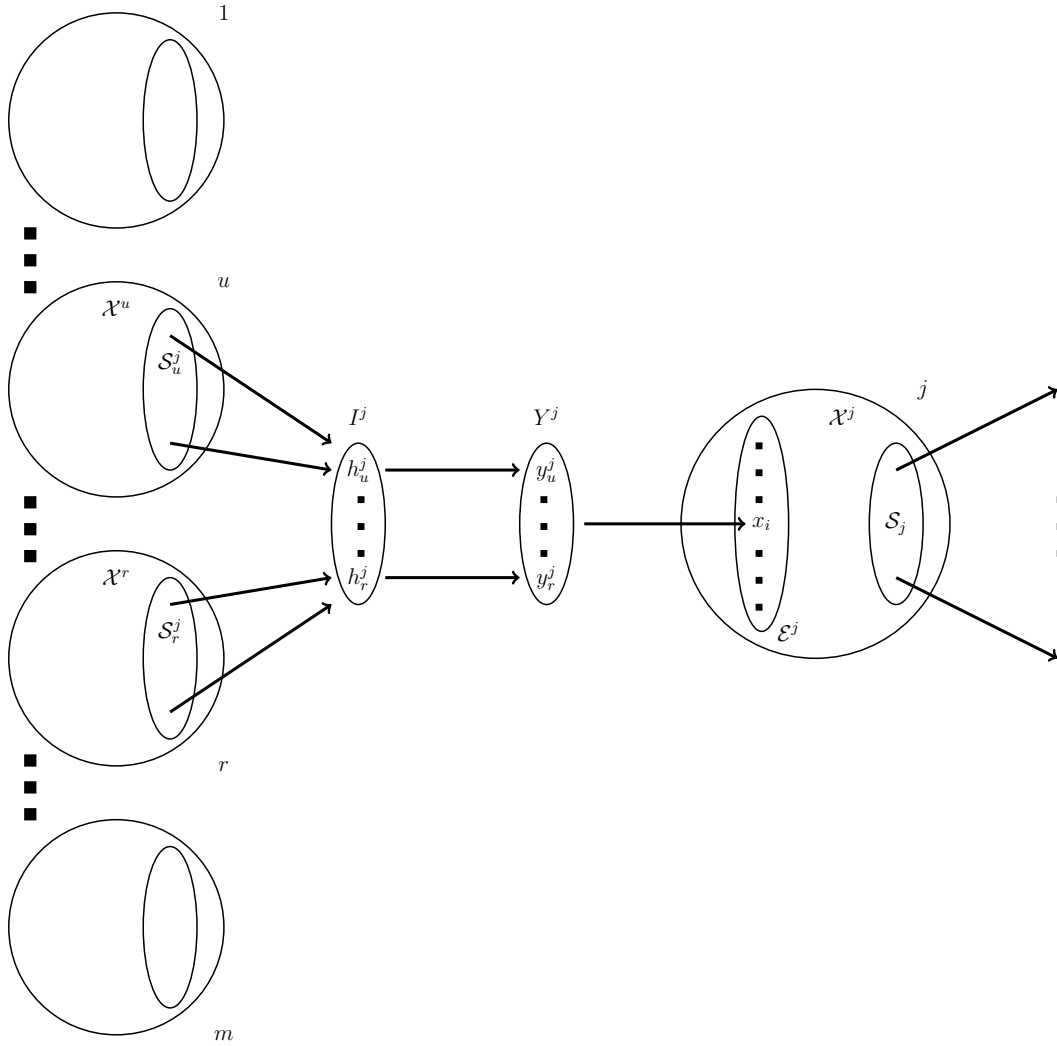


Figura 4 – Uma representação gráfica da forma pela qual as RDDs u, \dots, r (que são convizinhas de j) exercem influência sobre a RDD j .

Assim, tem-se que o estado de uma variável x_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:

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

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

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

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

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

2.5 Sincronização e Estabilidade em RDDAs

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

Uma *atribuição* A é um mapeamento $[(1, \mathbf{a}_w), \dots, (j, \mathbf{a}_v), \dots, (m, \mathbf{a}_z)]$, onde $\mathbf{a}_w \in \mathcal{A}^1, \dots, \mathbf{a}_v \in \mathcal{A}^j, \dots, \mathbf{a}_z \in \mathcal{A}^m$. 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. 5). Assim, pode-se dizer 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 será nomeada de *condição de estabilidade*: o(s) valor(res) de saída proporcionado(s) pelo(s) atrator(es) local(is) em A e que provém das RDDs vizinhas à RDD j é(são) compatível(is) com o(s) valor(res) do(s) sinal(is) de acoplamento requerido(s) pelo atrator mapeado para j em A .

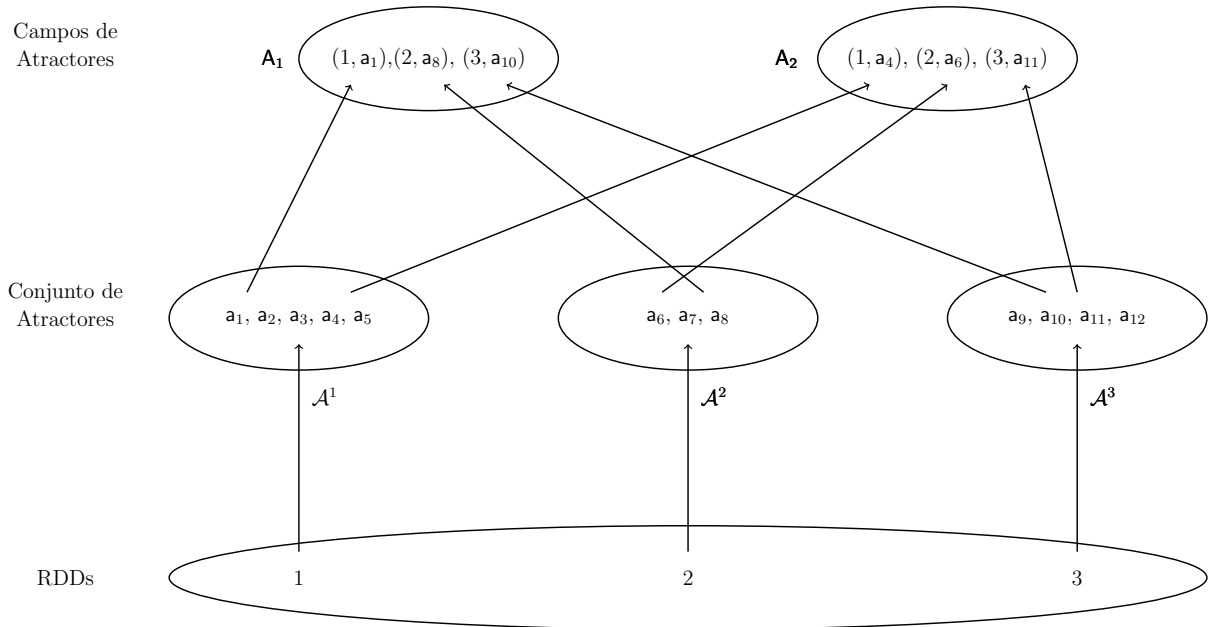


Figura 5 – Exemplo de uma atribuição num sistema com três RDDs. \mathcal{A}^1 , \mathcal{A}^2 e \mathcal{A}^3 denotam os conjuntos de atratores das redes locais 1, 2 e 3, respectivamente. A atribuição $A_1 = [(1, \mathbf{a}_1), (2, \mathbf{a}_3), (3, \mathbf{a}_{10})]$ contém o atrator \mathbf{a}_1 , oriundo da RDD 1; o atrator \mathbf{a}_3 oriundo da RDD 2 e o atrator \mathbf{a}_{10} oriundo da RDD 3. A atribuição $A_2 = [(1, \mathbf{a}_4), (2, \mathbf{a}_6), (3, \mathbf{a}_{11})]$ contém o atrator \mathbf{a}_4 , oriundo da RDD 1; o atrator \mathbf{a}_6 oriundo da RDD 2 e o atrator \mathbf{a}_{11} oriundo da RDD 3.

Em outras palavras, suponha uma atribuição A na qual o atrator \mathbf{a}_v está mapeado em j e os atratores $\mathbf{a}_w, \dots, \mathbf{a}_z$ estão mapeados em u, \dots, r , respectivamente, onde as RDDs u, \dots, r são vizinhas da RDD j . Pode-se dizer então 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:

$$x_i^j[\mathbf{a}_v] = f_i^j(x_k^j[\mathbf{a}_v], \dots, x_l^j[\mathbf{a}_v], y_u^j[\mathbf{a}_w], \dots, y_r^j[\mathbf{a}_z]), \forall x_i \in \mathcal{E}^j,$$

sendo que:

- $x_i^j[\mathbf{a}_v]$ indica o valor da variável x_i na RDD j quando j encontra-se no estado corrente do atrator \mathbf{a}_v ;
- $x_k^j[\mathbf{a}_v], \dots, x_l^j[\mathbf{a}_v]$ representam, respectivamente, os valores dos sinais locais x_k, \dots, x_l quando j encontra-se no estado corrente do atrator \mathbf{a}_v ;

- $y_u^j[\mathbf{a}_w], \dots, y_r^j[\mathbf{a}_z]$ denotam os valores dos sinais de acoplamento y_u^j, \dots, y_r^j quando as RDDs u, \dots, r estiverem nos estados correntes dos atratores $\mathbf{a}_w, \dots, \mathbf{a}_z$, respectivamente.

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 RDD fica “confinada” – para todo t – no atrator definido pela atribuição. No contexto desse trabalho o problema de estabilidade consiste em, dado uma RDDA com m RDDs, queremos encontrar seus campos atratores estáveis, caso existam.

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 \mathbf{X}^j o estado n -dimensional da j -ésima RDD, *sincronização completa* corresponde a

$$\mathbf{X}^1(t) = \mathbf{X}^2(t) = \mathbf{X}^3(t) = \dots = \mathbf{X}^m(t), \quad (2.3)$$

para todo t .

Sincronização parcial aqui refere-se ao cenário no qual um subconjunto $S = \{x_i, \dots, x_k\}$, $1 \leq i, \dots, k \leq n$, 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

$$S^1(t) = S^2(t) = S^3(t) = \dots = S^m(t), \quad (2.4)$$

para todo t . *Sincronização de saída* é um caso de sincronização parcial onde S é igual ao conjunto das variáveis de saída \mathcal{S} , assumindo que o \mathcal{S} 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 (do inglês *nondeterministic polynomial time*) (COOK, 1971). O SAT é o problema que consiste em determinar se existe uma valoração para as variáveis de uma dada fórmula Booleana F tal que essa 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 *Conjunctive Normal Form* (CNF) se é uma conjunção de cláusulas, onde uma cláusula contém uma disjunção de literais. A fórmula representada na Seção 2.6 é um exemplo de fórmula em CNF. Qualquer fórmula booleana F pode se converter a CNF, o método tradicional para transformar uma formulação Booleana na CNF são aplicadas regras de equivalências lógicas: eliminação de equivalência, eliminação de implicâncias, eliminação da dupla negação, leis de De Morgan e lei distributiva (RUSSEL; NORVIG et al., 2010).

Quando o número de variáveis n dentro de cada cláusula de uma fórmula em CNF é $n \leq 3$, dizemos que ela se encontra em 3CNF. A maioria dos SAT-solvers precisam que as fórmulas booleanas estejam em 3CNF. Em seguida, é apresentado 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 Hitting Set Problem

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

$$H \cap S \neq \emptyset, \forall S \in \mathcal{S} \quad (2.5)$$

Mais de um subconjunto de \mathbb{X} pode satisfazer as condições da cima. Denota-se de $\mathcal{H} = \{H_1, H_2, \dots, H_{|\mathcal{H}|}\}$ 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 Problema da Cobertura de Vértices (PCV)

Uma cobertura de vértices de um grafo (*Vertex Cover* em inglês) é um conjunto de vértices tal que cada aresta do grafo é incidente a pelo menos um vértice do conjunto. Ou seja, é um conjunto de vértices que contém pelo menos uma das pontas de cada aresta. Em outras palavras, uma cobertura de vértices é um conjunto V de vértices dotado da seguinte propriedade: toda aresta do grafo tem pelo menos uma ponta em V (COOK, 1971).

O problema da cobertura de vértices mínima é um problema de otimização que consiste em encontrar a menor cobertura de vértices em um grafo dado. Se o problema é enunciado como um problema de decisão, é chamado de problema da cobertura de vértices, o qual é um problema NP-completo: foi um dos Problemas NP-completos de Karp (KARP, 1972). O PCV é frequentemente usado em teoria da complexidade computacional como ponto de partida para provas de pertinência a classe de problemas NP-difícil.

Uma saída de HSP com k elementos é chamada k -HSP, um 2-HSP é uma generalização do problema da cobertura de vértices (GAREY; JOHNSON, 1979, 53–56), então um 3-HSP pode ser visto também como um PCV mediante hiper-grafos, onde se têm uma aresta entre até três, no lugar de dois vértices e a tarefa é determinar o subconjunto mínimo de vértices cobrindo todas as arestas (NIEDERMEIER; ROSSMANITH, 2003). Por sua vez, uma instância SAT pode ser reduzida a uma instância do PCV em tempo polinomial (GAREY; JOHNSON, 1979, 64).

2.10 Aplicações

A seguir menciona-se algumas das 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). Seus tipos celulares podem ser interpretados como suas bacias de atração (KAUFFMAN, 1993; WUENSCHÉ, 1998; WUENSCHÉ, 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 perturbações 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). Com isso, espera-se 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 podem 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. Por isso, um dos objetivos nessa área é descobrir se e quando um consenso completo ou parcial pode emergir de opiniões inicialmente diferentes. Para isso há estudos (PLUCHINO; LATORA; RAPISARDA, 2005) que fazem uso explícito de modelos de osciladores acoplados que veem a formação de grupos de opiniões como um processo de estabilização de estados sincronizados.

3 Cálculo de Atratores em RDDs

A fim de alcançar-se o objetivo anteriormente mencionado, foi adotada uma abordagem em dois passos: num primeiro momento calculam-se os atratores de todas as redes locais; num segundo momento, com os atratores de cada RDD em mãos, são exploradas as condições para calcular os campos atratores estáveis. Nesse capítulo são descritas duas abordagens para o primeiro passo, isto é, o cálculo de atratores locais. Uma abordagem usa *Hitting Set Problem* (HSP) e a outra usa *Boolean Satisfiability Problem* (SAT). Para ambas abordagens usa-se como base o algoritmo de [Dubrova e Teslenko \(2011\)](#), que é capaz de identificar atratores em redes Booleanas. Este algoritmo realiza chamadas para um SAT-*solver* com o objetivo de identificar caminhos no GTE nos quais os vértices são repetidos.

3.1 O Algoritmo de Dubrova e Teslenko

Esse algoritmo 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 rede Booleana no tempo t é representado por um vetor $x^t = [x_1^t, x_2^t, \dots, x_n^t]$ de variáveis de estado. Denota-se por S_k uma sequência de transições de estados

$$x^{t+0} \rightarrow x^{t+1} \rightarrow x^{t+2} \rightarrow \dots \rightarrow x^{t+k} \quad (3.1)$$

com k passos de tempo (que vão do passo t até o passo $t + k$), lembrando 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:

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

onde a função de transição f_i , $1 \leq i \leq n$, é como definida na Seção 2.1, sendo que “ \wedge ” e “ \leftrightarrow ” representam os operadores lógicos para conjunção e bicondicional, respectivamente. Esta representação da transição S_1 é chamada de *formulação proposicional* de S_1 , a qual é denotada por \widehat{S}_1 . A generalização para qualquer transição k é representada por \widehat{S}_k , onde esta formulação proposicional para ser analisada pelo SAT-*solver* deve estar na CNF. Essa nova fórmula booleana é denotada como \mathcal{P} e seu conjunto de cláusulas é definido como C .

Se existe uma atribuição que satisfaz as variáveis $x_1^t, \dots, x_n^t, x_1^{t+1}, \dots, x_n^{t+1}$ 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 satisfazível 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, sendo assim, uma vez que um caminho atinge um *loop* (atrator), ele nunca o deixa. Partindo-se desta premissa, podemos supor 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 mais uma vez antes em \widehat{S}_k .

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

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

onde x_i^k contém o valor da i -ésima variável na k -ésima transição. Com isso é possível notar que $Path$ armazena o valor das n variáveis para cada uma das k transições.

Dessa forma, temos que $Path = [Path_0, \dots, Path_k]$ e $0 \leq i \leq k$, onde $Path_i$ indica a sequência de valores que as n variáveis de estado assumem na i -ésima aplicação da função de transição. Esses valores podem ser 0 se for atribuído à variável o valor lógico *falso* ou 1 se for atribuído à variável o valor lógico *verdadeiro*. Assim, $Path$ 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, é necessário que ele seja atribuído ao conjunto solução \mathcal{A} . Essa operação de atribuição é feita por meio do operador lógico \vee , da seguinte forma:

$$\mathcal{A} \leftarrow \mathcal{A} \vee \mathcal{B}, \quad (3.3)$$

onde \mathcal{B} indica o atrator em forma de expressão Booleana, o qual é adicionado ao conjunto solução \mathcal{A} .

Para que a operação representada pela Eq. 3.3 seja válida é necessário que tanto \mathcal{A} como \mathcal{B} estejam representados via expressões Booleanas. Assim, existe um procedimento que converte o atrator representado em $Path$ (uma sequência de bits) em uma expressão Booleana \mathcal{B} , esse procedimento pode ser descrito da seguinte maneira:

- cada estado do atrator em $Path$ corresponde a uma cláusula em \mathcal{B} ;
- 0's no estado do atrator correspondem aos valores negativos das variáveis em \mathcal{B} ;
- 1's no estado do atrator correspondem aos valores positivos das variáveis em \mathcal{B} ;
- os literais (positivos e negativos das variáveis) são conectados pelo operador \wedge ;
- as cláusulas são conectadas através do operador \vee .

Para ilustrar o procedimento acima, vamos considerar 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:

$$\widehat{S}_k \leftarrow \widehat{S}_k \wedge \neg \mathcal{A} \quad (3.4)$$

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

Se em $Path$ não existirem estados repetidos, o tamanho de k é duplicado, ou seja, $k \leftarrow 2k$. Assim, com o aumento de \widehat{S}_k , a chance de se achar um atrator também aumenta. A descrição geral do algoritmo é a seguinte:

1. Ler as funções de transição F que governam as variáveis da rede Booleana.
2. Gerar a formulação proposicional \widehat{S}_k correspondente às sequências das transições.
3. Converter \widehat{S}_k à Forma Normal Conjuntiva (CNF), denotada por \mathcal{P} .

4. Executar o SAT-solver na instância \mathcal{P} :

- a) Se o SAT-solver retornar *satisfazível*, armazenar a saída em $Path$ e depois verificar se $Path$ contém um *loop*, e:
 - i. se contém um *loop*, guardar o *loop* em um conjunto solução \mathcal{A} , excluindo a bacia de atração do espaço de pesquisa e em seguida voltar para o passo 2;
 - ii. caso contrário, duplicar k ($k \leftarrow 2k$) e retornar para o passo 2.
- b) Se o SAT-solver devolver *insatisfazível*, retornar a lista de atratores \mathcal{A} e o algoritmo acaba.

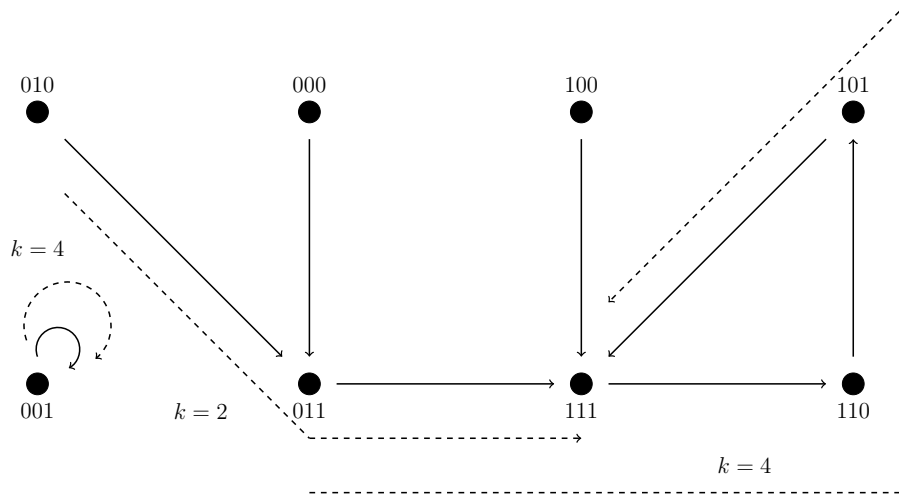


Figura 6 – Grafo de transições de estados para uma rede Booleana r de 3 variáveis. São mostrados em linhas pontilhadas os caminhos que o algoritmo analisa até encontrar os atratores.

Tomemos como exemplo uma rede Booleana r de três variáveis para ilustrar o algoritmo, como segue:

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

A formulação proposicional da sequência de transições $x^0 \rightarrow x^1 \rightarrow x^2$ 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 $x^0 \rightarrow x^1 \rightarrow x^2$ correspondem a um caminho válido no GTE representando a dinâmica da rede r . Depois \widehat{S}_2 é convertida para CNF (que passamos a denotar por \mathcal{P}), para ser avaliada pelo SAT-solver. No caso de ter-se a atribuição

$$Path = [010, 011, 111],$$

pode-se verificar que não existe atrator pois

$$Path_2 \neq Path_0, Path_2 \neq Path_1.$$

Então dobra-se k e \widehat{S}_4 é então gerada, para posteriormente usar o **SAT-solver** novamente e ter como resposta

$$Path = [011, 111, 110, 101, 111].$$

Com isso, pode-se verificar que $Path_1 = Path_4$, sendo que os estados das transições de 2 até k formam um atrator e são incluídos ao conjunto de atratores \mathcal{A} . Na iteração seguinte, como foi achado um atrator, k não é dobrado e utiliza-se a Equação 3.4 em

$$\widehat{S}_4 \leftarrow \widehat{S}_4 \neg \mathcal{A},$$

removendo assim o atrator encontrado e estados relacionados do espaço de busca.

Assim, o algoritmo continua até que todos os atratores sejam encontrados e não possa mais gerar caminhos válidos. No caso em que k é muito menor do que o diâmetro máximo do GTE, o algoritmo pode 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 do conjunto \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 terminado. Pode-se ver uma ilustração das bacias de atração e as atribuições de $Path$ na Figura 6.

3.2 Uma Nova Abordagem Baseada no HSP

Nesta seção é apresentada uma modificação do algoritmo de Dubrova e Teslenko (2011) descrito na Seção 3.1, usando um **HSP-solver** no lugar do tradicional **SAT-solver** para resolver a satisfatibilidade e assim validar os caminhos dentro do GTE. Esta abordagem requer a redução polinomial de SAT para HSP. Para isso, foi desenvolvido um procedimento de tempo linear para converter instâncias SAT em instâncias HSP, descrito na seção 3.2.2. Além disso, foi adaptado o algoritmo paralelo baseado em GPUs (*Graphics Processing Unit*) de Carastan-Santos et al. (2017) que é um **HSP-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), razão pela qual inicialmente foi utilizado 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, optou-se nessa abordagem não considerar a presença dos SLIs, isto é, na implementação dessa abordagem supomos que as RDDs estão “isoladas”. Para considerar a presença dos SLIs basta que seja realizada uma adaptação de modo a incorporar tratamento descrito na Seção 3.3.1.

3.2.1 Visão Geral

Dada uma rede Booleana r de n variáveis, quando o algoritmo de Dubrova e Teslenko (2011) é aplicado na rede, é criada uma formulação proposicional \widehat{S}_k que contém \mathcal{N} variáveis, onde $\mathcal{N} = n \times (k + 1)$. Depois \widehat{S}_k é convertida em uma instância do HSP mediante a abordagem da Seção 3.2.2 e se verifica sua satisfatibilidade. 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). Considerando que a resposta H contém os valores das variáveis das

transições no formato de nomes das variáveis x_i^k , esses valores devem ser transformados em 0's e 1's e agrupados em $Path$ pela transição à qual pertencem. Para fazer isto, as variáveis com atribuição negativa ($\neg x_{1\dots n}^{0\dots k}$) assumem o valor de "0" e as que não possuem o sinal ($x_{1\dots n}^{0\dots k}$) assumem o valor de "1".

A seguir, é verificado se $Path$ contém um $loop$, conforme foi descrito na Seção 3.1. Se $Path$ contém estados repetidos, o $loop$ é armazenado na lista de atratores \mathcal{A} pela operação descrita na equação 3.3. Isso faz com que a bacia de atração à qual o $loop$ pertence seja excluída do espaço de pesquisa. Caso contrário, se nenhum $loop$ é encontrado na $Path$, o tamanho k do \widehat{S}_k é duplicado e o processo é repetido iterativamente. Uma visão geral do algoritmo é apresentada abaixo (mais detalhes podem ser vistos na Seção 3.2.3):

1. Ler as funções de transição F que governam as variáveis da rede Booleana.
2. Gerar a formulação proposicional \widehat{S}_k correspondente às sequências das transições.
3. Converter \widehat{S}_k à Forma Normal Conjuntiva (CNF), denotada por \mathcal{P} .
4. Converter \mathcal{P} em uma instância \mathcal{S} do HSP.
5. Executar o HSP-solver para a instância \mathcal{S} gerada no passo anterior:
 - a) Se o HSP-solver retorna H tal que $|H| = \mathcal{N}$, transformar H em $Path$ e verificar se $Path$ contém um $loop$:
 - 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 2;
 - ii. caso contrário, duplicar k , $k \leftarrow k * 2$ e retornar ao passo 2.
 - b) Se o HSP-solver retorna H de tal modo que $|H| \neq \mathcal{N}$, retorne a lista de atratores \mathcal{A} e o algoritmo termina.

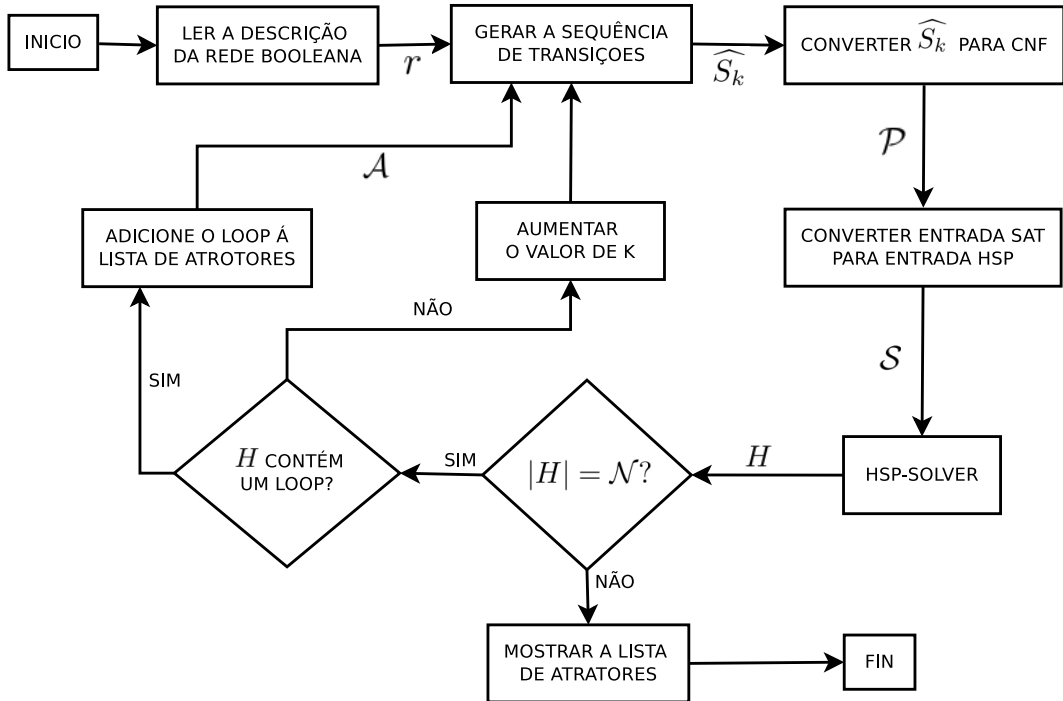


Figura 7 – Fluxograma representando as etapas do algoritmo para encontrar atratores com base no algoritmo Dubrova e Teslenko usando HSP para resolver a satisfazibilidade.

3.2.2 Uma Redução Simples e Linear do Problema SAT para HSP

De acordo com o definido acima, dado um conjunto \mathbb{X} e uma coleção \mathbb{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 $S \in \mathbb{S}$ e $|H|$ é mínimo. Assume-se que $\mathbb{X} = \cup_{S \in \mathbb{S}} S$, o que implica que um algoritmo que resolve HSP precisa apenas de \mathbb{S} como argumento. Nós chamamos o algoritmo que resolve HSP de **HSP-solver**.

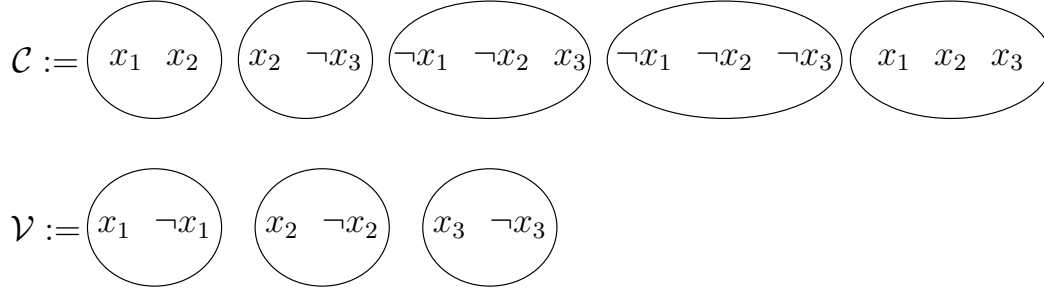


Figura 8 – Exemplo de conjuntos \mathcal{C} e \mathcal{V} cuja união é a instância do HSP de $\mathcal{P} = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$. Neste caso, observe que existe 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 $\{\neg x_1, x_2, x_3\}$.

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

Uma *interpretação* de \mathcal{X} é um conjunto $L = \{L_1, \dots, L_{\mathcal{N}}\}$, onde L_i é um literal da 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 cláusula 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, é mostrado que o algoritmo **Convert-SAT-to-HSP** 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 fórmula proposicional \mathcal{P} , **Convert-SAT-to-HSP** (\mathcal{P}) retorna a coleção

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

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 \mathbb{S} são dados como entrada para o **HSP-solver**. A Figura 8 mostra um exemplo de uma instância HSP obtida de uma instância SAT.

Em seguida, é mostrado o seguinte resultado direto.

Teorema 1. *Seja \mathcal{P} uma instância de SAT, $\text{Convert-SAT-to-HSP}(\mathcal{P}) = \mathbb{S}$, e $\text{HSP-solver}(\mathbb{S}) = H$. 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 $\text{HSP-solver}(\mathbb{S}) = H$, 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} . Isso implica que $H' \cap \{x_i, \neg x_i\} \neq \emptyset$ para cada i e $H' \cap C \neq \emptyset$ para cada $C \in \mathcal{C}$. Também deve-se atentar ao fato de H' ser uma interpretação que também implica que $|H'| = \mathcal{N}$. Com isso, como $|H|$ é mínimo, $|H| \leq |H'| = \mathcal{N}$.

Por outro lado, como o $\text{HSP-solver}(\mathbb{S}) = H$ 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, conclui-se que $|H| = \mathcal{N}$. Além disso, claramente **Convert-SAT-to-HSP** é um procedimento linear. \square

3.2.3 Implementação

Nesta seção serão fornecidos alguns detalhes adicionais do método, que é descrito com mais precisão através do Algoritmo 1. Para testar esta abordagem foram utilizadas duas redes de exemplo de 3 e 4 variáveis, respectivamente. O exemplo de 3 variáveis foi descrito na seção 3.1 e a seguir está descrito o exemplo de 4 variáveis, com suas funções de transição:

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

Com as informações das funções de transição da rede Booleana em mãos, é criada a \widehat{S}_2 , onde esta formulação proposicional é convertida para a CNF e armazenada em \mathcal{P} . Depois \mathcal{P} é convertida em \mathbb{S} , uma instância válida para HSP mediante o procedimento **Convert-SAT-to-HSP** descrito Seção 3.2.2, então \mathcal{P} é analisado por o **HSP-solver**. Para poder resolver a satisfatibilidade, 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 \widehat{S}_2 é *insatisfazível*. No caso em que $|H| = \mathcal{N}$, quer dizer que temos um caminho válido e o H armazenado na estrutura *Path* na forma de 0's e 1's. É utilizado 0 se a variável tem o valor negativo e 1 no caso positivo. Uma atribuição inicial válida H seria

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

e sua representação na estrutura de dados seria

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

Pode-se verificar que *Path* não contém um atrator e sendo assim, dobra-se o tamanho de k e gera-se a \widehat{S}_4 . Depois executa-se novamente o procedimento **Convert-SAT-to-HSP** e é gerada uma nova atribuição $Path = [1111, 1101, 1101, 1101, 1101]$, a qual contém o atrator $[1101]$. Esse atrator é transformado na fórmula booleana \mathcal{B} , a qual contém $(x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)$, depois \mathcal{B} é adicionado à lista de atratores \mathcal{A} através da Fórmula 3.3. Tudo esse processo desde a criação da formulação proposicional é repetido iterativamente como é descrito na seção 3.1 até encontrar todos os atratores.

Antes de usar o **HSP-solver**, é necessário converter as variáveis de \mathbb{S} em números fazendo uso de um dicionário, pois os programas **HSP-solver** (incluindo o programa do [Carastan-Santos et al. \(2017\)](#)) encontrados só permitem o uso de números inteiros positivos \mathbb{Z}^+ , não incluindo "0". Então, as variáveis x_i^t , junto com seus negativos $\neg x_i^t$ são convertidas em números positivos consecutivos e são processadas assim pelo **HSP-solver**. Quando é encontrada a resposta como conjunto de números, esta é convertida de volta a variáveis fazendo uso do dicionário e essas variáveis são guardadas em H . Essa é uma etapa adicional que não deve ser executada se for encontrado um **HSP-solver** que suporte todos os tipos de dados como elementos.

Um detalhe importante a respeito do tamanho inicial de k na \widehat{S}_k , é que seu comprimento ideal seria o valor do diâmetro do GTE, pois só é possível 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

Algoritmo 1 Find attractors in Boolean Network using HSP-solver

```

1:  $k := n$  /*  $n$  is the number of variables in  $r$  */
2:  $\widehat{S}_k := \text{genPropositionalFormulation}(r, k)$ 
3:  $\mathcal{P} := \text{convertCNF}(\widehat{S}_k)$ 
4:  $\mathcal{S} := \text{convertSATtoHSP}(\mathcal{P})$ 
5:  $H := \text{HSPsolver}(\mathcal{S})$ 
6:  $\text{loopFound} := \text{false}$ 
7:  $\mathcal{A} := \text{true}$ 
8: while  $|H| = n(k + 1)$  do
9:    $\text{Path} := \text{transformToListOfStates}(H)$ 
10:  for  $i = 0$   $k - 1$  do
11:    if  $\text{Path}[i] = \text{Path}[k]$  then
12:       $\mathcal{B} := \text{genBooleanFormulation}(\text{Path}[i], \text{Path}[k])$ 
13:       $\mathcal{A} := \mathcal{A} \vee \mathcal{B}$ 
14:       $\text{loopFound} := \text{true}$ 
15:      breakfor
16:    end if
17:  end for
18:  if  $\text{loopFound} = \text{false}$  then
19:     $k := k * 2$  /* double the length ( $k$ ) of the transitions sequence  $\widehat{S}_k$  */
20:  end if
21:   $\widehat{S}_k := \text{genPropositionalFormulation}(r, k)$ 
22:   $\widehat{S}_k := \widehat{S}_k \wedge \neg \mathcal{A}$  /*exclude from the search space the last computed attractor*/
23:   $\mathcal{P} := \text{convertCNF}(\widehat{S}_k)$ 
24:   $\mathcal{S} := \text{convertSATtoHSP}(\mathcal{P})$ 
25:   $H := \text{HSPsolver}(\mathcal{S})$ 
26:   $\text{loopFound} := \text{false}$ 
27: end while
28: return  $\mathcal{A}$ 

```

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, seguiu-se o algoritmo de [Dubrova e Teslenko \(2011\)](#), que adota $k = n$ (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. Por exemplo, na Figura 9, pertencente ao exemplo de 4 variáveis, seu GTE tem um diâmetro de 5, se o número de transições k for iniciado em 4, o \widehat{S}_4 retornará um atrator na primeira iteração do algoritmo, lembrando que \widehat{S}_4 contém 5 estados.

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, foi adotado o algoritmo HSP proposto por [Carastan-Santos et al. \(2017\)](#). Esse algoritmo segue uma busca exaustiva de soluções H , na qual as soluções candidatas, que são codificadas como combinações de variáveis de \mathbb{X} (Ver Seção 3.2.2). Pode-se observar que o processo de redução SAT-HSP (ver Seção 3.2.2) cria instâncias de HSP com a propriedade que suas soluções (se existirem) devem ter um tamanho igual a \mathcal{N} . Além disso, para encontrar atratores, é necessário apenas encontrar uma solução de uma dada instância, como contraposto a todas as soluções. Assim, exploram-se essas características modificando o algoritmo de [Carastan-Santos et al. \(2017\)](#) de tal forma que:

- O algoritmo de busca começa com soluções candidatas com tamanho $|H| = \mathcal{N}$.

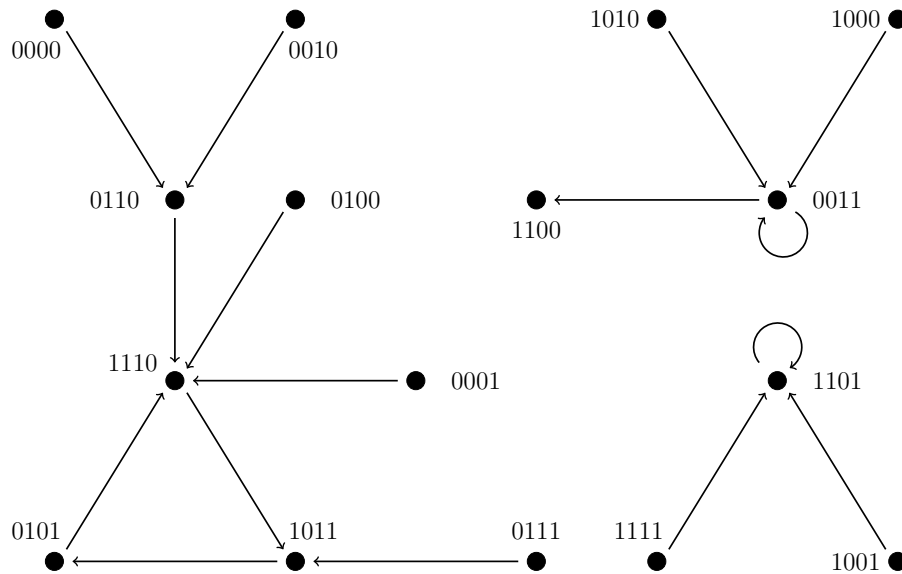


Figura 9 – GTE para a rede Booleana de 4 variáveis descrita na Seção 3.2.3, o GTE possui três atratores, dois deles são atratores pontuais e o outro é um atrator periódico composto por três estados.

- 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.

3.2.4 Experimentos da Abordagem Baseada em HSP

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. A implementação do método foi desenvolvida na linguagem Python versão 2.7, utilizando a biblioteca Satsipy a qual se encarrega de deixar a formulação proposicional em CNF. Também se faz uso do `HSP-solver` de [Carastan-Santos et al. \(2017\)](#) escrito em C++ e faz uso de GPUs `HSP-solver` e é usado para determinar a satisfatibilidade, mediante o método descrito na seção 3.2.2.

Foram realizados dois experimentos, um em uma RDD de 3 variáveis e outro usando uma RDD de 4 variáveis, descritas Seção 3.1 e 3.2.3, respectivamente. Para o experimento da RDD de 3 variáveis acima, o tempo de execução foi de 3,55 segundos, mas para o experimento de RDD 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](#)).

3.3 Um Método para Cálculo de Atratores em Redes Locais

A aplicação do algoritmo de Dubrova e Teslenko não é direta em uma RDD que é parte de uma RDDA, visto que 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. Além de serem regulados pelos sinais da própria rede, eles também podem ser regulados pelos SLIs.

Se excluirmos de uma rede local os seus SLIs, obteremos uma RDD “atômica” (uma rede única, não acoplada com qualquer outra). Dessa maneira, redes locais são como RDDs às quais são acrescentados os sinais localmente independentes (os sinais de acoplamento). Por isso, o uso do algoritmo de [Dubrova e Teslenko \(2011\)](#) 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. O produto dessa adaptação (aplicada a toda rede local j) é chamado de *rede local expandida*. O algoritmo para expansão e cálculo dos atratores de uma RDD j pode ser assim descrito:

1. Constrói-se uma extensão de j definida pelos seguintes procedimentos:
 - incorpora-se ao conjunto das variáveis de estado de j uma variável para cada um dos sinais de acoplamento que incidem sobre j ;
 - definem-se funções de transição tipo identidade para as variáveis de estado incorporadas.
2. Aplica-se o algoritmo de [Dubrova e Teslenko \(2011\)](#) na rede local expandida.

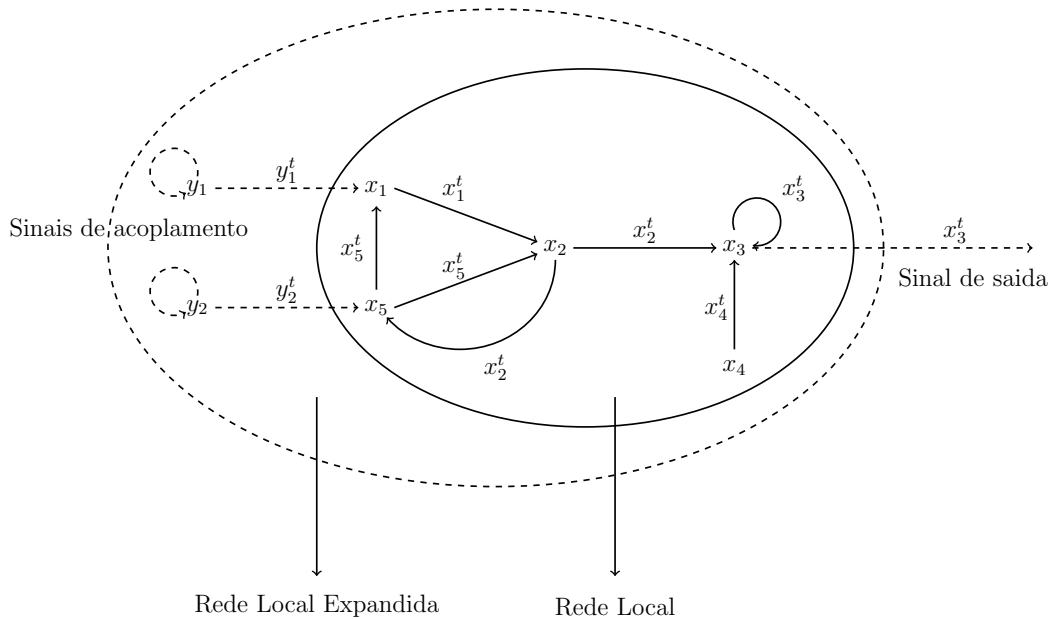


Figura 10 – Nesse exemplo de rede local, os sinais y_1^t , y_2^t são localmente independentes (os sinais de acoplamento) e os sinais x_1^t , x_2^t , x_3^t , x_4^t e x_5^t são localmente regulados. As variáveis x_1 e x_5 são as variáveis de entrada e a variável x_3 é variável de saída. Na rede local expandida é incorporada uma variável para cada sinal localmente independente e é definida uma função de transição do tipo identidade para cada uma.

Na Figura 10 são ilustrados 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”, que é um padrão de conexão entre as RDDs representado por um grafo regular, e tal que todas as redes locais sejam idênticas, ou seja, possuem a mesma dinâmica interna (mesmas variáveis \mathcal{X} e funções F), 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.

Nas abordagens e implementações para cálculo de atratores em redes locais descritas à frente assume-se uma importante restrição: dado uma RDD j , $1 \leq j \leq m$, pertencente a uma RDDA, com y_u^j, \dots, y_r^j sinais externos (sinais de acoplamento) incidindo sobre j , queremos computar o conjunto \mathcal{A}^j de atratores locais da dinâmica associada a j , em presença dos possíveis valores de y_u^j, \dots, y_r^j , assumindo que os valores de y_u^j, \dots, y_r^j são todos valores fixos.

Foi desenvolvida uma implementação para resolver o problema de cálculo de atratores em redes locais que usa um SAT-*solver* tradicional, mas que inova ao considerar explicitamente a presença dos sinais de acoplamento.

3.3.1 Tratando Sinais de Acoplamento em RDDs

Nesta seção, é apresentada uma implementação baseada no algoritmo de [Dubrova e Teslenko \(2011\)](#) que usa o SAT-*solver* de [Eén e Sörensson \(2003\)](#) para resolver a satisfatibilidade e, assim, encontrar os atratores. Mas dessa vez o algoritmo é adaptado para RDDs que fazem parte de uma RDDA, isto é, aqui consideramos que incidem sinais externos (sinais de acoplamento) sobre as RDDs (ver Seção 3.3).

Para esta implementação considera-se que cada RDD pode receber apenas um sinal de acoplamento y de outra RDD, mas pode receber sinais de várias RDDs. Também assume-se que os sinais de acoplamento terão valores fixos de 0 ou 1, já que as RDDs são redes booleanas. Para representar essa característica, os atratores devem ser calculados para cada combinação possível dos valores dos sinais de acoplamento resultando em um total de $2^{|Y|}$ grupos de atratores. Depois que os atratores de todas as combinações são calculados, eles são agrupados em uma única lista de atratores \mathcal{A} por RDD.

Outro ponto importante a se considerar é que uma RDD com sinais de acoplamento fixos é equivalente a uma RB com variáveis de estado fixo. Assim, um algoritmo que procura atratores em uma RB pode procurar atratores locais neste tipo de RDD. Além disso, algumas modificações têm 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 (ver Seção 2.5).

3.3.1.1 Visão Geral

O algoritmo começa a processar cada RDD da RDDA sequencialmente, o processo para encontrar atratores em uma RDD é o mesmo para as outras RDDs. Ao processar uma RDD, o algoritmo começa gerando as combinações para os sinais de acoplamento Y da RDD. Essas combinações são cadeias formadas por 0s e 1s, e seu comprimento é igual ao $|Y|$. Em seguida, para cada combinação, uma função que calcula os atratores é chamada. Essa função é nomeada como `FindLocalAttractors` e é baseada no algoritmo de [Dubrova e Teslenko \(2011\)](#). Adicionalmente, `FindLocalAttractors` gera variáveis que representam os sinais de acoplamento; essas variáveis fixam seus valores com os valores da combinação, também é responsável pelo processamento dessas variáveis. Finalmente, a lista de atratores e a combinação que a gerou são agrupadas e adicionadas a uma nova lista, essa lista é denotada como de *ListAtratorsCombinations*.

`FindLocalAttractors` recebe como parâmetros as informações da RDD e da combinação. As informações da RDD necessárias são: lista de variáveis \mathcal{X} , funções de transição F e variáveis de saída S correspondentes aos sinais de acoplamento Y . A função procura os atratores e gera \widehat{S}_k até ser avaliado pelo SAT-*solver* como insatisfatório, conforme descrito na seção Seção 3.1.

Para poder criar a \widehat{S}_k os valores das variáveis que representam os sinais de acoplamento devem ser definidos, porque essas variáveis farão parte da \widehat{S}_k . Para fazer isso, é necessário criar uma cláusula para fixar o valor da variável, em cada transição da \widehat{S}_k , isto é, $k + 1$ cláusulas são criadas por variável que representa os sinais de acoplamento. O valor fixo da variável depende da combinação, a posição na combinação indica qual valor fixo corresponde à variável, se o valor da variável for 1, são criadas cláusulas com atribuição positiva $(x_{1\dots n}^0\dots n)$, se o valor fixo for 0 se criam as cláusulas com atribuição negativa $(\neg x_{1\dots n}^0\dots k)$.

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 . Depois, a **FindLocalAttractors** analisa se o caminho retornado contém ou não um atrator; se o contém, ele os adiciona à lista de atratores \mathcal{A} , caso contrário, dobra o tamanho das transições k . Finalmente depois de varias iterações, **FindLocalAttractors** retorna a lista de atratores para a combinação que recebeu como entrada.

O algoritmo é resumido abaixo:

1. Gerar as combinações para o conjunto de sinais de acoplamento Y .
2. Repetir **FindLocalAttractors** 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 verificar se \widehat{S}_k contém um *loop*:
 - i. Se contém um *loop*, armazenar 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.
 - f) A lista de atratores e a combinação que a gerou são agrupadas e adicionadas a *ListAtrators-Combinations*.

3.3.1.2 Implementação

Para explicar a implementação dessa abordagem, apresentamos uma RDDA composta por 3 RDDs. Cada uma das RDDs possui 5 variáveis, para melhorar a identificação das variáveis, foi aplicada uma numeração correlativa aos índices das variáveis, com base nisso as variáveis das RDDs são respetivamente:

$$\begin{aligned}\mathcal{X}_1 &= \{x_1, x_2, x_3, x_4, x_5\} \\ \mathcal{X}_2 &= \{x_6, x_7, x_8, x_9, x_{10}\} \\ \mathcal{X}_3 &= \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}\end{aligned}$$

Cada uma das RDDs possui dois sinais de acoplamento: y_1 e y_2 , que representam as variáveis x_7 e x_{11} , respectivamente. Para explicar a implementação, ela será aplicada apenas à RDD 1, sendo o cálculo dos atratores nas RDDs 2 e 3 semelhante. As funções F da RDD 1 incluindo os sinais de acoplamento em forma de variáveis com valor fixo são descritos abaixo:

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

Como o cálculo de atratores em todas as RDDs segue o mesmo procedimento, serão pegos como exemplo somente a RDD 1 com suas sinais de acoplamento de $y_1^1 = 0, y_2^1 = 1$ correspondentes as variáveis x_7, x_{11}

das RDDs 2 e 3 respetivamente. para calcular os atratores de todas as combinações, é necessário calcular os estados possíveis que podem gerar, os quais gerarão $2^{|Y^1|}$ estados possíveis: $P(0, 1) = \{00, 01, 10, 11\}$.

`FindLocalAttractors` recebe as informações da RDD e os valores fixos da combinação dos sinais de acoplamento. Para este exemplo, a combinação 01 foi escolhida, para x_7 o 0 e para x_{11} o 1 respetivamente. Com esses dados, a função começa criando a \widehat{S}_2 e adiciona as variáveis de valor fixo x_7 e x_{11} para cada tempo t , lembrando que $0 \leq t \leq k$. O desenvolvimento da \widehat{S}_2 incluindo as variáveis dos sinais de acoplamento é:

$$\widehat{S}_2 = \widehat{S}_2 \wedge (\neg x_{11}^0) \wedge (x_7^0) \wedge (\neg x_{11}^1) \wedge (x_7^1) \wedge (\neg x_{11}^2) \wedge (x_7^2)$$

Em seguida, a \widehat{S}_2 é transformada à CNF e salva em \mathcal{P} , para depois ser avaliada pelo `SAT-solver`. Na primeira iteração do algoritmo, para \widehat{S}_2 o `SAT-solver` retorna os valores:

$$Path = [0001001, 1111001, 0011001]$$

Pode-se observar que os dois últimos valores nos estados atratores 01 são os valores fixos dos sinais de acoplamento Y . Ao avaliar $Path$, verifica-se que não possui um *loop*, então o valor do k é duplicado para $k = 4$. A \widehat{S}_4 é então gerada e os valores das variáveis fixas são adicionados. Em seguida, a \widehat{S}_4 é transformada na forma de CNF e armazenada em \mathcal{P} . Quando a formulação é avaliada com o `SAT-solver`, o resultado é:

$$Path = [0001001, 1111001, 0011001, 1111001, 0011001]$$

Ao analisar $Path$, pode-se verificar se o último estado $Path_5$ é repetido no estado $Path_3$, que indica um *loop*, e todos os estados entre $Path_5$ e $Path_3$ são reconhecidos como estados do atrator, o qual depois é adicionado á lista de atratores \mathcal{A} . Nas iterações a seguir, a \mathcal{A} no formato Booleano é adicionada de acordo com a Equação 3.4, depois é avaliada novamente pelo `SAT-solver`. Este processo se repete até 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) pelo Programa Satisfy no formato de 0's e 1's, precisamos – para efeitos de implementação do algoritmo – representá-los na forma de variáveis Booleanas e armazená-los 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.

No final, os atratores da lista \mathcal{A} pertencentes à combinação 01 são adicionados à *listAtratorsCombinations*. Quando o algoritmo termina de processar todas as combinações por meio da função `FindLocalAttractors`, suas listas de atratores também serão adicionadas à *listAtratorsCombinations*.

3.3.1.3 Experimentos

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. A implementação do método foi desenvolvida na linguagem Python versão 2.7, utilizando a biblioteca Satisfy a qual se encarrega de deixar a formulação proposicional no CNF e interagir com o `SAT-solver`, o programa MiniSAT de Eén e Sörensson (2003) feito em C++, muito utilizado para solucionar a satisfatibilidade nas fórmulas Booleanas.

Para poder testar a abordagem foi criado um gerador de RDDAs, o qual recebe como parâmetros: a quantidade de RDDs, o número de variáveis por RDD e o número de sinais de acoplamento. O gerador cria as RDDs em arquivos de texto, que tem como valores: o número de RDD, as variáveis locais, os sinais de acoplamento e as funções F . O gerador foi configurado para que as RDDs tenham um 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 cláusula, o qual esta no intervalo de 2 a 4. Este gerador foi melhorado na segunda parte e descrito na Seção 4.3.

Algoritmo 2 Find attractors in RDDs part of a RDDA using SAT-solver

```

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

```

Foram feitos vários experimentos para testar a aplicação prática da abordagem, variando os diferentes parâmetros das RDDAs.

Primeiro, o programa foi testado em uma RDDA, variando o número de RDDs em um intervalo de 50 a 500, o número de variáveis foi 5 e apenas um sinal de acoplamento foi considerado, ambos valores iguais para todas as RDDs. É possível verificar na Figura 11 que o tempo de execução cresceu linearmente em função do número de redes, levando em consideração que as redes possuem o mesmo número de variáveis e sinais de acoplamento e esse valor permanece fixo. Os dados deste teste se encontram na Tabela 1

Em seguida, foi realizado um teste em uma RDDA com 50 RDDs, o número de variáveis por rede variou de 3 a 30 e foi considerada apenas uma extremidade do acoplamento. É possível verificar na Figura 12 que o tempo de execução cresceu exponencialmente, esse comportamento está de acordo com a complexidade descrita no artigo de Dubrova e Teslenko (2011). Os dados deste teste encontram-se na Tabela 2.

Finalmente, foi realizado um teste em uma RDDA de 2 RDDs. Para esse teste, cada RDD possuía 5 variáveis e o número de sinais de acoplamento variou de 1 a 10. É possível verificar na Figura 13, que o tempo de execução também cresceu exponencialmente, isso porque dentro do algoritmo os sinais de acoplamento são convertidos em variáveis de valor fixo (consulte a Seção 3.3.1). A adição de sinais de acoplamento aumenta o número de variáveis na rede. 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 de uma RDDA com um intervalo de 50 a 500 RDDs, 5 variáveis e um sinal de acoplamento por RDD.

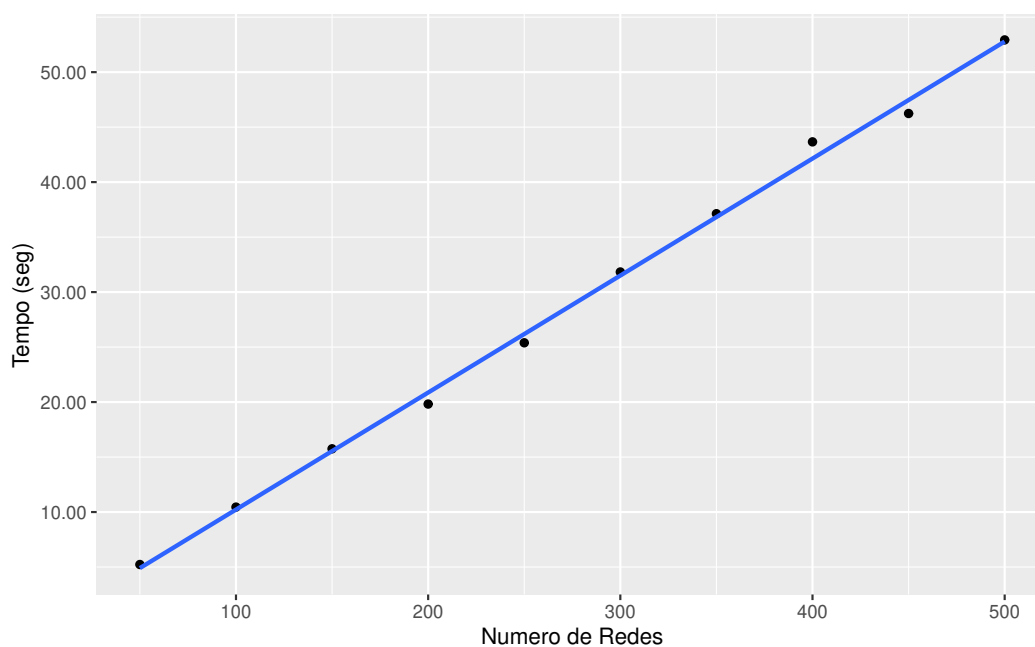


Figura 11 – Gráfico correspondente ao primeiro experimento em que a relação entre o número de redes e o tempo de execução do algoritmo é vista, quanto mais redes, maior o tempo de execução, mas esse crescimento é apenas linear, os dados com os quais esse gráfico gerou eles estão na Tabela 1

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 de uma RDDA com 50 RDDs cada RDDA, com um número de variáveis na faixa de 3 a 25 e um sinal de acoplamento por RDD.

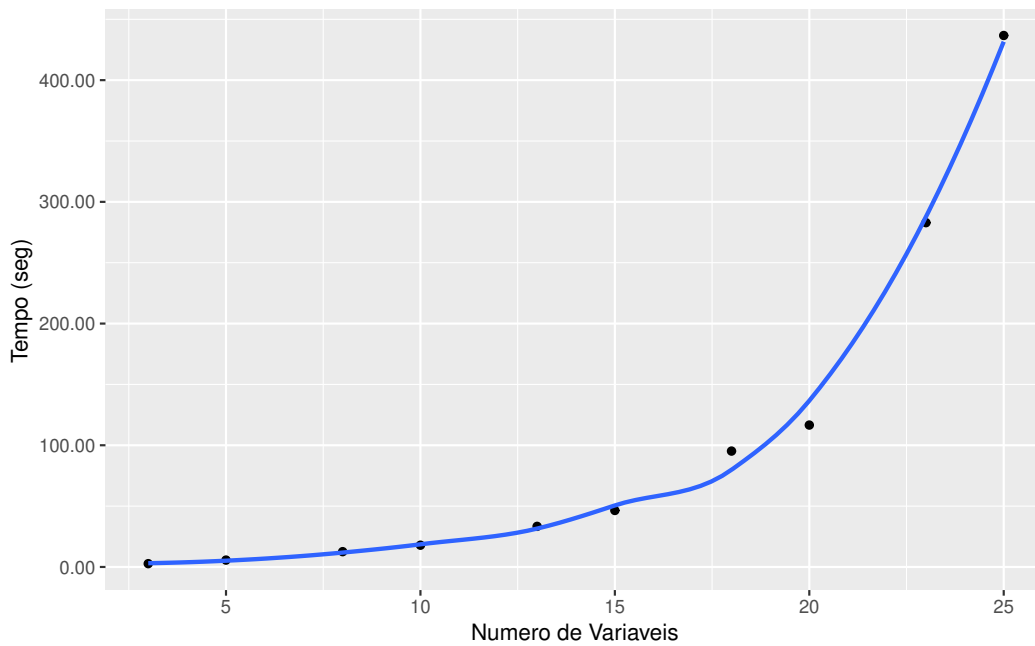


Figura 12 – Gráfico correspondente ao segundo experimento em que a relação entre o número de variáveis e o tempo de execução do algoritmo é observada, com maior número de variáveis na RDD maior tempo de execução, mas esse crescimento é exponencial, o que concorda com a complexidade apresentada em artigo de Dubrova e Teslenko (DUBROVA; TESLENKO, 2011), os dados com os quais foi gerado este gráfico estão na 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 de uma RDDA com 2 RDDs cada RDDA, com um número de variáveis de 15 e uma quantidade de sinais de acoplamento no intervalo de 1 até 10 por RDD.

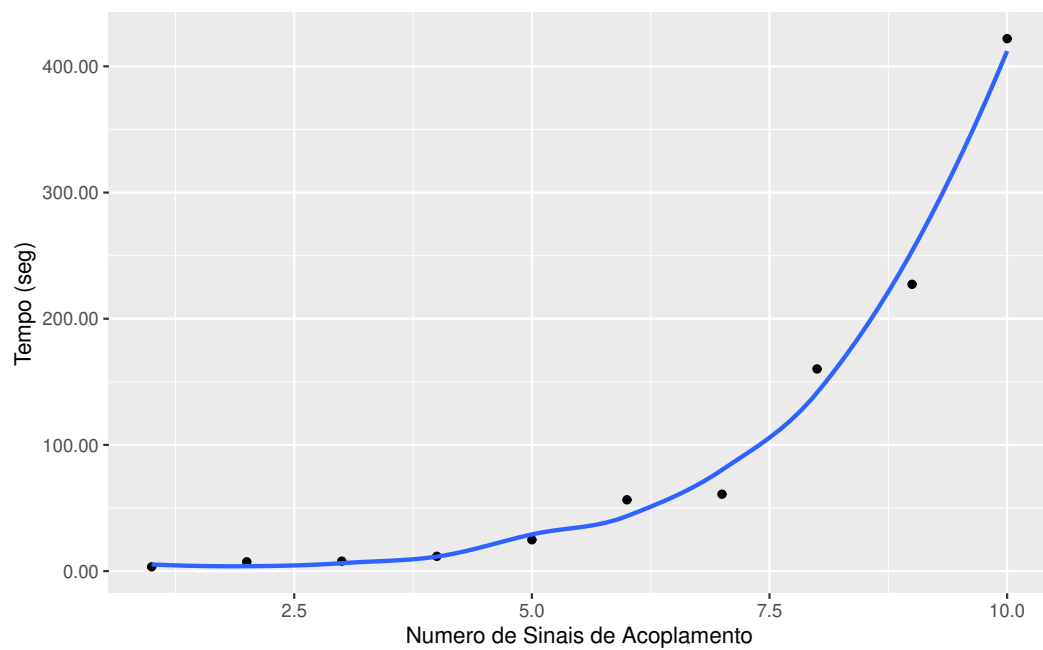


Figura 13 – Gráfico correspondente ao terceiro experimento em que a relação entre o número de sinais de acoplamento e o tempo de execução do algoritmo é vista, quanto maior o número de sinais de acoplamento por rede, maior o tempo de execução, mas esse crescimento é exponencial, o que concorda com a complexidade apresentada no artigo de Dubrova e Teslenko (DUBROVA; TESLENKO, 2011), como os sinais de acoplamento aumentam o número de variáveis na RDD, os dados com os quais eu gero esse gráfico estão na tabela 3.

4 Considerações para o Cálculo de Campos Atratores em RDDAs

Lembremos que foi adotada uma abordagem em dois passos: no primeiro passo calculam-se os atratores de todas as redes locais; no segundo passo, com os atratores de cada RDD em mãos, calculam-se os campos atratores estáveis. Neste capítulo apresentamos algumas considerações relacionadas ao passo dois.

4.1 Método Enumerativo

Dada a coleção de conjuntos de atratores $\mathcal{A}^1, \dots, \mathcal{A}^m$ (correspondentes às respectivas redes locais $1, \dots, m$) que é calculada conforme descrito na Seção 3.3 e em suas subseções, o passo seguinte (o “passo dois”) corresponde à computação dos campos atratores estáveis (ou simplesmente campos atratores) como combinações de atratores locais “compatíveis”.

Vale a pena lembrar que um campo atrator estável A é um mapeamento $[(1, \mathbf{a}_w), \dots, (j, \mathbf{a}_v), \dots, (m, \mathbf{a}_z)]$, com $\mathbf{a}_w \in \mathcal{A}^1, \dots, \mathbf{a}_v \in \mathcal{A}^j, \dots, \mathbf{a}_z \in \mathcal{A}^m$, que satisfaz a condição de estabilidade. Em outras palavras, um campo atrator A é um mapeamento $1 : 1$, um atrator para cada RDD, sendo que cada atrator é oriundo da dinâmica da RDD correspondente, onde para toda rede local j , $1 \leq j \leq m$, observamos a seguinte condição (chamada condição de estabilidade): o(s) valor(es) de saída proporcionado(s) pelo(s) atrator(es) local(is) em A e que provém das RDDs vizinhas à RDD j é(são) compatível(is) com o(s) valor(es) do(s) sinal(is) de acoplamento requerido(s) pelo atrator mapeado para j em A . Para mais detalhes, ver Seção 2.5.

Sem perda de generalidade, suponha que os conjuntos $\mathcal{A}^1, \dots, \mathcal{A}^m$ não têm elementos em comum, isto é,

$$\mathcal{A}^1 \cap \mathcal{A}^2 \cap \dots \cap \mathcal{A}^m = \emptyset.$$

Agora suponha que

$$|\mathcal{A}^1 \cup \mathcal{A}^2 \cup \dots \cup \mathcal{A}^m| = a.$$

Um algoritmo tipo força bruta para calcular os campos atratores consiste em computar todos os

$$\binom{a}{m}$$

mapeamentos/atribuições e registrar quais são aqueles que satisfazem a condição de estabilidade.

Devemos salientar também que os atratores em $\mathcal{A}^1, \dots, \mathcal{A}^m$ foram calculados assumindo a seguinte restrição: para toda RDD j , $1 \leq j \leq m$, pertencente a uma RDDA, com y_u^j, \dots, y_r^j sinais externos (sinais de acoplamento) incidindo sobre j , computamos o conjunto \mathcal{A}^j de atratores locais da dinâmica associada a j , na presença dos possíveis valores de y_u^j, \dots, y_r^j , assumindo que os valores de y_u^j, \dots, y_r^j são todos valores fixos.

Essa restrição reduz o espaço de busca de campos atratores porque descarta aqueles campos atratores que geram sinais de acoplamento periódicos (não fixos). Porém, no pior caso temos 2^n atratores associados a cada RDD, onde n é o número de variáveis por RDD, o que implica em

$$\binom{a}{m} = \binom{m2^n}{m}$$

possíveis mapeamentos/atribuições, tornando o uso de um algoritmo enumerativo inviável em termos práticos.

4.2 Método com Podas

Considerando que um método enumerativo não é viável, outro método baseado nas características da RDDA foi explorado com o intuito de promover ganhos de desempenho. Foram realizadas “podas” induzidas pela topologia da RDDA.

4.2.1 Visão Geral

Em uma RDDA (uma rede de m entidades dinâmicas interagentes, onde cada uma delas é uma RDD), o padrão de conexão “global” – a topologia da RDDA – está disponível na entrada do problema geral e é representado por um grafo dirigido conexo $G = (V, E)$, onde V representa o conjunto das RDDs e $(u, j) \in E$ se e somente se u é vizinha de j e j é vizinha de u , onde $u, j \in V$.

O método proposto neste trabalho para computação de campos atratores estáveis tem como entrada o grafo G que representa a topologia da RDDA e uma coleção de conjuntos de atratores $\mathcal{A}^1, \dots, \mathcal{A}^m$, correspondentes às RDDs $1, \dots, m$, respectivamente. Ele pode ser descrito – em linhas gerais – como um procedimento de duas etapas, assim descritas:

1. Cálculo dos pares estáveis de atratores.

Nessa etapa são identificados os pares estáveis de atratores que serão candidatos a compor o campo atrator. A ideia geral aqui é, dado que temos a topologia da RDDA e os atratores locais de cada RDD, fazer:

- a) se a RDD j é vizinha da RDD u , então computa-se o conjunto $\mathcal{A}^u \times \mathcal{A}^j$ de todos os pares de atratores possíveis de se formar com \mathcal{A}^u e \mathcal{A}^j , onde \mathcal{A}^u é o conjunto de atratores locais da RDD u e \mathcal{A}^j é o conjunto de atratores locais da RDD j ;
- b) para cada par em $\mathcal{A}^u \times \mathcal{A}^j$, testamos se ele satisfaz a condição de estabilidade (Ver Seção 4.2.2) para algum dos possíveis valores dos sinais de acoplamento que incidem sobre j e, em caso positivo, armazenamos esse par, produzindo assim um grafo que representará o conjunto de pares estáveis de atratores, o qual denotaremos por $G' = (V', E')$.

Essa ideia geral está descrita de um modo mais preciso pelo Algoritmo 3. Uma propriedade interessante do grafo $G' = (V', E')$ é que nele existe uma rotulação (coloração) nos vértices definida pelas RDDs de origem dos atratores. Em outras palavras, se o par $(\mathbf{a}_w, \mathbf{a}_z)$ é um par estável de atratores tal que \mathbf{a}_w vem da RDD u e \mathbf{a}_z vem da RDD j , então podemos assumir que u e j são as cores associadas a \mathbf{a}_w e \mathbf{a}_z , respectivamente; denotamos esse fato por \mathbf{a}_w^u e \mathbf{a}_z^j . A Fig. 14 ilustra a ideia geral desse esquema de coloração.

Algoritmo 3 : Computa pares estáveis de atratores

Require: um grafo dirigido $G = (V, E)$ representando a topologia da RDDA, uma coleção de conjuntos de atratores $\mathcal{A}^1, \dots, \mathcal{A}^m$

Ensure: o grafo $G' = (V', E')$ representando o conjunto dos pares estáveis de atratores

```

1:  $V' \leftarrow \emptyset$ 
2:  $E' \leftarrow \emptyset$ 
3: for cada  $(u, j) \in G$  do
4:   for cada  $(\mathbf{a}_w^u, \mathbf{a}_z^j) \in \mathcal{A}^u \times \mathcal{A}^j$  do
5:     if  $(\mathbf{a}_w^u, \mathbf{a}_z^j)$  satisfaz a cond. de estabilidade para algum dos possíveis sinais de acoplamento then
6:        $V' \leftarrow V' \cup \{\mathbf{a}_w^u, \mathbf{a}_z^j\}$ 
7:        $E' \leftarrow E' \cup (\mathbf{a}_w^u, \mathbf{a}_z^j)$ 
8:     end if
9:   end for
10: end for

```

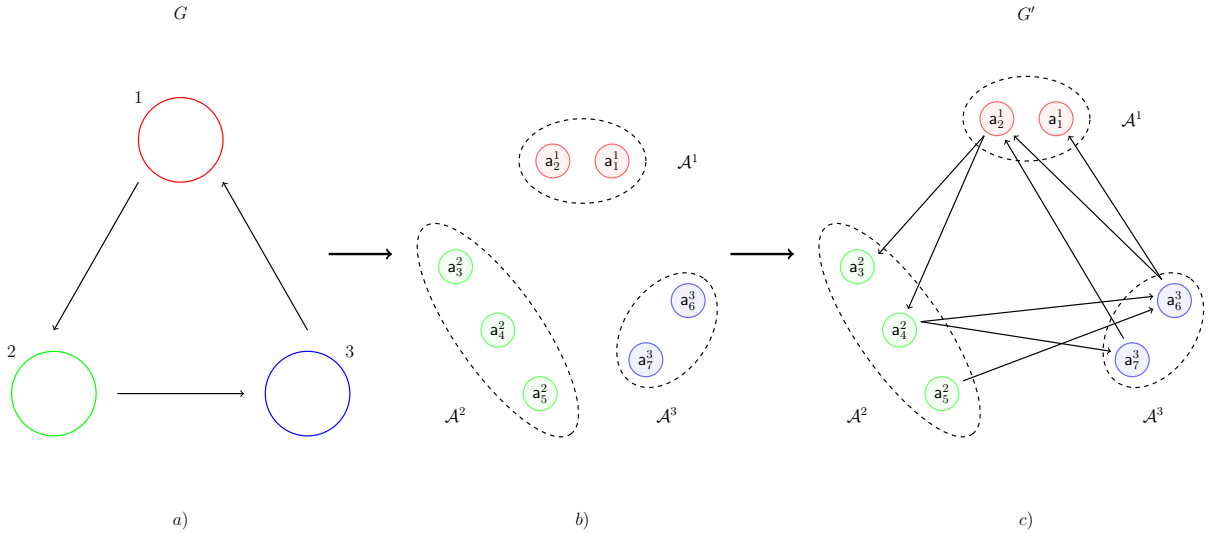


Figura 14 – (a): à esquerda, um exemplo de RDDA com três RDDs, sendo que nessa representação cada uma delas está associada a uma cor; (b): ao centro, uma representação da coleção de conjuntos de atratores locais obtida no “passo um”: o conjunto \mathcal{A}^1 contendo os atratores locais associados à RDD 1 (em vermelho), o conjunto \mathcal{A}^2 contendo os atratores locais associados à RDD 2 (em verde) e o conjunto \mathcal{A}^3 contendo os atratores locais associados à RDD 3 (em azul); (c): à direita, uma representação do grafo G' , sendo que cada vértice desse grafo corresponde a um atrator local e cada aresta representa um par estável de atratores, isto é, se existe uma aresta direcionada $a_2^1 \rightarrow a_3^2$ em G' , significa que a_2^1 e a_3^2 satisfazem a condição de estabilidade.

2. Montagem dos Campos Atratores.

Com os pares estáveis de atratores em mãos, podemos usá-los para compor os campos atratores. Como cada atrator em questão corresponde a um vértice do grafo $G' = (V', E')$, a identificação de um campo atrator pode ser definida como um problema em grafos, da seguinte forma: dado o grafo $G' = (V', E')$ com coloração nos vértices, queremos encontrar um subgrafo de G' , o qual denotaremos por $G'' = (V'', E'')$, tal que:

- $|V''| = |V'| = m$, onde m é o número de RDDs;
- $|E''| = |E|$, onde $|E|$ corresponde ao número de vizinhanças na RDDA;
- para todo $a_w^u, a_z^j \in V''$, vale que $u \neq j$, isto é, não existem dois vértices em V'' da mesma cor;
- todo par $(a_w^u, a_z^j) \in E''$, satisfaz a condição de estabilidade, isto é, todos os atratores em G'' são mutuamente estáveis.

Dado um grafo G' , podemos ter mais de um subgrafo G'' como definido acima, onde cada um deles corresponde a um campo atrator. Um procedimento inspirado em busca em profundidade para computar os subgrafos G'' pode ser descrito pelos Algoritmos 4 e 5.

Nesse procedimento denotamos por $[\rightsquigarrow i \rightsquigarrow j]$ (linha 1 do Algoritmo 5) o fato de que o valor de saída proporcionado pelo vértice/atrator i , considerando o sinal de acoplamento que incide sobre i frente ao caminho que está sendo computado em G' , é compatível com o valor do sinal de acoplamento requerido pelo vértice/atrator j . Em outras palavras, imagine que o algoritmo está testando se os vértices i e j satisfazem a condição de estabilidade – supondo que para chegar ao vértice i o algoritmo passou antes pelo vértice h (o caminho $h \rightarrow i \rightarrow j$ em G''). Então denotamos por $[\rightsquigarrow i \rightsquigarrow j]$ o fato de que i e j satisfazem a condição de estabilidade quando i recebe o sinal de acoplamento que vem de h . Denotamos por $leque(j)$ (linhas 3 e 7 do algoritmo 5) o conjunto de

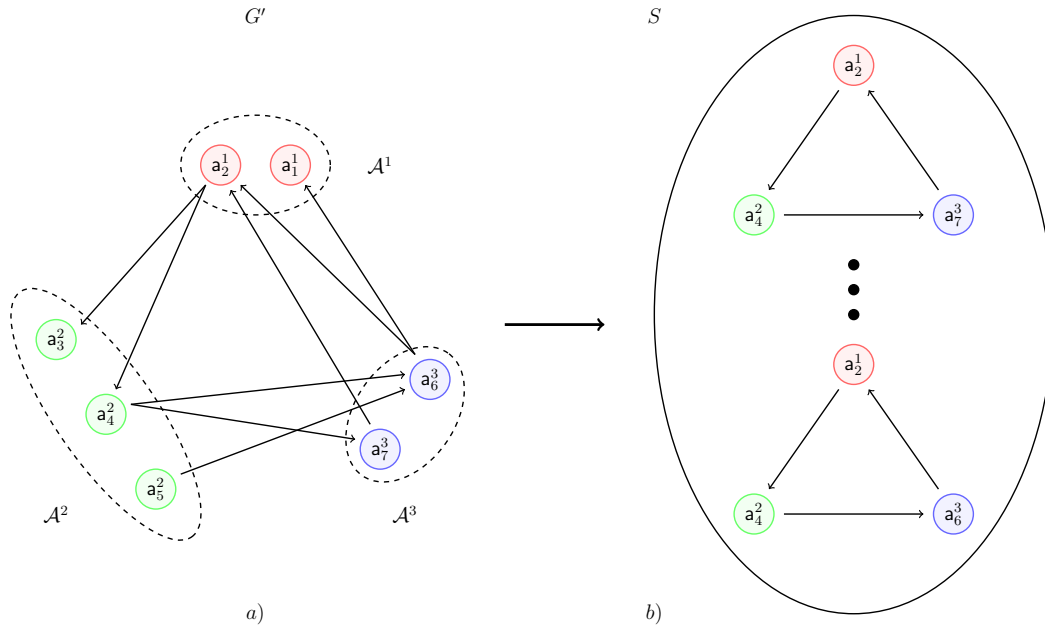


Figura 15 – (a): à esquerda, um grafo G' na entrada da etapa de montagem dos campos atratores; (b): à direita, uma representação de um conjunto solução S , com destaque para dois campos atratores: um, acima, contendo os atratores a_2^1 , a_4^2 e a_7^3 ; outro, abaixo, contendo os atratores a_2^1 , a_4^2 e a_6^3 .

arestas que saem do vértice j . Na Fig. 15 ilustramos graficamente dois possíveis campos atratores de um conjunto solução S .

Algoritmo 4 : Computa campos atratores

Require: um grafo dirigido $G' = (V', E')$ com uma função de cores nos vértices representando o conjunto de pares estáveis de atratores

Ensure: uma coleção S de subgrafos $G'' = (V'', E'')$, onde cada deles um representa um campo atrator

- 1: $E'' \leftarrow \emptyset$
 - 2: $paleta \leftarrow \emptyset$ /*conjunto das cores dos vértices de G' que já foram visitados*/
 - 3: $TamCaminho \leftarrow 0$ /* tamanho do caminho já visitado*/
 - 4: **for** toda $(i, j) \in E' - E''$ **do**
 - 5: $paleta \leftarrow paleta \cup \{cor(i)\}$
 - 6: $E'' \leftarrow E'' \cup Visita(TamCaminho, (i, j), paleta)$
 - 7: **if** $|E''| = |E|$ **then**
 - 8: $S \leftarrow S \cup E''$
 - 9: **end if**
 - 10: **end for**
-

4.2.2 Cálculo dos Pares Estáveis de Atratores

Antes de descrever o método em detalhes, é necessário definir a *condição de estabilidade* entre dois atratores, essa definição usa os conceitos descritos na Seção 2.2. Supondo uma RDDA denotada como RA formada por um conjunto de RDDs que vão de 1 até m , onde se tem dois atratores nos quais deseja-se testar a condição de estabilidade:

$$a_w \in \mathcal{A}_u, a_z \in \mathcal{A}_j, \{j, u\} \subset RA$$

Na Seção 2.5, foi definida a relação entre um atrator e os valores das variáveis que ele contém como $x_i^u[a_w]$. Mas um atrator como a_w pode conter um ou mais estados de transição, e é por isso que adicionamos um símbolo para indicar o estado $[e_n]$ onde n indica o numero de estado do atrator. Utilizando-o em

Algoritmo 5 Visita(TamCaminho, (i,j), paleta)

```

1: if TamCaminho = |E| e |Paleta ∪ {cor(j)}| = m e [↔ i ↔ j] then
   retorne (i, j)
2: else
3:   if |TamCaminho| < |E| e leque(j) = ∅ then
     retorne ∅
4:   else
5:     TamCaminho ← TamCaminho + 1
6:     paleta ← paleta ∪ {cor(j)}
7:     for toda (j, k) ∈ leque(j) do
8:       E'' ← E'' ∪ Visita(TamCaminho, (j, k), paleta)
9:     end for
10:  end if
11: end if

```

conjunto com a nomenclatura anterior, o valor de uma variável x_i^u dentro de um estado $[e_n]$ no atrator $[a_w]$ é definido como:

$$x_i^u[e_n][a_w]$$

É bom lembrar também que na Seção 3.3.1 foi definido que os atratores de cada RDD são calculados para as diferentes combinações de sinais de acoplamento que atingem á RDD, representamos este valor com o simbolo $\phi(y, a)$, para o caso do sinal do acoplamento y_u^j que foi considerado ao criar o atrator a_z como $\phi(y_u^j, a_z)$.

A condição de estabilidade é testada desde o atrator da RDD vizinha para o atrator da RDD vizinha. Supondo u vizinha de j , então $\exists y_u^j$ e a condição é testada desde a_w para o a_z , esse detalhe é importante ao momento da implementação do método, um exemplo da aplicação da condição de estabilidade é representada na Figura 16. Para que a condição de compatibilidade entre a_w e a_z seja satisfeita, deve-se cumprir que:

$$h_u^j(\forall x_i^u[e_n][a_w] | x_i \in S_u^j) = \phi(y_u^j, a_z^j), \forall e_n \in a_w^u \quad (4.1)$$

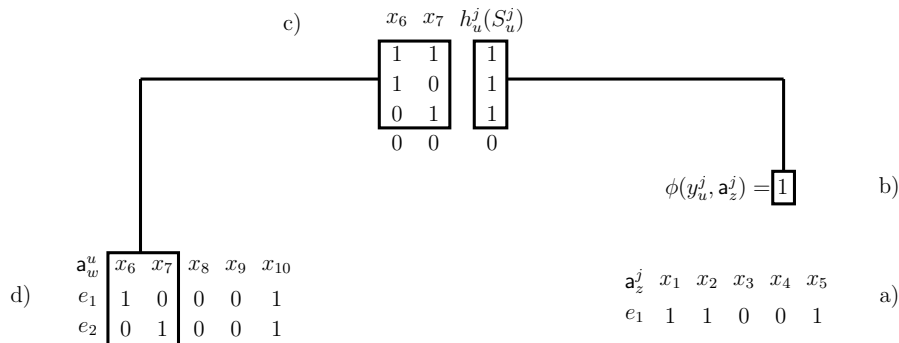


Figura 16 – Gráfico mostrando a relação da informação na condição de estabilidade, a) a_z^j com o e_1 que o compõe e os valores das variáveis, b) definição do valor do y_u^j quando o a_z^j é calculado $\phi(y_u^j, a_z^j) = 1$, c) tabela verdade relacionada ao y_u^j , primeiro selecionando as linhas com valor 1 e depois os valores das variáveis de saída, d) a_w^u do qual os valores pertencentes às S_u^j são selecionados; se esses valores forem iguais a alguma combinação de valores selecionados na tabela verdade, a condição de compatibilidade será atendida.

O método procura as relações entre os pares de atratores de RDDs diferentes, para as quais o método testa apenas os atratores cujas RDDs têm um sinal de acoplamento entre eles. O método começa reunindo todos os atratores em uma lista, depois é gerada uma tabela verdade (ENDERTON; ENDERTON, 2001)

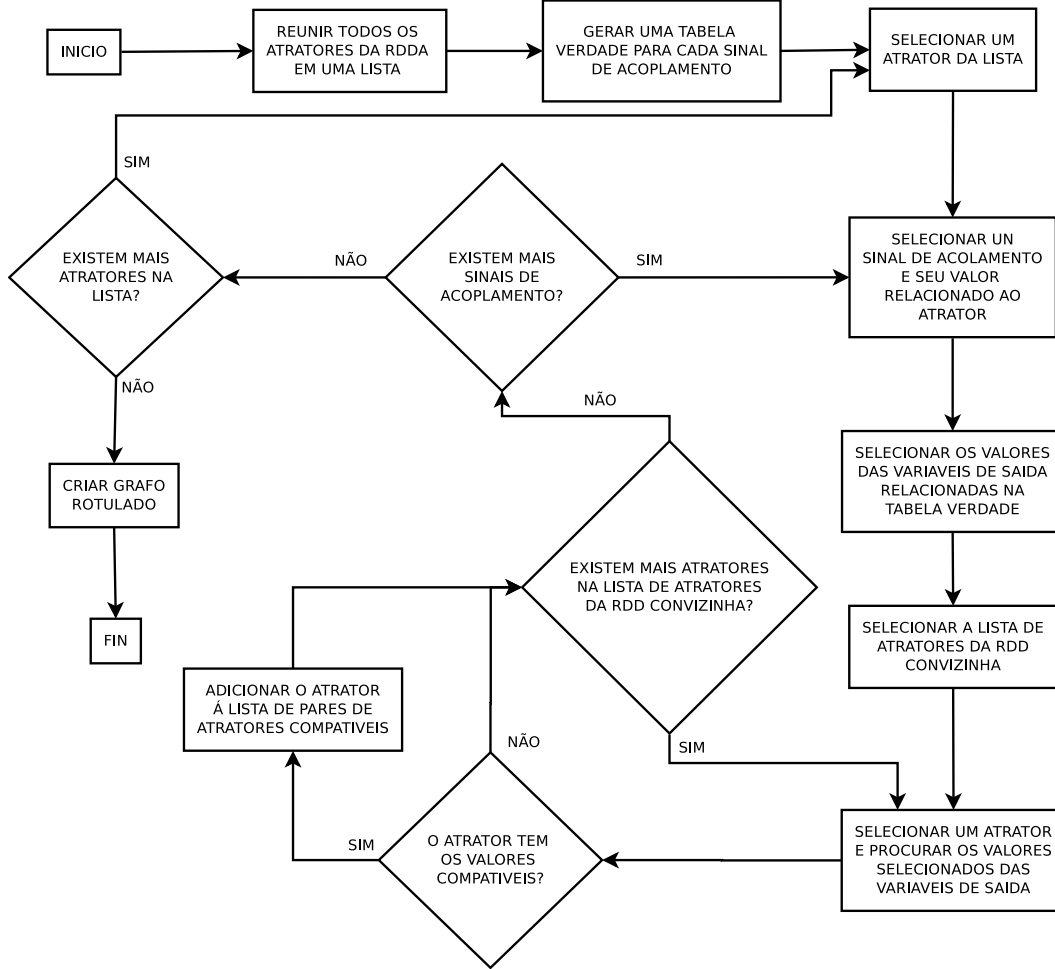


Figura 17 – Fluxograma do método do Cálculo dos pares estáveis de atratores.

para cada sinal de acoplamento. Uma vez feito isso, cada um dos atratores é analisado em busca de seus pares compatíveis; isso é feito testando sua condição de compatibilidade, e suportado pelo uso das tabelas verdadeiras. Finalmente, com os pares atratores compatíveis, o gráfico rotulado é gerado. O método é representado de forma geral em um diagrama de fluxo na Figura 17. As etapas do método são descritas abaixo:

1. Todas as listas de atratores das RDDs são agrupadas em uma única lista, denotamos esta lista como $\mathcal{A}^{\text{RDDA}}$. Supondo que se tem uma RDDA com RDDs que vão de 1 até m , a lista é conformada assim:

$$\mathcal{A}^1 \cup \mathcal{A}^2 \cup \dots \cup \mathcal{A}^m = \mathcal{A}^{\text{RDDA}}.$$

2. Uma tabela verdade é gerada para cada sinal de acoplamento da RDDA, na tabela verdade os termos são as variáveis do \mathcal{S} e a fórmula é a função de acoplamento h aplicada ao \mathcal{S} , as filas são preenchidas com os possíveis valores das variáveis de saída e suas respectivas respostas ao aplicar a função de acoplamento.

Seguindo o exemplo se tem duas RDDs u e j pertencentes á RDDA, onde j é vizinha de u . Lembrando que o sinal de acoplamento y_u^j depende de um conjunto de saída \mathcal{S}^j e de uma função de acoplamento h_u^j , seus valores designados são:

$$\mathcal{S}^j = \{x_a, x_b, x_c\}, h_u^j(\mathcal{S}^j) = x_a \vee (x_b \wedge x_c)$$

, a tabela verdade para o y_u^j é descrita na Tabela 4.

x_a	x_b	x_c	$h_u^j(\mathcal{S}^j) = y_u^j$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 4 – Tabela verdade que tem como entradas os valores das variáveis de saída \mathcal{S}^j e como fórmula a função de acoplamento h_u^j .

3. Os pares de atratores compatíveis são calculados utilizando a condição de estabilidade, para o qual são executados os seguintes passos:

a) Um atrator é selecionado da lista $\mathcal{A}^{\text{RDDA}}$ e todo sinal de acoplamento relacionado ao atrator é analisado procurando atratores compatíveis nas RDDs vizinhas, para o qual são executados os seguintes passos:

i. Um sinal de acoplamento y relacionado ao atrator é selecionado, o valor do y usado no cálculo do atrator é procurado em sua tabela verdade gerada no passo 2, depois o valor do y é procurado na coluna da fórmula $h(\mathcal{S})$. As linhas que correspondem a o valor do y são selecionadas.

Supondo que um atrator \mathbf{a}_n pertencente à RDD j e é selecionado depois o sinal de acoplamento y_u^j e o valor do y_u^j com o qual o \mathbf{a}_n foi gerado é "1", esse valor é pesquisado na coluna da fórmula $h_u^j(\mathcal{S}^u)$ na tabela verdade pertencente ao y_u^j . As linhas que correspondem a o valor do "1" são mostradas na Tabela 5.

x_a	x_b	x_c
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Tabela 5 – Tabela verdade com os valores das variáveis de saída \mathcal{S}^u correspondentes com uma sinal de acoplamento y_u^j com valor "1".

ii. Os valores das variáveis de saída \mathcal{S} que geram o valor do sinal de acoplamento são selecionados, esses valores são encontrados nas colunas dos termos da tabela verdade das linhas selecionadas no passo anterior. Lembrando que as variáveis de saída \mathcal{S} são um subconjunto do \mathcal{X} da RDD vizinha.

Continuando com o exemplo, para procurar os valores das variáveis de saída $\{x_a, x_b, x_c\}$ que geram o valor "1" no y_u^j , são selecionadas somente as linhas em que o valor de $h_u^j(\mathcal{S}^u) = 1$. Esses valores são mostrados na Tabela 5 nas colunas dos termos das filas respectivas.

iii. Todos os atratores da RDD vizinha relacionada ao y são analisados, procurando quais deles são compatíveis com os valores das variáveis de saída selecionadas na etapa anterior, para o qual são executados os seguintes passos:

A. Um atrator da RDD vizinha é selecionado e é avaliado se todos seus estados têm os mesmos valores que as variáveis previamente selecionadas de \mathcal{S} . Para o qual são executados os seguintes passos:

- Um estado atrator x é selecionado, os valores da suas variáveis são comparados com os valores selecionados da tabela verdade.
- Se o estado x não corresponder a nenhum conjunto de valores das variáveis de saída selecionadas da tabela verdade, o atrator é descartado e a etapa C. continua.
- Se a lista de estados no atrator da RDD vizinha ainda tiver estados não analisados, é selecionado o seguinte estado e é processado da mesma forma ate processar todos os estados do atrator.

Supondo que o atrator selecionado da RDD vizinha seja a_a , todos os estados que o compõem são analisados individualmente de acordo á condição de estabilidade, verificando se possuem os mesmos valores das variáveis de S^u mostrados na Tabela 5. Se o atrator a_a que possui dois estados e_1 e e_2 , na primeira iteração desta etapa o e_1 fosse escolhido, então, ao analisar seus valores para suas variáveis $\{x_a, x_b, x_c\}$, eles são comparados com os valores das variáveis de saída S^u que se encontram nas colunas $\{x_a, x_b, x_c\}$ das linhas selecionadas, na Tabela 5. O estado e_2 é processado da mesmo forma.

- B. Se o atrator da RDD vizinha estiver em conformidade com a etapa anterior, ele é unido em um conjunto com o atrator da RDD vizinha, esse conjunto é salvo em uma lista onde cada elemento contem os pares de atratores compatíveis, denotamos esta lista como P.

Se o atrator a_a estiver em conformidade como os valores das variáveis de S^u , é criado um conjunto de dois elementos, onde o primeiro elemento é o atrator da rede vizinha a_n e o segundo é o atrator da rede vizinha a_a , esse conjunto é adicionado a P.

$$P = P \cup \{(a_n, a_a)\}$$

- C. Se a lista de atratores na RDD vizinha ainda tiver atratores não analisados, retorne-se à etapa A.

iv. Se o atrator da RDD vizinha foi formado por mais sinais de acoplamento não processadas, retorna-se à etapa i., caso contrário, continue-se na próxima etapa.

- b) Se na lista \mathcal{A}^{RDDA} , ainda possui atratores não analisados, retorne-se à etapa a), caso contrário, se passa ao seguinte passo.

4. Depois que todos os atratores da lista \mathcal{A}^{RDDA} são processados, se tem todos os pares de atratores compatíveis na lista P. O conteúdo de P ao terminar de calcular os atratores compatíveis é:

$$P = \{(a_n, a_a), (a_n, a_b), (a_m, a_a), (a_m, a_b), (a_o, a_c), (a_a, a_f), (a_b, a_f), (a_c, a_g), (a_f, a_n), (a_g, a_o), \dots\}$$

5. É gerando o grafo rotulado $G' = (V', E')$, onde $V' = \mathcal{A}^{RDDA}$ e $E' = P$, os vértices são rotulados com o número da RDD à qual o atrator pertence. Esses rótulos podem ser representados por cores, as relações entre os cores e os números das RDDs são salvas no conjunto C' . Seguindo o exemplo, o conjunto C' é assim formado:

$$C' = \{(j, "verde"), (u, "vermelho"), (m, "azul") \dots\}$$

Dentro do grafo G' encontram-se em forma de subgrafos os campos de atratores estáveis, um exemplo desse grafo rotulado e um subgrafo representado um campo atrator estável é apresentado na Figura 18.

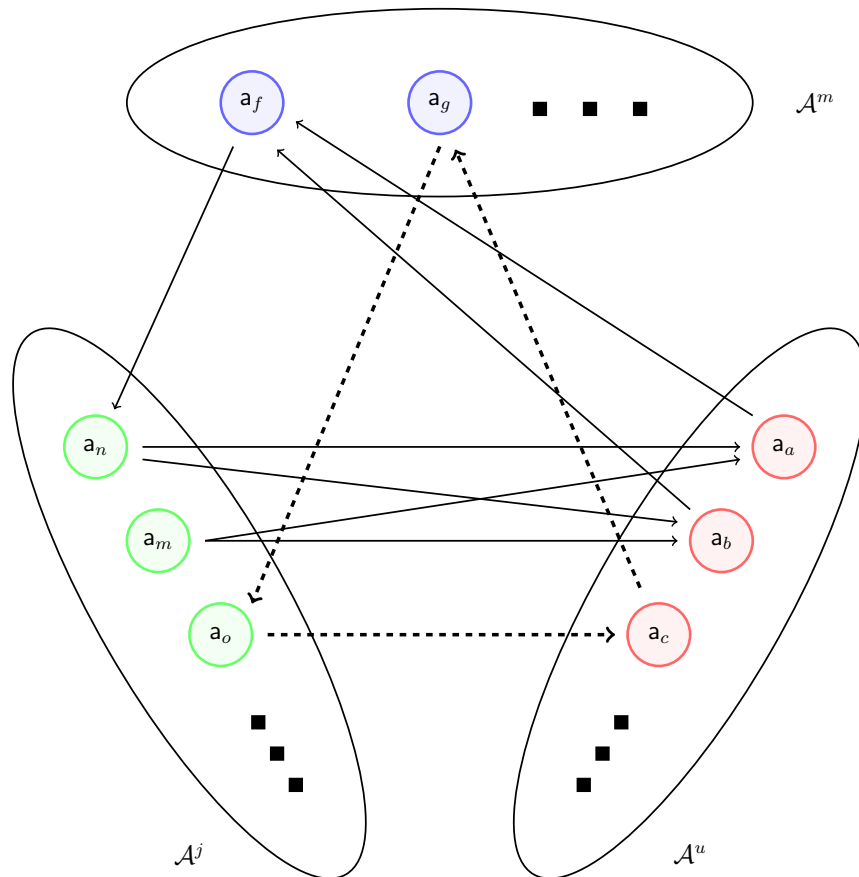


Figura 18 – Grafo G' de uma RDDA composta por 3 RDDs j, u, m . Os vértices foram coloridos de acordo com a RDD à qual o atrator representado pertence: vermelho para u , verde para j e azul para m . Um subgrafo representando um campo de atrator estável é composto de linhas tracejadas.

4.2.3 Implementação do Cálculo dos Pares Estáveis de Atratores

Nesta seção é explicada a implementação do método usando estruturas de dados, para o qual é utilizado o exemplo da RDDA apresentado na Seção 4.3 e cuja representação é descrita na Figura 21.

Lembrando que o método tem como entrada o grafo G que representa a topologia da RDDA e uma coleção de conjuntos de atratores $\mathcal{A}_1, \dots, \mathcal{A}_m$, correspondentes às RDDs $1, \dots, m$ respectivamente. Essas informações devem ser apresentadas na forma de uma estrutura de dados antes do início do método.

Para o método, são necessárias informações sobre a topologia da RDDA, esses dados são: sinais de acoplamento, funções de acoplamento e variáveis de saída. Esta informação é armazenada em uma lista onde cada elemento representa a informação de um sinal de acoplamento, chamamos essa lista de LSA . Cada elemento no LSA é formado por uma estrutura de dados que contém os seguintes elementos:

- O numero da RDD vizinha.
- O numero da RDD convizinha.
- O nome da variável na RDD vizinha que e criada para o sinal de acoplamento
- Um lista de nomes das variáveis de saída.

A estrutura de dados para o sinal de acoplamento y_2^1 é conformada assim:

$$[1, 2, x_{16}, [x_6, x_7]]$$

Todos os atratores das RDDs são calculados conforme o método descrito na Seção 3.3.1. Os atratores são guardados em listas individuais para cada RDD: $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. Os atratores são compostos por um o vários estados, os quais são salvos em uma lista por cada atrator, os estados são compostos de sequencias de "0" e "1", correspondentes em ordem a cada uma das variáveis da RDD em uma transição (ver Seção 4.2.2). A estrutura de dados para os atratores da RDD 1 é conformada assim:

$$\mathcal{A}_1 = [[0100010], [0110001, 100101], [0111011], [1110000]]$$

Com essas informações em mãos, o método é implementado da seguinte forma:

1. Cria-se uma tabela verdade para cada sinal de acoplamento na RDDA com a informação contida na *LSA*. Para gerar a tabela, os valores dos termos são preenchidos com os possíveis valores das variáveis de saída, o valor da fórmula é a função de acoplamento h é aplicada as variáveis do \mathcal{S} . O valor de $h(\mathcal{S})$ é o valor do sinal de acoplamento y . A tabela verdade para ao sinal de acoplamento y_1^2 é descrita na Tabela 6.

x_6	x_7	$h_2^1(\mathcal{S}_2^1)$	y_2^1
0	0	$x_6 \vee x_7$	0
0	1	$x_6 \vee x_7$	1
1	0	$x_6 \vee x_7$	1
1	1	$x_6 \vee x_7$	1

Tabela 6 – Tabela verdade que tem como termos os valores das variáveis de saída \mathcal{S}_2^1 .

As tabelas verdade são armazenadas em uma lista denotada como *TVG*, onde cada elemento da lista pertence a uma tabela verdade de um sinal de acoplamento. Por sua vez, essa tabela verdade também é uma lista que é denotada como *TV*, na qual cada elemento é uma estrutura de dados que representa uma linha da tabela verdade, onde esta estrutura é composta por dois elementos:

- Uma lista com os valores dos termos.
- O valor calculado da fórmula.

Seguindo o exemplo, a *TV* correspondente ao y_2^1 é composta assim:

$$[[[0, 0], [0]], [[0, 1], [1]], [[1, 0], [1]], [[1, 1], [1]]]$$

2. Com as listas de atratores de cada RDD prontas e com as tabelas verdade calculadas, é criada uma lista denotada como *LAIS*, essa lista contém informações dos atratores da $\mathcal{A}^{\text{RDDA}}$. Cada elemento em *LAIS* armazena as informações de um atrator em uma estrutura de dados, essa estrutura de dados é composta assim:

- O número da RDD ao qual o atrator pertence.
- Uma lista de estado(s) que compõem o atrator.
- A combinação dos sinais de acoplamento com os quais o atrator foi calculado, composto de sequencias de "0" e "1", correspondentes em ordem a cada um dos valores dos sinais de acoplamento.
- Uma lista denotada como *LY* que contém informações sobre os sinais de acoplamento que interagem com a RDD e cujos valores foram utilizados no cálculo do atrator. Como um atrator pode ter sido gerado com a influência de nenhum ou mais sinais de acoplamento, a *LY* pode estar vazia. Cada elemento em *LY* possui informações sobre um sinal de acoplamento e é composto por:

- O número da RDD vizinha.
 - O nome da variável que representa o sinal de acoplamento na RDD vizinha.
 - Uma lista de variáveis do conjunto de saída \mathcal{S} da RDD vizinha.
 - A tabela verdade correspondente ao sinal de acoplamento.
3. Quando todos os atratores e suas informações se encontram em *L AIS*, são calculados os pares de atratores compatíveis, para salvar esta informação é criada uma lista de incidência denotada como *LRAS*. Cada elemento em *LRAS* armazena os atratores compatíveis de um atrator, essas informações são armazenadas em uma estrutura de dados tipo lista de incidência (CORMEN et al., 2001) composta por:

- Na chave, uma tupla com a informação do atrator da rede vizinha : o numero da RDD, a lista dos estados do atrator.
- No arranjo, uma lista com as informações dos atratores compatíveis das redes vizinhas. Cada elemento da lista contem uma tupla com a informação do atrator da RDD vizinha : o numero da RDD, a lista dos estados do atrator.

Na criação da *LRAS*, as chaves são preenchidas com os valores dos atratores e o número da rede a que pertencem que estão no *L AIS*, deixando os arranjos a serem preenchidos posteriormente. Para para completar a *LRAS* todos os atratores da *L AIS* são analisados procurando seus atratores compatíveis, as etapas para esse processo são:

- a) Seleciona-se um registro da *L AIS*, que ainda não foi analisado. Supondo que o registro selecionado é correspondente ao atrator a_1 da RDD 1. O a_1 tem por estados os elementos da lista $[[0100010]]$ e foi calculado com os valores de 10 das combinações de valores dos sinais de acoplamento: y_2^1, y_3^1 , o registro na estrutura de dados *L AIS* é:

$$[1, 0100010, 10, LY]$$

Já que a RDD 1 tem duas RDDs vizinhas RDD 2 e RDD 3 (ver Figura 21), *LY* possui as seguintes informações:

$$[2, x_{16}, [x_6, x_7], [[[0, 0], [0]], [[0, 1], [1]], [[1, 0], [1]], [[1, 1], [1]]]$$

$$[3, x_{17}, [x_1, x_2], [[[0, 0], [0]], [[0, 1], [1]], [[1, 0], [1]], [[1, 1], [1]]]$$

- b) Analisa-se cada registro da *LY* para achar os atratores compatíveis das RDDs vizinhas, os passos deste processo são:
- i. Seleciona-se um registro da *LY*. Para fins práticos, apenas a relação y_2^1 entre a RDD 1 e a RDD 2 é analisada, as outras relações entre RDDs são processadas da mesma forma, lembrando que o registro correspondente ao y_2^1 é:

$$[2, x_{16}, [x_6, x_7]]$$

- ii. Seleciona-se o valor da combinação dos Y , para o sinal de acoplamento que é analisado. O y_2^1 é o primeiro sinal de acoplamento, então recebe o valor da primeira posição da combinação 10 que é 1. Mas também este valor pode ser encontrado na estrutura do atrator no valor da variável x_{16} , pois os valores das variáveis estão em ordem crescente:

$$[0100010] = (x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 0, x_{16} = 1, x_{17} = 0)$$

- iii. Verifica-se a compatibilidade entre o valor da combinação de y e a coluna da fórmula da tabela verdade; os valores das variáveis de \mathcal{S} que resultam no valor da combinação são selecionados. O valor 1 de y_2^1 é pesquisado em sua tabela verdade na coluna da fórmula, de acordo com a Tabela 6, os valores para x_6, x_7 que têm como resultado $y_2^1 = 1$ são:

$$[(0, 1), (1, 0), (1, 1)]$$

- iv. Seleciona-se a lista de atratores da RDD vizinha, nas informações do sinal de acoplamento no registro da LY se tem o número da RDD de onde o sinal vem. Com o número da RDD, a lista de atratores \mathcal{A} da RDD vizinha é selecionada. O valor da RDD vizinha é 2, este valor é usado para selecionar sua lista de atratores \mathcal{A}_2 , que contém:

$$\mathcal{A}_2 = [00010], [[10101], [01010]], [10111]$$

- v. Seleciona-se os atratores compatíveis com os valores das variáveis do \mathcal{S} identificados anteriormente, estes valores tem que se encontrar nos valores das variáveis correspondentes nos estados, leve-se em consideração que os estados não precisam ter os mesmos conjuntos de valores para as variáveis de saída. Os valores de x_6, x_7 são procurados no \mathcal{A}_2 . Esses valores de x_6, x_7 têm que ser os mesmos em todos os estados que fazem parte do atrator. Fazendo esse processo, tem-se uma lista de atratores compatíveis para o atrator [0100010]:

$$[[[10101], [01010]], [10111]]$$

- vi. Os atratores das RDD vizinha identificados como compatíveis, são adicionados à $LRAS$, dentro do arranjo pertencente ao atrator da RDD vizinha analisado. Os atratores compatíveis são representados em tuplas, onde o primeiro elemento é a lista dos estados do atrator e o segundo elemento é a RDD à qual o atrator pertence. Da \mathcal{A}_2 são selecionados os atratores que têm compatibilidade com o atrator a_1 da RDD 1. A chave do a_1 é:

$$([0100010], 1)$$

Os atratores compatíveis são representados assim:

$$[[[[10101], [01010]], 2], ([10111], 2)]$$

As tuplas dos atratores compatíveis são adicionados no arranjo do atrator da RDD vizinha. O registro da $LRAS$ ate este ponto é:

$$([0100010], 1) \longrightarrow [[[[10101], [01010]], 2], ([10111], 2)]$$

- vii. Volta-se para ao passo i). O processo é repetido para o outro sinal de acoplamento com seu respectivo valor. Os novos atratores compatíveis são então adicionados ao registro da $LRAS$. O registro para a_1 ficaria assim:

$$([0100010], 1) \longrightarrow [[[[10101], [01010]], 2], ([10111], 2), ([01010], 3), ([01011, 11111], 3)]$$

- c) Volta-se para ao passo a). O processo continua com o seguinte atrator aplicando o mesmo procedimento até terminar todos os atratores de $LAIS$.

4. Finalmente quando todos os pares de atratores compatíveis são calculados, é criado o grafo Rotulado G' com os dados da lista de incidência $LRAS$.

Este método foi testado com uma RDDA de 3 RDDs cada uma delas com 5 variáveis, sua topologia é mostrada na Figura 19, o grafo rotulado G' para esta RDDA é mostrado na Figura 20. Para criar o G' , foi necessário criar um dicionário designando números inteiros positivos \mathbb{Z}^+ incluindo o "0" para cada vértice de G' . Isso foi necessário porque o programa gerador da imagem do grafo apenas permitia números.

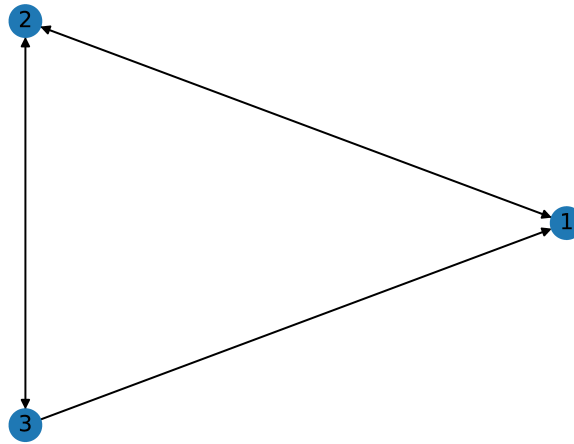


Figura 19 – Representação topológica por meio de um gráfico do teste feito com RDD formada por 3 RDDs, incluindo conexões entre as RDDs.

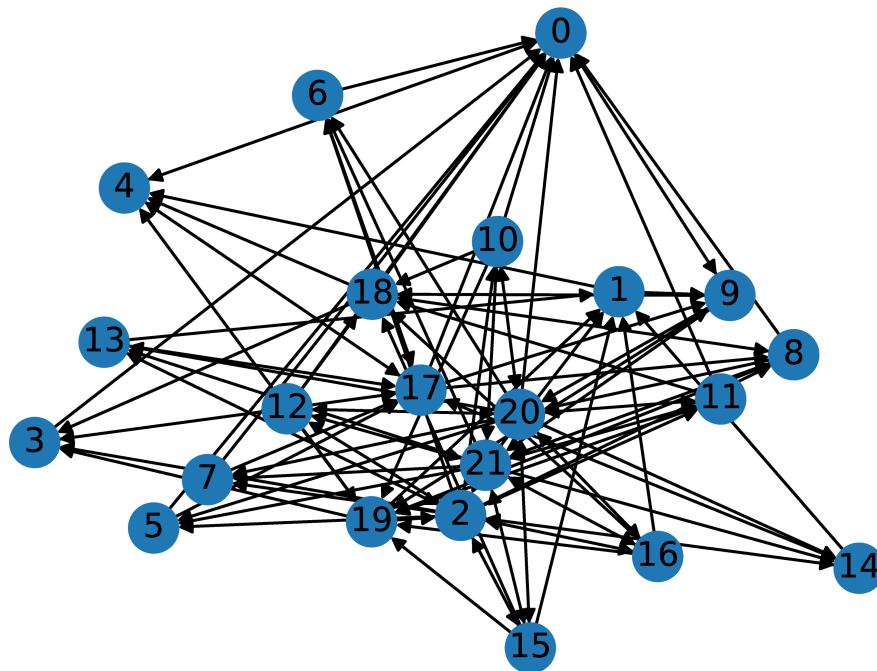


Figura 20 – Grafo de conexões compatíveis pertencente ao teste da RDDA de 3 RDDs e 5 variáveis, os nomes dos vértices foram trocados por números para melhorar a visualização.

4.3 Gerador de RDDAs

A fim de apoiar a realização de experimentos, foi implementado um gerador de RDDAs de acordo com as especificações descritas na Seção 2.4. As RDDAs geradas têm como característica principal a aleatoriedade no nível de: topologia da RDDA e funções de transição das RDDs. Essa característica é

baseada na escolha dos elementos de forma aleatória e é descrita quando esses elementos são gerados. O gerador recebe como parâmetros os seguintes valores:

- número m de RDDs que formam a RDDA;
- número de variáveis $|\mathcal{X}|$ que cada RDD terá;
- número máximo de sinais de acoplamento $|Y|_{max}$ que incidirão sobre cada RDD;
- número máximo de cláusulas $|C|_{max}$ das funções F (ver Seção 2.1). Foi decidido que essas funções fossem criadas diretamente na CNF (ver Seção 2.7);
- número máximo de literais denotado como LC_{max} , que compõem as cláusulas C , de acordo com o item anterior.

Em seguida, é apresentado um exemplo para explicar a operação do gerador, onde se tem como parâmetros: $m = 3$, $|\mathcal{X}| = 5$, $|Y|_{max} = 2$, $C_{max} = 3$, $LC_{max} = 3$. O programa gerador inicia criando a lista de variáveis \mathcal{X} de cada RDD, assumindo o valor $|\mathcal{X}|$ inserido. As variáveis começam sua numeração em 1 e aumentam uma a uma, de modo que essa numeração é mantida em todos os RDDs. Seguindo o exemplo, as respectivas variáveis para os 3 RDDs são:

$$\mathcal{X}^1 = \{x_1, x_2, x_3, x_4, x_5\}$$

$$\mathcal{X}^2 = \{x_6, x_7, x_8, x_9, x_{10}\}$$

$$\mathcal{X}^3 = \{x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$$

A seguir, é criado o padrão de conexão da RDDA, o qual é a estrutura da RDDA definida pelas conexões entre as RDDs (ver Seção 2.4). O número de conexões é igual ao número de sinais de acoplamento $|Y|$ que uma RDD possui. Sendo assim, o primeiro passo então é determinar $|Y|$, onde esse valor é escolhido aleatoriamente a partir de um intervalo de números inteiros entre 0 até $|Y|_{max}$. $|Y|$ é calculado desse modo para cada RDD, e por esse motivo, cada RDD tem números diferentes de sinais de acoplamento. Após essas etapas, uma quantidade $|Y|$ de RDDs são selecionadas, excluindo a RDD local, onde essa seleção é aleatória e sem repetição. As conexões entre as RDDs deste exemplo são mostradas na Figura 21.

Quando o padrão de conexão é definido, o próximo passo é definir a natureza da interação entre as RDDs (ver Seção 2.4). Para isso, é necessário definir para cada conexão: o conjunto de variáveis de saída \mathcal{S} ; a função do acoplamento h ; e o sinal de acoplamento y . As variáveis que compõem \mathcal{S} são selecionadas aleatoriamente no conjunto de variáveis locais \mathcal{X} da RDD vizinha. Seguindo o exemplo, usa-se a conexão que sai do RDD 2 e chega ao RDD 1, o \mathcal{S}^2 da RDD 2 é o seguinte:

$$\mathcal{S}^2 = \{x_6, x_7\}$$

Com os elementos do \mathcal{S}^2 escolhidos, a função de acoplamento h é definida. Normalmente, essa h deve admitir todos os tipos de expressões Booleanas formadas com as variáveis do conjunto de saída, mas por motivos de tempo e simplicidade, a h foi reduzida ao uso dos operadores Booleanos \wedge e \vee e pode ser escolhida manualmente. Também por simplicidade, todas as funções de acoplamento usarão o mesmo operador. Esse aspecto pode ser aprimorado em trabalhos futuros, pois pode ser substituído por um gerador de funções Booleanas. A aplicação de h no \mathcal{S}^2 resulta no sinal de acoplamento y . Aplicando esses conceitos ao nosso exemplo, o y_2^1 gerado com um operador \vee é:

$$y_2^1 = (x_6 \vee x_7)$$

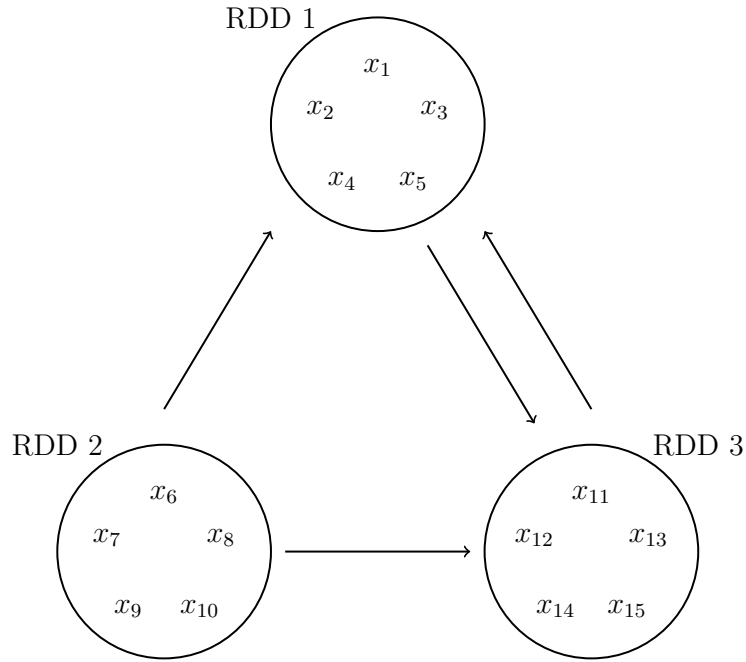


Figura 21 – Representação topológica do exemplo de uma RDDA de 3 RDDs e 5 variáveis cada, incluindo conexões entre as RDDs.

O próximo passo é gerar as variáveis que representam os sinais de acoplamento; essas variáveis terão o índice correlativo no índice mais alto de todas as variáveis de RDDA. Para calcular esse valor, todas as variáveis de todas as RDDs são combinadas em um único conjunto, que é indicado como $\mathcal{X}^{\text{RDDA}}$, com o objetivo de obter o último índice igual ao comprimento do conjunto $|\mathcal{X}^{\text{RDDA}}|$. Neste último índice, adicionamos 1 e teremos o índice da variável do primeiro sinal de acoplamento, para os seguintes sinais de acoplamento, 1 é adicionado ao índice já atingido, aplicando isso ao nosso exemplo, temos:

$$|\mathcal{X}^{\text{RDDA}}| = 15$$

$$y_2^1 \leftrightarrow x_{16}$$

A seguir, é criada a dinâmica interna de cada RDD, que é governada pelas funções de transição F (ver Seção 2.1). Ao gerar F para cada RDD, as variáveis criadas para os sinais de acoplamento Y são incluídas. As funções f que regulam cada uma das variáveis são criadas em CNF (ver Seção 2.7). Para determinar o número de cláusulas $|C|$ dentro de f , seu valor é escolhido aleatoriamente em um intervalo de números inteiros entre 1 e $|C|_{\text{max}}$. Da mesma forma, no caso da quantidade de literais em cada cláusula, seu valor é escolhido aleatoriamente a partir de um intervalo de números inteiros entre 1 e $|LC|_{\text{max}}$, os literais são as variáveis da RDD, incluindo as variáveis que representam os sinais de acoplamento, essa quantidade de literais é calculado para cada cláusula, e portanto, cada uma delas pode ter um número diferente de literais. As variáveis são escolhidas aleatoriamente sem repetição para cada cláusula, o símbolo de que elas terão positivo ou negativo também é escolhido aleatoriamente. Aplicando esse conceito ao nosso exemplo, vemos que f_1 correspondente à variável x_1 gera:

$$|C| = 2$$

$$|LC| = 3$$

$$f_1^{t+1} = (x_2^t \vee x_6^t \vee x_7^t)$$

$$|LC| = 2$$

$$f_1^{t+1} = (x_4^t \vee x_5^t \vee x_7^t) \wedge (\neg x_1^t \vee x_6^t)$$

Resumindo os passos do algoritmo do gerador de RDDAs, tem-se:

1. Criar o conjunto de variáveis \mathcal{X} para cada RDD;
2. As conexões entre as RDDs são geradas aleatoriamente, as variáveis de saída \mathcal{S} são selecionadas e, em seguida, a função de acoplamento h é inserida manualmente, h é a mesma para todas as conexões formando I ;
3. As funções I são aplicadas às variáveis do \mathcal{S} para obter os sinais de acoplamento Y ;
4. Os sinais de acoplamento Y são nomeados como variáveis locais \mathcal{X} e seus índices são correlativos a $|\mathcal{X}^{\text{RDDA}}|$;
5. A dinâmica local é determinada, criando as funções de F que determinam a dinâmica interna dos RDDs, os componentes dessas funções são selecionados aleatoriamente, as funções de F são geradas em CNF;

5 Conclusões Finais

5.1 Principais Contribuições

As principais contribuições desse trabalho são:

- Foi desenvolvida uma modelagem de sistemas de entidades interagentes por Redes Dinâmicas Discretas Acopladas (RDDAs). Conforme descrito nas Seções 1.3 e 2.10, os conceitos de estabilidade e sincronização em redes de entidades dinâmicas interagentes são de grande importância e têm várias aplicações possíveis em Bioinformática, Mineração de Dados, Telecomunicações, Redes de Distribuição e Redes Sociais, por exemplo.
- Foi apresentada uma redução linear $[SAT \rightarrow HSP]$ a fim de poder-se usar o algoritmo de Carastan-Santos et al. (2017) como um SAT-solver.
- Foi implementada uma abordagem original (baseada em SAT “clássico”) proposta por Dubrova e Teslenko; porém, adaptada de modo a considerar a presença de sinais externos.
- Foi descrito e implementado um método para calcular pares estáveis de atratores de diferentes RDDs que fazem parte de uma RDDA. Esse método baseia-se no uso das informações do sinal de acoplamento.
- Para testar os métodos e dar suporte aos experimentos, implementamos um gerador de RDDAs que atende às características das RDDAs descritas na Seção 2.4 e recebe como parâmetros: número de RDDs, número de variáveis por RDD, número de conexões máximas entre as RDDs, número máximo de variáveis de saída e número máximo de cláusulas por função de transição.

5.2 Limitações e Desafios

Quando foi concebida a estratégia de usar o algoritmo de Carastan-Santos et al. (2017) como um SAT-solver, a ideia era que se pudesse 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 $[SAT \rightarrow Hitting Set]$ levar a instâncias do *Hitting Set* de pior caso para o algoritmo de Carastan-Santos et al. (2017). 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 Trabalhos Futuros

O desdobramento futuro mais direto e importante para essa pesquisa corresponde à implementação e análise da etapa de montagem dos campos atratores a partir de pares estáveis de atratores, conforme descrito na Seção 4.2.1. A conclusão desta etapa representaria o alcance de todos os objetivos inicialmente propostos no trabalho.

A implementação atual exige que as RDDs sejam redes Booleanas. Então, a fim de dar alguma generalidade ao método, seria interessante admitir entradas onde as RDDs são funções lógicas com domínio multi-valor. Daí que um outro possível desdobramento do trabalho corresponde a desenvolver

um procedimento de pré-processamento, por exemplo baseado no trabalho de [Dubrova, Teslenko e Ming \(2010\)](#), que converta funções lógicas multi-valor para Funções Booleanas.

Outra possível e interessante extensão do trabalho corresponde ao desenvolvimento de uma solução paralela e distribuída para o “passo um” (computação dos atratores locais). Quando consideramos a classe de instâncias do problema geral onde os sinais de acoplamento são fixos (que é afinal o escopo desse trabalho), não há relações de dependência entre atratores locais oriundos de diferentes RDDs – desde que consideremos todos os possíveis sinais de acoplamento fixos. Daí que torna-se “natural” a possibilidade de desenvolvimento de uma solução de alto-desempenho onde o cálculo dos atratores locais seja distribuído entre diversos núcleos de processamento.

Para romper a limitação a que nos referimos acima (Seção 5.2), 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 Problem* que não são de pior caso para o HSP-solver de [Carastan-Santos et al. \(2017\)](#), o que permitiria usá-lo sem a necessidade de promover maiores modificações estruturais.

A atual implementação do gerador de RDDAs contempla apenas o tratamento de parâmetros mínimos à geração de RDDAs. Um outro interessante caminho a seguir corresponde à evolução desse gerador para uma plataforma mais geral de apoio à modelagem e à experimentação com RDDAs. Essa plataforma poderia, entre outros recursos, permitir: geração de RDDAs cujo padrão de conexão inter-RDDs corresponda a diversos modelos de redes, incluindo redes complexas; manipulação e tratamento de diversos tipos de RDDs e de funções de acoplamento; ferramentas para visualização amigável de RDDs, RDDAs e resultados dos experimentos; interfaces para integração com outras plataformas que trabalham com redes de elementos interagentes; etc.

5.4 Artigos Publicados

Artigo publicado nos anais de congresso internacional (ver Anexo A):

Finding Attractors in Biological Models Based on Boolean Dynamical Systems Using Hitting Set, Carlos R. P. Tovar, Eloi Araujo, Danilo Carastan-Santos, David C. Martins-Jr, Luiz C. S. Rozante, In IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), Athens, Greece, 2019.

ANEXO A – Artigo BIBE2019

Finding Attractors in Biological Models Based on Boolean Dynamical Systems Using Hitting Set

1st Carlos R. P. Tovar
Center for Mathematics,
Computing and Cognition
Federal University of ABC
Santo Andre-SP, Brazil
carlos.reynaldo@ufabc.edu.br

2nd Eloi Araujo
Cidade Universitária
Federal University of Mato Grosso do Sul
Campo Grande, Mato Grosso do Sul, Brazil
feloi@facom.ufms.br

3rd Danilo Carastan-Santos
Center for Mathematics,
Computing and Cognition
Federal University of ABC
Santo Andre-SP, Brazil
danilo.santos@ufabc.edu.br

4th David C. Martins-Jr
Center for Mathematics, Computing and Cognition
Federal University of ABC
Santo Andre-SP, Brazil
david.martins@ufabc.edu.br

5th Luiz C. S. Rozante
Center for Mathematics, Computing and Cognition
Federal University of ABC
Santo Andre-SP, Brazil
luiz.rozante@ufabc.edu.br

Abstract—Boolean networks are discrete-time dynamic systems that have been used as a model for a wide range of applications in different areas, especially in Systems Biology. The analysis of Boolean networks includes the search for attractors, which may represent important biological conditions such as gene expression patterns in models of gene regulatory networks, among others. Attractors can be found through exploring the network paths by achieving the solution to the SAT problem, which is known to be NP-complete. In this paper, we propose an approach to find all attractors by first transforming the corresponding instance of the SAT problem to a Hitting Set instance in linear time through a new direct linear reduction. Finally, the instance of the Hitting Set problem is solved by applying a fast parallel algorithm implemented in GPU. As a proof of principle, we tested the method for Boolean networks with 3 and 4 variables, returning the result in about 3 seconds and 9 hours respectively. However, for larger networks the execution time grows substantially due to the algorithm used in the Hitting Set problem solver. But the result achieved for networks with 3 and 4 variables encourages improvements in the method for dealing with large-scale Boolean networks, specially by incorporating some parameter restrictions based on prior information about the state diagram transition graphs structure and optimizing the method by means of dynamic programming and parallelism.

Index Terms—Hitting Set, Boolean Network, Attractors, Systems Biology, GPU

I. INTRODUCTION

Boolean networks are discrete-time dynamical systems composed by Boolean state variables, which have been extensively used as model for a wide range of applications in areas ranging from physics to biology, especially in Systems Biology. For example, Boolean networks have been used to model processes involving cellular signaling networks in plants [1], programmed cell death in yeast [2], cell differentiation [3] and gene regulatory networks [4], among others.

Supported by CAPES Finance Code 1811952, CNPq grant #307145/2017-4, and FAPESP grant #2015/01587-0.

As in Boolean networks the state space is finite (2^n elements where n is the number of variables) and the transitions are represented by functions, all the trajectories – sequence of consecutive states – of the system converge to either a single state, or a cycle of states, generally named attractor. Due to its self-stabilizing nature, attractors can provide good insights into the functional characteristics of the system. For example, since attractors are discrete self-stabilizing states, they naturally capture the stability of the gene expression profiles associated with the cell types. Therefore, they help to explain how cells can assume distinct phenotypes determined by complex gene expression profiles.

In this work, we present a new algorithm to find attractors in synchronous Boolean networks, based on an innovative approach which checks valid paths in State Transitions Graph (STG for short) using the Hitting Set problem (HSP for short). Similarly to Dubrova and Teslenko algorithm [5], it identifies potential paths in the STG using logical expressions (propositional formulas), however, we use an HSP-solver instead of a traditional SAT-solver to validate these paths. This approach requires a reduction from SAT to HSP. In this way, we develop a linear time procedure for converting SAT instances to HSP instances. Besides, we adapt the parallel GPU-based Carastan-Santos *et al* algorithm [6] to work as an HSP-solver.

II. RELATED WORK

Several studies use attractors in Boolean networks to model and analyze dynamic events in important applications, for example: Huang *et al* [7], who argue that cancer cells are trapped in abnormal attractors, among other ideas in cancer biology involving attractors; Fumia and Martis [8], who use Boolean dynamical system attractors to identify stationary protein activation patterns in signaling pathways involved in cancer; Davidich and Bornholdt [9], who simulate and reproduce – with attractors – the known activity sequence

of regulatory proteins along the living cell cycle in yeast (*Schizosaccharomyces pombe*); Benítez *et al* [10], who model and analyze the formation of robust cellular patterns (spatial dynamics modeled as attractors) in *Arabidopsis epidermis*; among others. Hence, due to its relevance, several methods to identify attractors in Boolean networks have already been proposed, which can be grouped as follows:

- Enumerative or partially enumerative methods, such as that proposed by Berntsen and Ebeling [11], which operates on selected subspaces of the network state space.
- Simulation-based methods that attempt to find attractors by heuristically choosing several initial states and simulating the evolution of the system for each initial condition [12]. These methods have the potential to fail to cover all attractors due to the random generation of the initial states.
- Methods based on problem formulation in terms of the binary decision diagram (BDD) problem [13], which is a data structure used to describe Boolean functions and support the computation of logical operations, whose size grows exponentially with the number of variables in the network and in the representation of the functions, hence limiting its application only to very simple networks due to the excessive memory requirements involved.
- Methods based on the formulation of the aggregation problem [14], whose main idea is to partition the input Boolean network into subnets, and to each subnet is applied an algorithm (for example the Johnson’s algorithm [15]) to find attractors.
- Methods based on the formulation of the problem in terms of the Satisfiability (SAT) problem [5]. These SAT-based methods require less space than those based on BDD, and in general lead to more efficient results - in terms of time - because they operate without computing the entire state space. These algorithms use a SAT-solver to identify paths in the state transition graph (STG).
- Methods based on the use of semi-tensor product (STP) of matrices [16], whose basic idea is to create a matrix representation of variables and logical functions of Boolean network. From the analysis and manipulation of the algebraic properties of this matrix it is possible to identify structures (for example attractors) of STG.

In addition, there are methods [17] focused on high-performance techniques whose goal is to identify attractors in “larger” networks (hundreds of variables). There are also methods that exploit properties of specific classes of Boolean networks. For instance, Akutsu *et al* [18] proposed a method to find small attractors (period 2) in Boolean networks consisting of n OR functions of positive literals. Finally, methods have already been developed whose goal is to identify attractors with particular characteristics, such as singleton attractors and small attractors [19] or fixed length attractors [20].

III. PRELIMINARIES

Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be a set of variables that assume values in a finite set X . Let $\mathcal{F} = [f_1, f_2, \dots, f_n]$ be functions

such that $f_i : X^n \rightarrow X$ determines the state of the variable x_i , $1 \leq i \leq n$. The *discrete dynamic system* (X^n, \mathcal{F}) consisting of state space X^n and the mapping \mathcal{F} is said to be a *Boolean network* if $X = \{0, 1\}$ and if the functions in \mathcal{F} are Boolean logic functions. Assuming discrete time steps and synchronous updating of variables in \mathcal{X} , the next state (time $t + 1$) of a Boolean network is given by the transition functions \mathcal{F} applied to the current state (time t) as follows:

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

The complete dynamics of a Boolean network can be represented by a directed graph $(X^n, \mathcal{F}) \equiv (X^n, E)$, where $y \rightarrow z \in E$ if and only if $\mathcal{F}(y) = z$. This graph is named *State Transitions Graph* (STG). We denote a path with k edges from x in STG by $\mathcal{F}^k(x)$, i. e., $\mathcal{F}^k(x)$ is x if $k = 0$ and it is $\mathcal{F}^{k-1}(\mathcal{F}(x))$ if $k > 0$. A state $x \in X^n$ is a *point attractor* (or *steady state*) if $\mathcal{F}(x) = x$ in the STG and we say that $\{x, \mathcal{F}^1(x), \mathcal{F}^2(x), \dots, \mathcal{F}^k(x) = x\}$ is a *periodic attractor*.

The problem tackled by this paper is – given a synchronous Boolean network (X^n, \mathcal{F}) – to find the set of all attractors (steady states and/or all states in periodic attractors), denoted by \mathcal{A} . This problem has been shown to be NP-hard [21].

The state x of a Boolean network at time t is represented by a vector $x^t = [x_1^t, x_2^t, \dots, x_n^t]$ of state-variables.

We denote by S_k a sequence of transitions

$$x^{t+0} \rightarrow x^{t+1} \rightarrow x^{t+2} \rightarrow \dots \rightarrow x^{t+k}$$

with k time steps (from step t to step $t + k$).

A transition S_1 (the same as $x^t \rightarrow x^{t+1}$) according to Burch *et al* [22] is given by

$$[x_1^{t+1} \leftrightarrow f_1(x^t)] \wedge \dots \wedge [x_n^{t+1} \leftrightarrow f_n(x^t)], \quad (2)$$

where f_i , $1 \leq i \leq n$, is as defined above, and “ \wedge ” and “ \leftrightarrow ” denote the logical operators for conjunction and biconditional, respectively. This representation of the transition S_1 (equation 2) is named *propositional formulation* of S_1 , which we denote by \widehat{S}_1 .

If there is a satisfying assignment for the variables $x_1^t, \dots, x_n^t, x_1^{t+1}, \dots, x_n^{t+1}$ in \widehat{S}_1 , then the transition $x^t \rightarrow x^{t+1}$ corresponds to an edge in the STG. Thus, a sequence S_k of transitions corresponds to a valid path in the STG if there is a satisfying assignment for the variables in \widehat{S}_k .

A definition of the *Hitting Set problem* (HSP) is as follows. Given a finite set \mathbb{X} , a collection \mathcal{S} of subsets of \mathbb{X} , the goal is to find a subset $H \subseteq \mathbb{X}$ with smallest size such that:

$$H \cap S \neq \emptyset, \forall S \in \mathcal{S}. \quad (3)$$

IV. HITTING SET PROBLEM BASED APPROACH

The key idea of our algorithm consists in deriving sequences of transitions and checking whether they correspond to valid paths in the STG using Hitting Set. This task requires conceiving a reduction from SAT to HSP, i.e., we develop a linear time

procedure for converting SAT instances to HSP instances (see section IV-A).

Let r be a Boolean network of n variables and a sequence of transitions containing \widehat{S}_k variables in its corresponding propositional formulation \widehat{S}_k in STG. Notice that $\mathcal{N} = n \times (k+1)$. We convert \widehat{S}_k into an instance of HSP and if a procedure named `HSPsolver` returns a solution H and $|H| = \mathcal{N}$, it means that there is a satisfying assignment for the variables in \widehat{S}_k , and therefore \widehat{S}_k corresponds to a valid path in the STG (see section III).

If \widehat{S}_k corresponds to a sequence of transitions of a valid path in the STG, then it is checked whether S_k contains a loop. If so, the loop vertices are stored in solution set (list of attractors) and they are removed from the search space; otherwise, the size of the sequence S_k is incremented and the process is repeated iteratively.

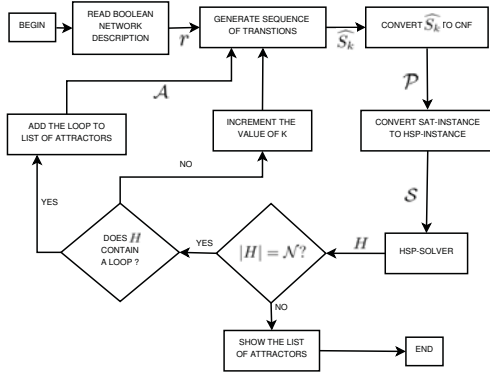


Fig. 1. Flowchart with the main steps of the algorithm.

Fig. 1 illustrates the main steps of the algorithm, which is described more precisely through Algorithm 1.

A. A simple and linear reduction from SAT to HSP problem

Following the above definition, given a set \mathbb{X} and a collection \mathcal{S} of subsets of \mathbb{X} , the Hitting Set problem (HSP) consists in finding a set $A \subseteq \mathbb{X}$, such that $A \cap S \neq \emptyset$ for all $S \in \mathcal{S}$ and $|A|$ is minimum. We assume that $\mathbb{X} = \cup_{S \in \mathcal{S}} S$ which implies that an algorithm that solves HSP needs only \mathcal{S} as argument. We call the algorithm that solves HSP by `HSPsolver`.

Let \mathcal{X} be an ordered set of \mathcal{N} variables. In what follows, we denote the *literals* of i -th variable in \mathcal{X} by x_i and $\neg x_i$. A *clause* is a disjunction of literals. A propositional formula is in a *conjunctive normal form* (CNF) if it is a conjunction of clauses. An *interpretation* of \mathcal{X} is a set $L = \{L_1, \dots, L_{\mathcal{N}}\}$, where L_i is a literal of the i -th variable of \mathcal{X} , or in other words, L is a set having exactly one literal of each variable in \mathcal{X} . The interpretation L *satisfies* a propositional formula \mathcal{P} in a CNF if every clause in \mathcal{P} has a literal in L . The *Boolean Satisfiability problem* (SAT) consists in deciding if there exists an interpretation L of \mathcal{X} that satisfies a given propositional formula \mathcal{P} in CNF.

In this section, we show the algorithm `ConvertSATtoHSP` that transforms an instance of SAT problem into an instance

of HSP in $\Theta(N)$ time, where N is the size of the instance of SAT problem.

Given the propositional formula \mathcal{P} , `ConvertSATtoHSP`(\mathcal{P}) returns the collection

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

where \mathcal{C} is a collection of literal sets corresponding to clauses of \mathcal{P} , $|\mathcal{C}| = m$, and \mathcal{V} is the collection of all variable sets, i. e., $\mathcal{V} = \{\{x_i, \neg x_i\} : x_i \in \mathcal{X}\}$, $|\mathcal{V}| = \mathcal{N}$. Thus, $\mathcal{N} + m$ sets of \mathcal{S} is given as input to `HSPsolver`. Figure 2 shows an example of an HSP instance obtained from a SAT instance.

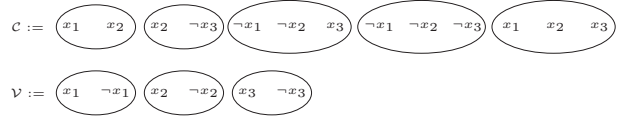


Fig. 2. Examples of sets \mathcal{C} and \mathcal{V} whose union is the instance of HSP from $\mathcal{P} = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$. In this case, notice that there exists an interpretation $(\neg x_1, x_2, x_3)$ satisfying \mathcal{P} since every set in $\mathcal{C} \cup \mathcal{V}$ has at least one literal from $\{\neg x_1, x_2, x_3\}$. Notice that \mathcal{N} represents an \mathcal{P} interpretation.

Next, we show the following straightforward result.

Proposition 1. *Let \mathcal{P} be an instance of SAT with \mathcal{N} variables in \mathcal{X} , `ConvertSATtoHSP`(\mathcal{P}) = \mathcal{S} and `HSPsolver`(\mathcal{S}) = H . Then, H is an interpretation of \mathcal{X} that satisfies \mathcal{P} if and only if $|H| = \mathcal{N}$. Besides `ConvertSATtoHSP` is a linear procedure.*

Proof. (\Rightarrow) Suppose that H is an interpretation of \mathcal{X} that satisfies \mathcal{P} . Since H is an interpretation of \mathcal{X} and \mathcal{P} has \mathcal{N} variables, we have by definition that $|H| = \mathcal{N}$.

(\Leftarrow) Suppose that $|H| = \mathcal{N}$. Let \mathcal{V} and \mathcal{C} be sets of \mathcal{S} . Since H has an element of each set in $\mathcal{V} \cup \mathcal{C}$, we have $H \cap V \neq \emptyset$ for each $V \in \mathcal{V}$ and $H \cap C \neq \emptyset$ for each $V \in \mathcal{C}$. Since $|H| = |\mathcal{V}| = |\mathcal{X}| = \mathcal{N}$ and $H \cap V \neq \emptyset$ for each $V \in \mathcal{V}$, we have that H is a set having exactly one literal of each variable in \mathcal{X} which implies that H is an interpretation of \mathcal{X} . Also, since $H \cap C \neq \emptyset$ for each $V \in \mathcal{C}$, we have that every clause in \mathcal{P} has a literal in H which implies that H is an interpretation of \mathcal{X} that satisfies \mathcal{P} .

Besides, clearly `ConvertSATtoHSP` is a linear procedure. \square

V. IMPLEMENTATION

Regarding the initial size (k) of the sequence of transitions \widehat{S}_k (line 1 of the Algorithm 1) and the way we increase the size of the transition sequences (line 19 of the Algorithm 1), we follow Dubrova and Teslenko [5].

Suppose a sequence \widehat{S}_k of state transitions

$$x^{t+0} \rightarrow x^{t+1} \rightarrow x^{t+2} \rightarrow \dots \rightarrow x^{t+k}$$

is a valid path in the STG. Then, to check whether \widehat{S}_k contains a loop, the algorithm only needs to check whether the last state (x^{t+k}) occurs at least twice in \widehat{S}_k (lines 11 to 16 of Algorithm 1). Note that we represent the sequence of states (satisfying assignment) corresponding to a valid path in the STG in

Algorithm 1 Find Attractors using Hitting Set

Input: a Boolean network r with n variables.

Output: the vertices of the STG that belong to its attractors.

```
1:  $k := n$  /*  $n$  is the number of variables in  $r$  */
2:  $\widehat{S}_k := \text{genPropositionalFormulation}(r, k)$ 
3:  $\mathcal{P} := \text{convertCNF}(\widehat{S}_k)$ 
4:  $\mathcal{S} := \text{convertSATtoHSP}(\mathcal{P})$ 
5:  $H := \text{HSPsolver}(\mathcal{S})$ 
6:  $\text{loopFound} := \text{false}$ 
7:  $\mathcal{A} := \text{false}$ 
8: while  $|H| = n(k+1)$  do
9:    $\text{Path} := \text{transformToListOfStates}(H)$ 
10:  for  $i = 0$  to  $k-1$  do
11:    if  $\text{Path}[i] = \text{Path}[k]$  then
12:       $\mathcal{B} := \text{genBooleanFormulation}(\text{Path}[i], \text{Path}[k])$ 
13:       $\mathcal{A} := \mathcal{A} \vee \mathcal{B}$ 
14:       $\text{loopFound} := \text{true}$ 
15:      breakfor
16:    end if
17:  end for
18:  if  $\text{loopFound} = \text{false}$  then
19:     $k := k * 2$  /* double the length ( $k$ ) of the transitions
    sequence  $\widehat{S}_k$  */
20:  end if
21:   $\widehat{S}_k := \text{genPropositionalFormulation}(r, k)$ 
22:   $\widehat{S}_k := \widehat{S}_k \wedge \neg \mathcal{A}$  /*exclude from the search space the
  last computed attractor*/
23:   $\mathcal{P} := \text{convertCNF}(\widehat{S}_k)$ 
24:   $\mathcal{S} := \text{convertSATtoHSP}(\mathcal{P})$ 
25:   $H := \text{HSPsolver}(\mathcal{S})$ 
26:   $\text{loopFound} := \text{false}$ 
27: end while
28: return  $\mathcal{A}$ 
```

a data structure returned by `transformToListOfStates(H)` and named Path , where $\text{Path}[i]$, $0 \leq i \leq k$, denotes the state x^{t+i} in \widehat{S}_k . Thus, if $\text{Path}[i] = \text{Path}[k]$, $0 \leq i \leq k-1$, it means the last state (x^{t+k}) has already occurred and therefore there is a loop in \widehat{S}_k .

Once an attractor has been identified, it must be assigned to the solution set \mathcal{A} (line 13 of the Algorithm 1). Another important detail concerns how we remove from the search space the attractors already found (line 22 of the Algorithm 1). For these procedures we also use the same techniques used by Dubrova and Teslenko. More details can be obtained in [5].

One advantage of the proposed Hitting Set based attractor search is that it enables to use Hitting Set algorithms that are tailored for High Performance Computing (HPC), notably hybrid CPU-GPU computation. In this regard, we adopted the Hitting Set algorithm proposed by Carastan-Santos *et al.* [6]. This algorithm follows an enhanced exhaustive search of Hitting Sets, in which candidate solutions – which are encoded as combinations of variables of \mathbb{X} (See Section IV-A) – are enumerated and evaluated in increasing order of cardinality

i , $1 \leq i \leq k$ and stops when a solution is found. The evaluation process is a disjunction check with the clauses in a sorted collection `SortS`, which is the collection \mathcal{S} whose clauses are sorted in increasing order of their sizes. The evaluation of the candidate solution with `SortS` allows an efficient discarding of non solution candidates. This process can be efficiently done in parallel by multiple processing units (CPUs and GPUs) by a master-slave approach, where the master assigns candidate solutions to the slaves and they independently search for Hitting Set solutions among its assigned candidates.

Our linear SAT-HSP reduction process creates Hitting Set instances whose solutions (if they exist) must have a size k of at least \mathcal{N} , and for finding attractors it is enough to find one solution of a given instance, as opposed to all of the solutions. At this light, we modified the Hitting Set algorithm of Carastan-Santos *et al.* [6] such that (i) the search algorithm starts with candidate solutions with size $k = \mathcal{N}$ and (ii) the Hitting Set search terminates as soon as a slave finds a solution.

VI. PRELIMINARY EXPERIMENTS

We run the algorithm with the following Boolean networks:

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

and

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

As expected, the attractor states returned for the first network (3 variables) were [001], [111, 110] and for the second network (4 variables) the attractor states returned were [1101], [1011, 0101, 1010, 0011, 0111].

These experiments were done as a proof of concept, showing that it is possible to find attractors using Hitting Set.

The algorithm was implemented in Python 2.7 and the experiments were executed using O.S. Ubuntu 18.04 on a desktop with Intel® Core(TM) i7-3930K processor, 3.20GHz, 12 CPUs, 32 GB RAM and two Nvidia® GK 104 GeForce GTX boards. Under this setup, for the experiment with 3 variables aforementioned, the execution time was 3.55 sec., but for the 4 variables network experiment the execution time was 9.15 hours.

As mentioned earlier, the problem in question has already been shown to be NP-Hard [21], hence impracticable response times are expected when n (number of variables in Boolean network) tends to huge values. However, there are many applications in Systems Biology where the problem inputs are of reasonable size, i.e., computationally treatable, for which we make the following performance considerations.

The method's response time is strongly dependent on the `HSPsolver` runtime, which receives inputs whose size depends mainly on the:

- number of variables in the logical sentences \widehat{S}_k , which is given by $2n(k+1)$, where n is the number of variables of the Boolean network and k is the size of the sequence of transitions;
- number of clauses in the logical sentences \widehat{S}_k , which depends on the \mathcal{F} functions of the Boolean network;
- number of transitions k , which depends directly on the STG diameter, the number of attractors, and the initial set of states chosen internally by the HSPsolver, implying that runtimes for the same Boolean network may vary for different runs.

The HSPsolver can be optimized in future versions by adapting it to efficiently solve the problem of Satisfiability.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented a new algorithm based on Hitting Set for detection of attractors in synchronous Boolean networks. We adopted an innovative approach based on the use of Hitting Set to verify the existence of valid paths in the STG, which required the development of a linear procedure to convert SAT instances to HSP instances, i.e., a reduction from SAT to HSP. Farther, we modified the Carastan-Santos' method to work as an HSPsolver.

In order to complete the proof of principle, showing that it is possible to identify attractors in Boolean networks using Hitting Set, we implemented our algorithm and performed preliminary experiments using small (3 and 4 variables) networks, whose attractors we knew in advance. The implementation responded according to the expected results with reasonable execution time, specially considering that the experiments were run in a desktop.

The method proposed here was just a first-step towards fast recovering of attractors from Boolean networks. However improvements are still required to become the method feasible for large scale networks, such as using dynamic programming to store transitions previously calculated, and adding constraints on method parameters to avoid an excessive number of variables and clauses provided to the HSP-solver. These restrictions could be based on prior information about the STG structure provided by the target application.

In addition, it is important to note that an HSP-based method for detecting attractors would allow the use of high-performance algorithms (such as the Carastan-Santos' algorithm [6]) to find attractors in large scale networks. This would be of great interest to a large community of researchers working with Boolean networks.

ACKNOWLEDGMENT

We would like to specially thank Carlos Fernando Montoya Cubas for the fruitful discussions that clarified some concepts. This work was financially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 1811952, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) grant #307145/2017-4, and São Paulo Research Foundation (FAPESP) grant #2015/01587-0.

REFERENCES

- [1] S. Li, S. M. Assmann, and R. Albert, "Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling," *PLoS biology*, vol. 4, no. 10, p. e312, 2006.
- [2] L. Kazemzadeh, M. Cvijovic, and D. Petranovic, "Boolean model of yeast apoptosis as a tool to study yeast and human apoptotic regulations," *Frontiers in physiology*, vol. 3, p. 446, 2012.
- [3] M. Villani, A. Barbieri, and R. Serra, "A dynamical model of genetic networks for cell differentiation," *PLoS one*, vol. 6, no. 3, p. e17703, 2011.
- [4] F. F. Borelli, R. Y. de Camargo, D. C. Martins, and L. C. Rozante, "Gene regulatory networks inference using a multi-GPU exhaustive search algorithm," *BMC Bioinformatics*, vol. 14, 2013.
- [5] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 5, pp. 1393–1399, 2011.
- [6] D. Carastan-Santos, R. Y. de Camargo, D. C. Martins, S. W. Song, and L. C. Rozante, "Finding exact hitting set solutions for systems biology applications using heterogeneous GPU clusters," *Future Generation Computer Systems*, vol. 67, pp. 418–429, 2017.
- [7] S. Huang, I. Ernberg, and S. Kauffman, "Cancer attractors: A systems view of tumors from a gene network dynamics and developmental perspective," pp. 869–876, 2009.
- [8] H. F. Fumia and M. L. Martins, "Boolean network model for cancer pathways: predicting carcinogenesis and targeted therapy outcomes," *PLoS one*, vol. 8, no. 7, p. e69008, 2013.
- [9] M. I. Davidich and S. Bornholdt, "Boolean network model predicts cell cycle sequence of fission yeast," *PLoS ONE*, vol. 3, no. 2, 2008.
- [10] M. Benítez, C. Espinosa-Soto, P. Padilla-Longoria, and E. R. Alvarez-Buylla, "Interlinked nonlinear subnetworks underlie the formation of robust cellular patterns in Arabidopsis epidermis: A dynamic spatial model," *BMC Systems Biology*, vol. 2, 2008.
- [11] N. Berntsen and M. Ebeling, "Detection of attractors of large Boolean networks via exhaustive enumeration of appropriate subspaces of the state space," *BMC Bioinformatics*, vol. 14, no. 1, 2013.
- [12] S. Karl and T. Dandekar, "Jimena: Efficient computing and system state identification for genetic regulatory networks," *BMC Bioinformatics*, vol. 14, 2013.
- [13] A. Di Cara, A. Garg, G. De Micheli, I. Xenarios, and L. Mendoza, "Dynamic simulation of regulatory networks using SQUAD," *BMC Bioinformatics*, vol. 8, 2007.
- [14] Y. Zhao, J. Kim, and M. Filippone, "Aggregation algorithm towards large-scale boolean network analysis," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 1976–1985, 2013.
- [15] D. B. Johnson, "Finding All the Elementary Circuits of a Directed Graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, 2005.
- [16] D. Cheng and H. Qi, "A linear representation of dynamics of Boolean networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2251–2258, 2010.
- [17] W. Guo, G. Yang, W. Wu, L. He, and M. Sun, "A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks," *PLoS ONE*, vol. 9, no. 4, 2014.
- [18] T. Akutsu, S. Kosub, A. A. Melkman, and T. Tamura, "Finding a periodic attractor of a Boolean network," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 5, pp. 1410–1421, 2012.
- [19] S. Q. Zhang, M. Hayashida, T. Akutsu, W. K. Ching, and M. K. Ng, "Algorithms for finding small attractors in boolean networks," *Eurasip Journal on Bioinformatics and Systems Biology*, vol. 2007, 2007.
- [20] X. Y. Li, G. W. Yang, D. S. Zheng, W. S. Guo, and W. Hung, "An efficient algorithm for computing fixed length attractors based on bounded model checking in synchronous boolean networks with biochemical applications," *Genetics and Molecular Research*, vol. 14, no. 2, pp. 4238–4244, 2015.
- [21] D. S. Johnson and M. R. Garey, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman San Francisco, 1979, vol. 1.
- [22] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 1020 States and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

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.
- CORMEN, T. H. et al. Introduction to algorithms, 2nd edn cambridge. MA: *The MIT Press.*, p. 420, 2001.
- 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.
- ENDERTON, H.; ENDERTON, H. B. *A mathematical introduction to logic*. [S.l.]: Elsevier, 2001. 24–26 p.
- GAREY, M. R.; JOHNSON, D. S. *Computers and intractability*. [S.l.]: freeman San Francisco, 1979. v. 174.
- 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.
- KARP, R. M. Reducibility among combinatorial problems. In: *Complexity of computer computations*. [S.l.]: Springer, 1972. p. 85–103.
- 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.
- NIEDERMEIER, R.; ROSSMANITH, P. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, Elsevier, v. 1, n. 1, p. 89–102, 2003.
- 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.

- RUSSEL, S.; NORVIG, P. et al. *Artificial intelligence: a modern approach*. Third. [S.l.]: Pearson Education Limited, 2010. 253–254 p.
- 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.
- WUENSCHÉ, A. Genome regulation modeled as a network with basins of attraction. *Complexity*, v. 4, n. 3, p. 47–66, 1998.
- WUENSCHÉ, 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.