UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

FÁBIO SEBASTIAN SILVEIRA

ESCALABILIDADE DO PROBLEMA DE GERAÇÃO DE ESTRUTURAS DE COALIZÃO: APLICAÇÃO DE UM ALGORITMO BASEADO EM DETECÇÃO DE COMUNIDADES A GRAFOS REAIS

DISSERTAÇÃO

CURITIBA

2017

FÁBIO SEBASTIAN SILVEIRA

ESCALABILIDADE DO PROBLEMA DE GERAÇÃO DE ESTRUTURAS DE COALIZÃO: APLICAÇÃO DE UM ALGORITMO BASEADO EM DETECÇÃO DE COMUNIDADES A GRAFOS REAIS

Dissertação apresentada ao Programa de Pós-graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Computação Aplicada" – Área de Concentração: Engenharia de Sistemas Computacionais.

Orientador: Gustavo Alberto Gimenéz-Lugo

CURITIBA

2017

Dados Internacionais de Catalogação na Publicação

S587e 2017

Silveira, Fábio Sebastian
Escalabilidade do problema de geração de estruturas de coalizão : aplicação de um algoritmo baseado em detecção de comunidades a grafos reais / Fábio Sebastian Silveira.-- 2017.
68 f.: il.; 30 cm.

Disponível também via World Wide Web.
Texto em português, com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. Área de Concentração: Engenharia de Sistemas Computacionais, Curitiba, 2017.
Bibliografia: f. 65-68.

1. Sistemas multiagentes. 2. Algoritmos heurísticos. 3. Teoria dos grafos. 4. Agentes inteligentes (Software). 5. Inteligência artificial distribuída. 6. Métodos de simulação. 7. Engenharia de sistemas. 8. Computação – Dissertações. I. Giménez Lugo, Gustavo Alberto, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. III. Título.

CDD: Ed. 22 -- 621.39

Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 53

Aos 15 dias do mês de agosto de 2017, realizou-se na sala B205 a sessão pública de Defesa da Dissertação de Mestrado intitulada **"Escalabilidade do Problema de Geração de Estruturas de Coalizão: Aplicação de um Algoritmo Baseado em Detecção de Comunidades a Grafos Reais"**, apresentado pelo aluno **Fábio Sebastian Silveira** como requisito parcial para a obtenção do título de Mestre em Computação Aplicada, na área de concentração "Engenharia de Sistemas Computacionais", linha de pesquisa "Sistemas Inteligentes e Lógica".

Constituição da Banca Examinadora:

Gustavo Alberto Giménez Lugo (Orientador e Presidente) – UTFPR _____

André Luís Vignatti – UFPR _____

Cesar Augusto Tacla - UTFPR _____

Heitor Silvério Lopes . – UTFPR _____

Em conformidade com os regulamentos do Programa de Pós-Graduação em Computação aplicada e da Universidade Tecnológica Federal do Paraná, o trabalho apresentado foi considerado _____ (aprovado/reprovado) pela banca examinadora. No caso de aprovação, a mesma está condicionada ao cumprimento integral das exigências da banca examinadora, registradas no verso desta ata, da entrega da versão final da dissertação em conformidade com as normas da UTFPR e da entrega da documentação necessária à elaboração do diploma, em até _____ dias desta data.

Ciente (assinatura do aluno): _____

(para uso da coordenação)

A Coordenação do PPGCA/UTFPR declara que foram cumpridos todos os requisitos exigidos pelo programa para a obtenção do título de Mestre.

Curitiba PR, _____/_____/_____ _____

"A Ata de Defesa original está arquivada na Secretaria do PPGCA".

RESUMO

Silveira, Fábio Sebastian. ESCALABILIDADE DO PROBLEMA DE GERAÇÃO DE ESTRUTURAS DE COALIZÃO: APLICAÇÃO DE UM ALGORITMO BASEADO EM DETECÇÃO DE COMUNIDADES A GRAFOS REAIS. 68 f. Dissertação – Programa de Pós-graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Este estudo apresenta resultados experimentais sobre a escalabilidade da formação de estruturas de coalizão para grafos reais que passam de 5 mil vértices. Um algoritmo heurístico simples denominado Algoritmo de Propagação para Formação de Estrutura de Coalizão (APFEC) com garantias experimentais é apresentado e sondado, com base em uma versão de propagação balanceada de rótulos para detecção de comunidades em grafos muito grandes. Os limites da proposta são avaliados, comparando-o com o estado-da-arte em relação aos algoritmos exatos (ODP-IP - A junção do algoritmo IP, baseado em representação de partições de inteiros, e ODP, programação dinâmica ótima) e heurístico (CFSS - Formação de coligação para grafos esparsos). Os experimentos são executadas com um conjunto de 14 grafos do mundo real, e os resultados mostram que esta abordagem consegue calcular estruturas de coalizão de maneira rápida, mesmo na presença das limitações discutidas. Finalmente, os resultados preliminares são analisados considerando a influência da habilidade e a inter-relação entre os agentes na avaliação das coalizões.

Palavras-chave: Sistemas multi-agentes, Algoritmos em grafos, Coalizões

ABSTRACT

Silveira, Fábio Sebastian. SCALABILITY OF THE PROBLEM OF GENERATION OF COALITION STRUCTURES: APPLICATION OF AN ALGORITHM BASED ON THE DETECTION OF COMMUNITIES TO REAL GRAPHS. 68 f. Dissertação – Programa de Pós-graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

This study presents experimental results on the scalability of the coalition's structures formation for real graphs that go from 5 thousand vertices. A simple heuristic algorithm called Propagation Algorithm for Coalition Structure Formation (APFEC in Portuguese) with experimental guarantees is presented and probed, based on a balanced label propagation version for detection of communities in very large graphs. The limits of the proposal are evaluated, comparing it with the state-of-the-art in relation to the exact algorithms (ODP-IP - The IP algorithm, based on representation of integer partitions, and ODP, optimal dynamic programming) and heuristic (CFSS - Coalition Formation for Sparse Synergies). The experiments are performed with a set of 14 real-world graphs, and the results show that this approach can calculate coalition structures quickly, even in the presence of the limitations discussed. Finally, the preliminary results are analyzed considering the influence of the ability and the interrelation between the agents in the evaluation of the coalitions.

Keywords: Multi-agent system, Graphs algorithms, Coalition

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

APFEC	Algoritmo de Propagação para Formação de Estrutura de Coalizão
CSGP	Acrônimo em inglês para Problema de Geração de Estrutura de Coalizão
CSPP	Acrônimo em inglês para Problema de Partição Completa de Conjunto
OCSG	Acrônimo em inglês para Problema de Geração de Estrutura de Coalizão Ótima
CSS1	Acrônimo em inglês para o algoritmo Busca de Estruturas de Coalizões
ODP	Acrônimo em inglês para o algoritmo Programação Dinâmica Ótima
IP	Acrônimo em inglês para o algoritmo Partição de Inteiros
IDP	Acrônimo em inglês para o algoritmo Programação Dinâmica Melhorada
TAVCF	Acrônimo em inglês para o algoritmo Alocação de Tarefas Via Formação de Coalizões
OBGA	Acrônimo em inglês para o algoritmo Algoritmo Genético Baseado em Ordem
SAMCF	Acrônimo em inglês para o algoritmo Arrefecimento Simulado para Formação de Coalizões em Multi-agentes
WSG	Acrônimo em inglês para o algoritmo Grafo de Sinergia Ponderado

LISTA DE SÍMBOLOS

A	Um conjunto de indivíduos ou agentes.
P(A)	O conjunto potência de A
C	O conjunto de coalizões
n	A quantidade de agente dada por $\|A\|$
CS	Uma estrutura de coalizão
c	Uma coalizão do conjunto de coalizões C.
ρ	Peso para as inter-relações.
G	Um grafo não direcionado.
G	Um grafo não direcionado.
K_n	Grafo completo com n vértices.

SUMÁRIO

1 INTRODUÇÃO

É cada vez mais frequente a necessidade de agrupar indivíduos de maneira automática, aplicando os grupos obtidos a tarefas. Particionar um conjunto é uma forma de criar esses agrupamentos. Assim, dado um conjunto de indivíduos e um conjunto de tarefas, o conjunto de indivíduos pode ser particionado em subconjuntos, tal que cada um deles seria atrelado a uma tarefa específica.

Definição 1 *Conjunto Potência (powerset) de um conjunto A é o conjunto de todos os subconjuntos de A, denotado por P(A). (RAM, 2010). Note que* $|P(A)| = 2^{|A|}$.

Definição 2 *Partição de conjuntos (Set Partitioning): Dado um conjunto de n elementos* $A = \{1,...,n\}$ *e um conjunto de conjuntos de A,* $P(A) = \{C_1,...,C_m\}$*, tal que* $C_i \subseteq A$ *e* $|P(A)| = 2^{|A|}$ *e* $\forall C_i, C_j \in P', i \neq j, C_i \cap C_j = \emptyset$ *(SHEHORY; KRAUS, 1998). Cada subconjunto formado pela partição é nomeado de bloco.*

Dependendo das propriedades desses grupos eles recebem denominações distintas, alguns exemplos são: time, mercado, congregação, federação. A presente dissertação concentra-se no conceito denominado coalizão. Uma coalizão é um conjunto de agentes que coordenam suas atividades para alcançar um objetivo comum e a mesma existe apenas enquanto o objetivo não é alcançado (RAHWAN; JENNINGS, 2008). Uma das propriedades desejadas é que nenhum dos integrantes de uma coalizão tenha interesse em deixá-la (CHEVALEYRE et al., 2007).

Definição 3 *Uma coalizão é um par (A, v) no qual* $A = \{1,...,n\}$ *é um conjunto de jogadores e* $v : 2^{|A|} \rightarrow \mathbb{R}$ *é uma função característica (função de avaliação) que associa a cada coalizão* $C \subseteq A$ *um valor de ganho* $v(C)$ *no qual* $v(\emptyset) = 0$ *(AZIZ; KEIJZER, 2011).*

Uma partição valorada do conjunto de indivíduos é denominada de Estrutura de Coalizão e uma dificuldade adicional para sua escolha é o fato de que cada um de seus subconjuntos

possíveis tem um valor de ganho atribuído, e o ganho correspondente à partição é a somatória desses valores. A geração de coalizões é um problema computacional desafiador dada a necessidade exponencial de tempo ou memória, sendo classificada como NP-Difícil (AZIZ; KEIJZER, 2011). Essa complexidade computacional impõe uma barreira de tempo e esforço, e um dos desafios da formação de estruturas de coalizão é torná-la tratável, pois são feitas a todo momento: desde um conjunto de pessoas para limpar a vizinhança até ajudar outras em desastres. Esse tipo de tarefa não pode, nem deve demorar ou tornar-se um problema já que muitas vezes um sistema multi-agente tem tempo limitado para tomar suas decisões. Um exemplo claro dessa necessidade é o auxílio em desastres, que os organizadores devem formar conjunto de indivíduos que atuarão em diversas tarefas a fim de minimizar os danos causados. Mesmo sendo complexo calcular uma coalizão ótima, já existem algoritmos que o fazem para um pequeno número de agentes, outros gerando resultados satisfatórios a qualquer momento. Também foram criados outros algoritmos para tornar possível a geração de estruturas de coalizão em grafos com milhares de vértices, mas sem a garantia de encontrar a melhor solução.

Definição 4 *Uma estrutura de Coalizão CS é uma partição de N em um jogo de coalizão* (N, v)*, isto é* $|CS| = 2^n$*, tal que* $\forall C_i, C_j \in CS, i \neq j, C_1 \cap C_2 = \emptyset$*. O valor alcançado por uma estrutura de coalizão CS, denotado* $v(CS)$ *é definido por* $\sum_{C \in CS} v(C)$*. Uma Estrutura de Coalizão CS é tida como ótima quando* $v(CS) > v(CS') \forall CS'$ *(AZIZ; KEIJZER, 2011).*

Podem ser adicionadas restrições ao problema dado pela Definição 2 visando a tratabilidade computacional sendo uma delas a restrição que podem ser consideradas apenas as relações entre agentes expressas em um grafo. A complexidade computacional da geração de coalizões está relacionado a um problema clássico, mais especificamente com a Definição 2. Especificamente o particionamento de grafos, trata de avaliar a divisão de um conjunto qualquer em subconjuntos menores e, cada elemento do conjunto maior deve, necessariamente, pertencer a um subconjunto e a apenas um. Soluções que necessitem gerar o Conjunto Potência (*powerset*), no qual todas as combinações de vértices do conjunto são possíveis, são impraticáveis para um grande número de elementos. De fato, experimentos são chegam a processar 50 vértices. Caso seja aplicado a grafos extraídos de redes sociais mediadas computacionalmente como como o Facebook, que no final de 2014 estava com quase 1,4 bilhão de usuários ativos (HU et al., 2016) e o MSN, com 42 milhões (LESKOVEC; HORVITZ, 2008) ultrapassam esses valores, sendo que cada usuário ativo seria mapeado como um vértice no grafo, e cada amigo ou contato seria uma aresta neste grafo.

A necessidade de restringir a complexidade de geração deve-se não somente ao fato do tempo dispendido para calcular as mesmas, mas também ao fato de que, em cenários reais,

há restrições para a formação desses agrupamentos, sejam elas emocionais (um agente pode simplesmente não ter afinidade para trabalhar com outro), técnicas (no caso de dois agentes terem os mesmos conhecimentos; ou a falta ou limitações das habilidades, portanto não agregaria nada tê-los no mesmo grupo) ou físicas (agentes distantes fisicamente, o que impossibilitaria a formação do conjunto).

O principal limitador para o cálculo de estruturas de coalizão é a falta de escalabilidade dos algoritmos existentes. Esse problema se deve à combinação entre todos os agentes para formarem uma coalizão e as combinações destas coalizões para formarem uma estrutura. A complexidade computacional de gerar estruturas é exponencial no número de vértices do grafo (dada como entrada). Como a enumeração explícita de todas essas estruturas é inviável outras abordagens foram desenvolvidas, deixando a enumeração explícita e a necessidade de obter uma resposta exata. Uma abordagem alternativa são algoritmos heurísticos que abandonam o objetivo de otimalidade. Sendo assim, o problema tratado neste estudo é tornar possível o cálculo de estruturas de coalizão em grafos com milhares de vértices.

Problemas reais envolvendo formação de coalizões podem incluir dezenas e até centenas de milhares de indivíduos, como o particionamento igualitário que melhorou a recomendação de um serviço do Facebook através de detecção de comunidades utilizando propagação de rótulos, uma maneira de contágio que ocorre com os vizinhos de grau um do vértice analisado (UGANDER; BACKSTROM, 2013). O trabalho de (UGANDER; BACKSTROM, 2013) é um motivador já que mostra que modificando a modelagem do problema e de particionamento pode ser computado para grafos com milhares de vértices.

O presente trabalho pretende explorar modificações que poderiam ser introduzidas em algoritmos de formação de coalizões caso eles partissem para o conceito análogo a propagação de rótulos e detecção de comunidades. Esta dissertação tem o objetivo de responder as seguintes perguntas:

1. É possível, através de um mecanismo de propagação de rótulos, tornar escalável a geração de estruturas de coalizão para grafos com milhares de vértices?

2. O método de contágio pode ser caracterizado por um jogo onde os agentes querem maximizar seus ganhos?

3. Para evitar diminuir o valor da estrutura a função de agregação, que computa o valor da estrutura, poderia intervir no rótulo do agente?

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Verificar experimentalmente o ganho de escala, auferido pelo número de vértices para os quais é possível realizar o cálculo, decorrente do uso de uma heurística baseada em propagação de rótulos para o problema de geração de estruturas de coalizões.

1.1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo principal deste estudo foram criados os seguintes marcos como objetivos específicos:

- Modificar um algoritmo de particionamento em grafos para computar estruturas de coalizão;
- Desenvolver um método modificado de contágio para a propagação de rótulos;
- Aplicar o algoritmo modificado a grafos reais para gerar estruturas de coalizão;
- Elaborar uma métrica de comparação entre o algoritmo proposto e dois outros já existentes, um exato e outro heurístico;
- Analisar os resultados com relação ao ganho de escala do problema.

1.1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte maneira: Capítulo 2 introduz os trabalhos relacionados a este estudo; Capítulo 3 apresenta o algoritmo heurístico modificado, suas limitações e configurações; Capítulo 4 descreve as condições experimentais, quais materiais foram utilizados e quais experimentos foram realizados e seus resultados; Capítulo 5 a conclusão obtida com este estudo.

2 REVISÃO DA LITERATURA

Este capítulo apresenta o problema de particionamento de conjuntos, sendo que o problema de formação de estruturas de coalizão é uma instância dele. Apresentam-se aqui, o problema de particionamento em grafos; o mecanismo de formação de coalizão; os algoritmos mais comuns para tratar o problema e as modelagens abordadas para tentar minimizar esse problema.

2.1 PARTICIONAMENTO DE CONJUNTOS

O Problema de particionamento vem de domínios do dia a dia, como: gerenciamento de entregas, agendamento, roteamento e localização, no qual cada agente é servido por uma e apenas uma localização, veículo, pessoa, serviço ou outro recurso. (VOICE et al., 2012).

O particionamento de conjuntos (um problema NP-Difícil) pode ser resolvido com algoritmos de otimização, que são classes de algoritmos que podem soluciona-lo por resolver uma versão relaxada e computacionalmente mais fácil deste (RASMUSSEN; LARSEN, 2011), entre os algoritmos de otimização tem-se o Modelo Unificado que se utiliza dos problemas de cobertura de conjunto e empacotamento (dois problemas NP-Difícil) como forma de restrição, mas nesse método não há garantia de solução ótima. Outro exemplo de algoritmo de otimização é o Partição de Conjuntos Generalizada, como outros modelos são específicos demais para abranger casos reais de partição de conjuntos, este o faz modificando as entradas das restrições de matrizes.

Restrições computacionais, como o tempo ou memória, podem não permitir a obtenção de uma solução ótima, principalmente quando executados algoritmos de enumeração explícita (como alguns algoritmos exatos). Nesse caso, pode-se utilizar uma solução heurística para o problema. A heurística pode ser utilizada como meio de acelerar o processo de descoberta de uma solução (RASMUSSEN; LARSEN, 2011), como por exemplo, uma que poda ramos de uma busca em árvore (acelera o processo, pois é um espaço que não será investigado, mas que pode ou não conter a solução ótima). Remover equações de programas lineares (que serviriam

para decidir se um caminho deve ou não ser percorrido) na busca em árvore que pode levar a uma solução melhor ou pior também é uma heurística. De acordo com (RASMUSSEN; LARSEN, 2011) esses algoritmos são de cinco tipos, a saber:

- Heurísticos: É aquele que propõe a realização de uma pesquisa pela aproximação a um valor objetivo, ou seja, quanto mais próximo desse valor, melhor a heurística.
- Metaheurísticos: São métodos heurísticos empregados para resolver de forma genérica problemas de otimização. Esses utilizam escolhas aleatórias e conhecimento histórico de resultados anteriores, normalmente evitando ótimos locais. É, portanto, um algoritmo que junta partes de outros. Algoritmos servem-se da metaheurística como método de solução que adapta interações entre ganhos locais e estratégias de auto nível para criar processos capazes de escapar de ótimos locais e realizar uma busca robusta do espaço de solução (GLOVER; KOCHENBERGER, 2003);
- Aproximação: É um algoritmo utilizado quando uma heurística consegue garantir que o valor objetivo de qualquer solução achada por ela está dentro de uma distância do valor objetivo ótimo;
- Exatos: São algoritmos capazes de achar a solução ótima, mas não necessariamente rápidos, por exemplo a busca exaustiva;
- Baseado em métodos de otimização: São algoritmos exatos modificados que passam a ter componentes heurísticos como meio de aceleração da busca, o que pode fazê-lo perder a capacidade de achar a solução ótima;

Embora os autores tenham feito essa caracterização, os algoritmos heurísticos e metaheurísticos podem ser caracterizados como métodos de otimização, por isso, nesta dissertação, a partir de agora os algoritmos foram restritos apenas a: Heurísticos, Exatos e Aproximação.

O particionamento de conjuntos em grafos é um subproblema do particionamento de conjuntos já que leva em consideração as restrições trazidas pelos grafos. Com isso se tem uma coalizão formada pelo cominho entre os vértices. Essas restrições trazem melhorias aos algoritmos já que não necessitam trabalhar com todas as possibilidades de agrupamento (UGANDER; BACKSTROM, 2013; BISTAFFA et al., 2017; UGANDER, 2014).

Definição 5 *Caminho é uma sequência de vértices tal que para cada vértice há uma aresta para o próximo vértice em sequência, e um caminho é considerado simples se este não contém repetições de vértices. (VOICE et al., 2012)*

O problema de geração de estruturas de coalizão é semelhante ao problema de particionamento de conjuntos e um dos problemas fundamentais no ramo de otimização combinatorial que tem uso em muitos campos de conhecimento, como ciências políticas, econômicas e computacionais (YEH, 1986; VOICE et al., 2012). Este assunto é visto na próxima sessão.

2.2 FORMAÇÃO DE ESTRUTURAS DE COALIZÕES

Na forma de função característica, todo subconjunto de agentes é uma coalizão viável, fazendo com que o problema de geração de estruturas de coalizão (*Coalition Structure Generation Problem* - CSGP) seja um problema de partição completa de conjuntos (*Complete Set Partitioning Problem* - CSPP), uma subclasse do problema de particionamento de conjuntos (*Set Partitioning Problem* - SPP) (HOFFMAN; PADBERG, 2009), e portanto NP-Difícil (RAHWAN et al., 2013; MICHALAK et al., 2016).

Para se obter uma coalizão são necessários três passos básicos (SANDHOLM et al., 1999):

- Gerar as estruturas de coalizão: Particionar exaustivamente o conjunto de agentes em coalizões não sobrepostas (isto é, um agente participa de uma e apenas uma coalizão);
- Resolver o problema de otimização de cada coalizão: Maximizar o valor (monetário, utilidade, tempo, ou qualquer outra função utilizada como objetivo da formação da coalizão);
- Dividir o valor da solução gerada entre os agentes: Distribuir, entre os agentes da coalizão, de forma justa os valores recebidos (RILEY et al., 2015).

Nesta dissertação tratar-se-á apenas do primeiro e segundo item, sendo este um problema não somente de gerar a estrutura de coalizão, mas também de valorar cada coalizão utilizando uma função de valoração. E nota-se por $A = \{a_1, a_2 ... a_n\}$ o conjunto de agentes, assim $|A| = n$ será o número de agentes. Assumindo que todo subconjunto não vazio de A é uma coalizão possível, denota-se como C o conjunto de coalizões. Tem-se: $C = \{C \,|\, C \subseteq A, \forall c \in C, c \neq \emptyset\}$. O que implica que com n agentes podemos ter $2^n - 1$ coalizões.

O problema de maximização das estruturas de coalizão é conhecido como Problema de Geração de Estrutura de Coalizão Ótima (*Optimal Coalition Structure Generation* ou simplesmente OCSG), que é uma versão particular do problema de particionamento de conjuntos. Ainda de acordo com (CERQUIDES et al., 2014) o cerne do problema é que o conjunto de

coalizões, no pior dos casos, é exponencial no número de agentes, e este conjunto de partes (*powerset*) é a entrada para o problema de Geração de Estruturas de Coalizão.

Esse problema pode ser ainda maior quando levado em consideração uma externalidade, fazendo com que a complexidade dos algoritmos tornem-se $\mathcal{O}(n^n)$. As externalidades podem ser de dois tipos: (i) negativa, se forma quando uma coalizão influencia outra negativamente, por exemplo, quando se tem recursos limitados ou compartilhamento de recursos. Se uma coalizão pegar para si o recurso outra coalizão não poderá tê-lo e, com isso, pode deixar de exercer efetividade completa; e (ii) positiva, se forma quando uma coalizão exerce influência positiva em outras coalizões, por exemplo quando países se juntam para diminuir a emissão de dióxido de carbono na atmosfera outros países se beneficiam com essa decisão.

É comum representar as coalizões com grafos. Na teoria de jogos existem duas vertentes quanto a externalidades: (i) Jogo de função característica com a vantagem de que todo algoritmo o valor de uma estrutura de coalizão é sempre a mesma em toda e qualquer estrutura de coalizão (não tem externalidade) (RAHWAN et al., 2009); e (ii) Jogo de Função Particionária que contempla as externalidades.

Definição 6 *A externalidade negativa ocorre quando a união de duas coalizões faz com que uma outra piore seu valor. É dada por:* $\upsilon(c;\{c'\cup c'',c\}\cup CS') < v(c;\{c',c'',c\}\cup CS'')$ *(HAFALIR, 2007).*

Definição 7 *A externalidade positiva ocorre quando a união de duas coalizões faz com que uma outra incremente seu valor. É dada por:* $\upsilon(c;\{c'\cup c'',c\}\cup CS') > v(c;\{c',c'',c\}\cup CS')$ *(HAFALIR, 2007).*

O número de estruturas de coalizões possíveis é: $\sum_{i=1}^{n} Z(n,i)$, em que $Z(n,i)$ é o número de estruturas de coalizão com i coalizões e é calculada por: $Z(n,i) = iZ(n-1,i) + Z(n-1,i-1)$, em que $Z(a,a) = Z(n,1) = 1$ (SANDHOLM et al., 1999). O número de estruturas de coalizões é tão grande que enumerar todas por uma busca exaustiva pela solução ótima não seria viável - exceto para um pequeno número de agentes como entrada - devido às restrições computacionais que excederiam limites de tempo ou memória. O número de estruturas de coalizão está entre $\mathcal{O}(n^n)$ e $\iota(n^{n/2})$ (SANDHOLM et al., 1999).

No decorrer dos anos foram estudados muitos algoritmos e foram aplicadas várias técnicas para tornar computável a tarefa de criação e cálculo de coalizões e de estruturas de coalizões. Além dos algoritmos, e das técnicas, foram aplicadas também diversas modelagens para problemas específicos ou para redução do domínio. Para criar e calcular coalizões existem

diferentes maneiras, uma delas é o uso de grafos. Qualquer atributo quantificável do grafo, seus vértices, suas arestas, e as propriedades intrínsecas a ele, podem ser inseridas em um algoritmo para realizar o seu particionamento. Uma forma pode ser a partição simples do grafo, quebrando-o no local em que há menos ligações; outra é quebrando-o nas arestas com menor peso, o que faz o problema mudar de configuração, por exemplo: pode-se ter duas cliques em um grafo que tenderiam a formar duas coalizões, mas ligadas por uma aresta com um peso muito forte, tornando sua separação impossível de acordo com a função; outra forma é atribuir a qualquer conjunto de vértices um valor de utilidade e particionando o grafo baseado nessa função de utilidade. Outras formas de particionamento podem ser observadas em (DENG; PAPADIMITRIOU, 1994), que caracteriza essa tarefa como um jogo cooperativo e enfatizando que o Valor de Shapley (um conceito de solução na teoria de jogos cooperativos) é sempre fácil de computar, sendo ele metade da soma das arestas incidentes ao vértice, e sua complexidade $\mathcal{O}(n^2)$.

Definição 8 *Valor de Shapley (Shapley value): reflete a contribuição de cada vértice ao grafo de acordo com sua ordem de entrada:* $\oslash(i) = \sum_{i \in c \subseteq A} \frac{(n-|c|)!(|c|-1)!}{n!}(val(c) - val(c-i))$ *(SHAPLEY, 1971).*

Uma modelagem comum é mapear os agentes como vértices; as arestas como o relacionamento de dois agentes trabalhando juntos; o peso é o quão bem eles trabalham juntos e o valor da coalizão é a soma dos pesos das arestas presentes entre os membros da coalizão. Nessa modelagem a coalizão ótima é considerada NP-Difícil mas com restrições na entrada tornando-o tratável (BACHRACH et al., 2013) até certo ponto, já que as restrições computacionais não são solucionadas com esta abordagem.

O estudo da complexidade computacional associada aos conceitos de solução uma vez que a entrada é, por si só, exponencial feita em (DENG; PAPADIMITRIOU, 1994), em que o autor caracterizou a formação de coalizões como um grafo não direcionado com peso, no qual todas as arestas tinham pesos positivos, esse conjunto se caracteriza como uma função subaditiva e, portanto, convexa. E como os pesos são positivos, o problema pode ser formulado, geralmente, como um problema de fluxo máximo que contém vários algoritmos que o resolvem (um deles é a Programação Linear).

Grafos e redes tem uma grande correlação, nesta dissertação redes e grafos são tratados como iguais. Para ilustrar o vocabulário dos pesquisadores de redes complexas, nesta dissertação foram dadas as mesmas definições e denominações que aparecem na literatura.

O acesso a inúmeras redes reais estimulou um grande interesse em tentar classifica-

las e enumerar suas propriedades genéricas. Medir algumas propriedades básicas de uma rede complexa, como: o caminho médio, o coeficiente de agrupamento (que diz a proporção de ligações em um determinado grupo de vértices), e o grau de distribuição, é o primeiro passo para a compreensão de sua estrutura. O segundo passo é o desenvolvimento de um modelo matemático com topologia de propriedades estatísticas semelhantes, obtendo-se, assim, uma plataforma sobre a qual a análise matemática é possível (WANG; CHEN, 2003). Essas três propriedades básicas são levadas em consideração ao realizar a classificação de uma rede.

Definição 9 *O coeficiente de agrupamento de um vértice v com seus vizinhos $N(v)$ e existindo no máximo $|N(v)| * (|N(v)| - 1)/2$ arestas entre eles (isto ocorre quando cada vizinho de v está conectado a qualquer outro vizinho de v). Deixe C_v denotar a fração das arestas que realmente existem. C_v reflete até que ponto os amigos de v também são amigos um do outro; e, portanto o coeficiente de agrupamento mede a capacidade de clique de um círculo de amizade típico (WATTS; STROGATZ, 1998).*

2.2.1 COALIZÕES LEVANDO EM CONSIDERAÇÃO AS ESTRUTURA DE GRAFOS

A motivação para estudar estruturas específicas de grafos para determinados problemas é enfatizada por (GOLUMBIC, 2014) ao dizer que a elaboração de novas estruturas teóricas tem motivado a busca por novos algoritmos compatíveis com essas estruturas, ao invés do estudo árduo e sistemático de cada novo conceito definível com um grafo, mantendo apenas o ”profundo”problema matemático.

Coloração de grafos, *Domatic partitioning*, *Weighted k-cut* e muitos outros problemas podem ser vistos como sendo um caso especial de particionamento de A (o conjunto de agentes) em subconjuntos de uma dada família, a partir de uma função que quer somar o produto ou otimizar sua soma (BJöRKLUND et al., 2009).

Considerando os estudos já realizados (BJöRKLUND et al., 2009) e (HOEFER et al., 2015), pode-se dizer que o problema de formação de coalizão também pode aparecer em estruturas de grafos. A seguir serão apresentadas algumas dessas estruturas, um exemplo de como o problema se comportaria nessa classe, e como ele poderia ser resolvido utilizando as propriedades da classe. O enfoque dado é que cada coalizão é formada por um clique. O problema passa a ser achar todas as cliques do grafo para realizar a combinatória da melhor estrutura.

Definição 10 *Clique: é um subgrafo completo (GOLUMBIC, 2014).*

- Classe dos Grafos Completos

  A quantidade de cliques (não maximais) em um grafo completo é o próprio *powerset* do conjunto, uma vez que todas as combinações são possíveis.

  Definição 11 *Grafo é completo: ocorre se todo par distinto de vértices é adjacente (GOLUMBIC, 2014). Sua representação é dada por K_n, e n é o número de vértices do grafo.*

- *Classe dos Grafos Bipartidos*

  A quantidade de cliques em um grafo bipartido é o número de arestas no grafo, pois não existem cliques maiores do que dois nem em grafos bipartidos nem no complemento de grafos bipartidos.

  Definição 12 *Grafo Bipartido: é aquele quem tem vértices que podem ser particionados em dois conjuntos disjuntos $V = V' + V''$ e, cada aresta tem um ponto em V' e outro em V'', é equivalente dizer que pode ser colorido com duas cores (GOLUMBIC, 2014).*

- Classe dos Grafos Planares

  Pode-se dizer que são Grafos Planares, os isomórficos à aqueles que podem ser desenhados em um plano sem que as arestas se cruzem (TRUDEAU, 2013). Um grafo planar é esparso, isso é, a quantidade de arestas é pequena (normalmente quando $|E| \ll n^2$). Euler provou que a quantidade máxima de arestas em um grafo desse tipo é $\|E\| \leq 3n-6$. Essa redução na quantidade de ligações faz a quantidade de combinações diminuir drasticamente. Segundo (DEAN; BODDY, 1988), o número máximo de cliques em um grafo planar é 8(n - 2).

  Definição 13 *Grafo Planar: é um grafo que não tem como subgrafo um K_5 nem um $K_{3,3}$, isso faz com que nenhuma clique em um grafo planar tenha mais do que quatro vértice (GOLUMBIC, 2014).*

- Classe dos Grafos Árvore

  A quantidade de cliques em um Grafo Árvore é o número de arestas no grafo para qualquer grafo com mais de um vértice. A intuição da prova basta seguir a definição e lembrar que uma árvore é um grafo conexo, então ou ela tem um único vértice (e, portanto, 1-clique) ou ela tem vários vértices e todos contidos no conjunto E de arestas. Como cada aresta é uma clique em um Grafo Árvore, todas as arestas são cliques.

A maior clique existente em uma árvore é 2-clique e a intuição se faz demostrando que para ter-se uma clique de tamanho maior, necessariamente existiria um ciclo de tamanho três e, por definição, uma árvore não tem ciclos.

Definição 14 *Árvore: é um grafo conexo sem ciclos (MESBAHI; EGERSTEDT, 2010). (VOICE et al., 2012) define como um grafo no qual qualquer par de vértices estão conectado por exatamente um caminho.*

Em um grafo aleatório (Erdos–Rényi, mais especificamente o G(n,1/2)) apesar da clique máximo ser próximo de $2log_2 n$, algoritmos gulosos não acham bons resultados, e o número de cliques maximais (cliques que não podem crescer já que a adição de qualquer vértice faria o conjunto deixar de ser uma clique) tem um limite superior de $3^{n/3}$ (MOON; MOSER, 1965).

Definição 15 *Rede aleatória (Random Networks ou Erdos–Rényi, mais especificamente o G(n,1/2)): refere-se a probabilidade de que dois vértices ligados a um terceiro também estejam conectados não é maior do que a probabilidade que esses dois vértices escolhidos ao acaso estejam conectados (WANG; CHEN, 2003).*

Algumas outras formas de modelagem para computar coalizões são apresentadas a seguir:

- *Weakest Link Game* (Jogo de ligação mais fraco): Um jogo cooperativo no qual o valor do conjunto é determinado pelo membro mais fraco, tendo uma aproximação $\mathcal{O}(\log n)$ (BACHRACH et al., 2014), a modelagem dada pelo autor utilizou-se de um grafo que teve peso atribuído as arestas, com vértices destino e origem pré-determinados, em que os agentes são os vértices, o caminho entre os vértices é calculado pela aresta mais fraca e o valor de um agente em uma coalizão é o melhor caminho de todos os caminhos contido na coalizão: $v(c) = \sum_{i \in c} p_i$, em que p_i é o *payoff* (o valor de ganho do agente) do agente i.

- *Bounded Treewidth Graph* (Gráfico de Limite de Largura de Árvore): Representa uma coalizão como sendo o caminho em um grafo que tem arestas com pesos. Este, por sua vez, também é representado em formato de árvore e sua busca é realizada com um algoritmo comum de busca em largura (Breadth First Search) em tempo polinomial (BACHRACH et al., 2013).

- *Planar Graph* (Grafo Planar): A modelagem proposta coloca os agentes como vértices e o trabalho conjunto entre dois agentes como arestas nas quais uma coalizão é alcançada evitando-se arestas negativas e ganhando uma porção constante do peso das arestas positivas em que o valor ótimo é a soma dos pesos positivos, com tempo polinomial (BACHRACH et al., 2013).

A utilização de Teoria de Jogos como base matemática e de modelagem demostraram que coalizões também são encontradas. Segundo a modelagem de (PELETEIRO et al., 2014), que facilita a cooperação entre os agentes, utiliza um jogo de reciprocidade indireta [1], no qual os agentes podem criar coalizões para compartilhar informações sobre a reputação dos agentes e mudar a sua rede de vizinhos, obtiveram as seguintes conclusões:

- Redes Livre de Escala, permitem que os agentes sejam naturalmente cooperativos apenas juntando-se a uma coalizão. A conclusão encontrada foi que em redes Livre de Escolha os *Hubs* (agentes com muitas ligações) têm grande influência sobre outros agentes e detém muita informação sobre eles. A existência desses Hubs faz com que a convergência para uma estratégia dê-se rapidamente (PELETEIRO et al., 2014).
- Redes de Mundo Pequeno, os agentes são naturalmente não cooperativos. A cooperação em redes de Mundo Pequeno se deu quando foi adicionada a possibilidade de mudar de vizinhança (remover e adicionar arestas). Nesse tipo de rede, os agentes têm uma semelhança em seus graus (quantidade de vizinhos a distância um), o que significa que eles têm um nível semelhante de informação no início e quando adiciona-se a possibilidade de religação os agentes começam a criar grupos de influência compostas por alguns agentes que têm conexões mais elevadas do que os outros (PELETEIRO et al., 2014).

O fato de diminuir a quantidade de vértices certamente melhora o desempenho de alguns algoritmos, mas somente esse fato pode, além de não trazer benefícios, prejudicar a procura por estruturas de coalizão como pode ser visto em (NUNES, 2015), que elaborou seu estudo nas restrições do espaço de busca para a formação de estruturas de coalizão. No estudo realizado por Nunes, mesmo grafos esparsos podem levar a uma complexidade exponencial como o caso do grafo estrela.

A Figura 1 que compara o número de coalizões possíveis $(2^n - 1)$ de um grafo comum, as coalizões possíveis em um grafo estrela $S_{1,n-1}$, as coalizões considerando cliques e considerando subgrafo conexo. Em (a) é mostrada a linha de progressão da quantidade de coalizões

[1] O jogador coopera apenas quando o destinatário tem a mesma ou maior reputação do que a estratégia do doador.

possíveis em um grafo comum, essa linha tem o mesmo comportamento no grafo estrela em (b). Ao visualizar em (b) a quantidade de subgrafos conexos, é observada a mesma linha já que a quantidade de subgrafos conexos em um grafo estrela é $2^{n-1} + (n-1)$ (NUNES, 2015). Ainda em (b) é observado uma grande queda no número de coalizões quando, para a formação desta, é considerado apenas os subgrafos que formam uma clique, tendo seu número restrito a $n-1$.

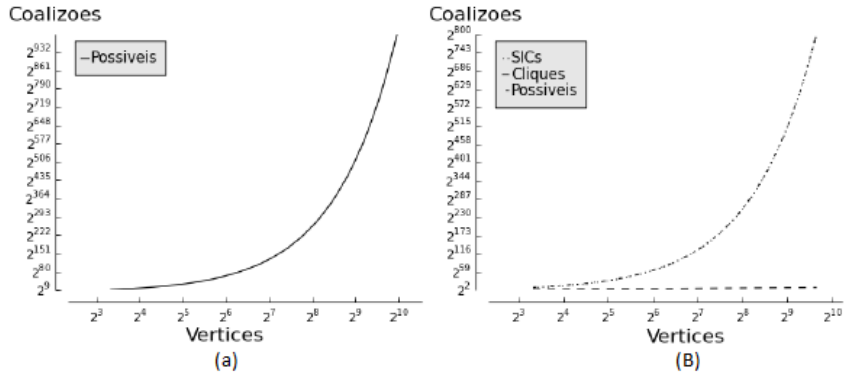

Figura 1: Comparação entre a quantidade de coalizões de um grafo comum e estrela considerando todas as possibilidades e as restritas em cliques e subgrafo conexo.

Fonte: (NUNES, 2015)

2.2.2 ALGORITMOS DE FORMAÇÃO DE COALIZÕES

O algoritmo Busca de Estruturas de Coalizões (*Coalition-Structure-Search-1* ou simplesmente CSS1) pode ser considerado o primeiro algoritmo *anytime* criado para a geração de estruturas de coalizão (SANDHOLM et al., 1999). Além de dar uma resposta a qualquer instante (por isso *anytime*), ele ainda garante um limiar com a solução ótima isto é, consegue dizer o quão distante a solução dada está da ótima, tratando o problema como uma busca em um grafo. As estruturas de coalizão podem ser visualizadas na Figura 2, em que os vértices são estruturas de coalizão no formato de partição de inteiros e as arestas são a fusão (ou fissão) de duas coalizões. Ao pesquisar os dois níveis inferiores o algoritmo consegue entrar no limiar da solução ótima, tendo assim uma aproximação realista da solução ótima, quanto mais níveis percorre no grafo, melhor será essa aproximação. O algoritmo CSS1 é apresentado no algoritmo 1.

noend 1 Algoritmo CSS1 para geração de estruturas de coalizão. Fonte: (NUNES, 2015)

1: Pesquisar os dois níveis inferiores do grafo de estruturas de coalizão
2: Efetuar busca em largura a partir do nível superior do grafo de estruturas de coalizão enquanto houver tempo ou até que o grafo inteiro tenha sido percorrido
3: Retornar a estrutura de coalizão com maior valor encontrada

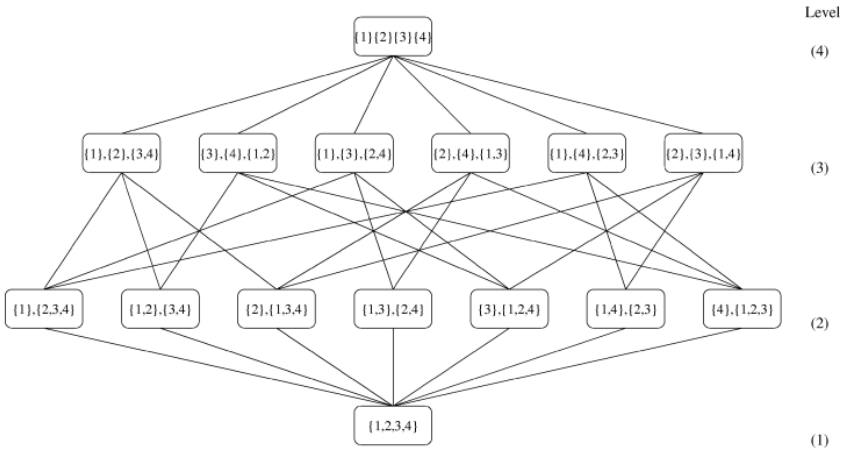

Figura 2: Grafo do processo de geração de estruturas de coalizão.
Fonte: (SANDHOLM et al., 1999)

A Figura 2 representa um grafo gerado a partir do processo de geração de estruturas de coalizão para quatro agentes. Os vértices representam estruturas de coalizão. As arestas representam a fusão (ou fissão) de duas coalizões. Esse algoritmo percorre os dois últimos níveis do grafo para impor um limite inferior e a cada novo nível esse limite se aproxima mais da solução ótima. Após delimitar, o algoritmo pode, ou não, verificar se outras partes do grafo devem ser inspecionadas, fazendo com que seja uma busca em grafo com restrições.

Programação Dinâmica (*Dynamic Programming* ou simplesmente DP) é um algoritmo projetado para resolver o problema de particionamento de conjuntos (YEH, 1986), também pode ser utilizado para calcular estruturas de coalizão pois, ambos os problemas tem como objetivo particionar um conjunto de elementos em subconjuntos, e no caso das estruturas de coalizão de forma a maximizar seus valores (SANDHOLM et al., 1999). Neste algoritmo $f(c)$ é uma coluna de uma tabela que guarda o valor da coalizão c e $t(c)$ guarda a coalizão c. O algoritmo DP é apresentado no Algoritmo 2.

noend 2 Algoritmo DP para geração de estruturas de coalizão. Fonte: (NUNES, 2015)

1: **for all** $a \in \{1,...n\}$ **do**
2: $f_1[a_i] \leftarrow a_i$
3: $f_2[a_i] \leftarrow v(a_i)$
4: **for all** $s \in \{2,...n\}$ **do**
5: **for all** c $\subseteq$ A, com $|c|$ = s **do**
6: $f_2[c] \leftarrow \max\{f_2[c'] + f_2[c''] : c' \subset c, c'' = c - c', 1 \leqslant |c'| \leqslant \frac{|c|}{2}\}$
7: **if** $f_2[c] \leqslant v$(c) **then**
8: $f_1[c] \leftarrow$ c* (onde c* é o maior c' do passo anterior)
9: **else**
10: $f_1[c] \leftarrow c$
11: CS* $\leftarrow$ {a}
12: **for all** c $\subseteq$ CS* **do**
13: **if** $f_2[a_i] \neq$ c **then**
14: CS* $\leftarrow$ (CS* - {c} $\cup$ ($f_1[c]$)

A intuição por trás desse algoritmo é selecionar as coalizões com número maior que 2, que podem ser divididas em coalizões de número menor, e dividi-las repetidamente até terem o tamanho 1. Então, ao realizar o cálculo de uma coalizão maior que 2, é necessário verificar numa tabela, se a coalizão é melhor junta ou separada, se for melhor junta troca-se o valor de $v*$ (o melhor valor) e $c*$ (a melhor coalizão) pelos novos valores.

Programação Dinâmica Ótima (*Optimal Dinamic Programing* ou simplesmente ODP) foi proposto por (MICHALAK et al., 2016), é uma modificação do algoritmo DP para evitar avaliar alguns movimentos (evitar caminhos que levam ao mesmo lugar). Em seu trabalho foi provado que, enquanto existir um caminho que leve a solução ótima, o algoritmo DP pode chegar a ela. Essa limitação do espaço de busca é realizada pela remoção de arestas do nível inferior para o superior sem afetar a otimalidade do algoritmo. Nesse algoritmo, M é o conjunto de todos os movimentos possíveis no grafo e M^* o subconjunto de M que tem somente os movimentos suficientes e necessários para chegar a solução ótima. Os autores provaram que não há como garantir uma solução ótima enquanto não forem realizadas as $|M^*|$ operações. O algoritmo ODP evita, com esse corte, 2/3 das operações e sua complexidade é calculada em $\mathcal{O}(3^n)$.

noend 3 getBestPartition (subfunção do ODP). Fonte: (RAHWAN et al., 2013)

1: **if** $t(c) = \{c\}$ **then**
2: **return** c
3: **else**
4: $\pi \leftarrow \emptyset$
5: **for all** $c_i \in t(c)$ **do**
6: **if** $f_{M^*}(c_i) = v(c_i)$ **then**
7: $t(c_i) \leftarrow \{c\}$
8: **else**
9: **for all** $\{c_i', c_i''\} \in \Pi^{c_i} \cup (\{\emptyset, c_i\}) \in t(c)$ **do**
10: **if** $f_{M^*}(c_i') + f_{M^*}(c_i'') = f_{M^*}(c_i)$ **then**
11: $t(c_i) \leftarrow \{c_i', c_i''\}$
12: **break**
13: $\pi \leftarrow \pi \cup getBestPartition(c_i, t(c_i))$
14: **return** π

noend 4 Algoritmo ODP. Fonte: (RAHWAN et al., 2013)

for all $c \subseteq A : |c| = 1$ **do**
  $f_{M^*}(c) \leftarrow v(c)$
for all $c \subseteq A : |c| = 2$ **do**
  $f_{M^*}(c) \leftarrow v(c)$
if $f_{M^*}(\{a_1, a_2\} < v(\{a_1\}) + v(\{a_2\})$ **then**
  $f_{M^*}(\{a_1, a_2\}) \leftarrow v(\{a_1\}) + v(\{a_2\})$
for all s = 3 **to** n - 1 **do**
  for all $S \subseteq A \backslash \{a_1, a_2\} : |S| = s - 2$ **do**
    $c \leftarrow S \cup \{a_1, a_2\}$
    $f_{M^*}(c) \leftarrow v(c)$
    for all $\{S', S''\} \in (\Pi^S \cup \{\emptyset, S\})$ **do**
      $j \leftarrow \min_{a_j \in A \backslash c} i$
      $A^j \leftarrow \{a_3, ..., a_{j-1}\}$
      if $f_{M^*} < v(S' \cup \{a_1\}) + v(S'' \cup \{a_2\})$ **then**
        $fM^* \leftarrow v(S' \cup \{a_1\}) + v(S'' \cup \{a_2\})$
      if $f_{M^*} < v(S' \cup \{a_2\}) + v(S'' \cup \{a_1\})$ **then**
        $fM^* \leftarrow v(S' \cup \{a_2\}) + v(S'' \cup \{a_1\})$
      if $S' \cap A^j \neq \emptyset$ **then**
        if $f_{M^*} < v(S') + v(S'' \cup \{a_1, a_2\})$ **then**
          $f_{M^*} \leftarrow v(S') + v(S'' \cup \{a_1, a_2\})$
      if $S'' \cap A^j \neq \emptyset$ **then**
        if $f_{M^*} < v(S' \cup \{a_1, a_2\}) + v(S'')$ **then**
          $f_{M^*} \leftarrow v(S' \cup \{a_1, a_2\}) + v(S'')$
  for all $c \subseteq A : |c| = s, \{a_1, a_2\} \nsubseteq c$ **do**
    $f_{M^*}(c) \leftarrow v(c)$
$f_{M^*}(A) \leftarrow v(A)$
$t(A) \leftarrow \{A\}$
for all $\{c', c''\} \in \Pi_2^A$ **do**
  if $f_{M^*}(A) < f_{M^*}(c') + f_{M^*(c'')}$ **then**
    $f_{M^*}(A) \leftarrow f_{M^*}(c') + f_{M^*}(c'')$
    $t(A) \leftarrow \{c', c''\}$
$CS^* \leftarrow getBestPartition(A, t(A))$
return CS*

Partição de Inteiros (*Integer Partition* ou simplesmente IP) proposto por (RAHWAN et al., 2009) é um algoritmo *anytime* que utiliza técnicas de Divisão e Conquista para geração de estruturas de coalizão, que tem sua complexidade em $\mathcal{O}(n^n)$. O algoritmo cria uma função que

retorna todas as estruturas de uma determinada configuração (estruturas iguais às encontradas na Figura 2) que são representadas em forma de lista, depois aplicam-se conceitos de divisão e conquista definindo limites para as configurações e retirando as estruturas que possuem pouca possibilidade de serem a solução ótima. Os limites são refinados sempre que a busca é aprofundada (por isso *anytime*).

Programação Dinâmica Melhorada (*Improved Dynamic Programming* ou simplesmente IDP) é uma modificação do algoritmo DP, com o intuito de limitar os espaços de busca sem comprometer a solução ótima (RAHWAN; JENNINGS, 2008). Esta limitação foi feita a partir da propriedade de que cada vértice necessita apenas de uma aresta direcionada a ele para garantir que a solução ótima seja encontrada. Com essa diminuição do espaço de busca menos operações são realizadas exigindo menos memória. Removendo-se as redundâncias tem-se o grafo representado pela Figura 2, ao analisar os níveis em ordem crescente, há uma estrutura de coalizão com todos os agentes juntos que se divide até gerar uma estrutura com todos os agentes separados, a cada passo ela se divide uma vez. A cada passo o particionamento da coalizão $c \in CS$ em c' e c" sendo que o passo seguinte a estrutura será $(CS \backslash c) \cup \{c', c''\}$. O algoritmo IDP é mostrado no algoritmo 5.

noend 5 Algoritmo IDP para geração de estruturas de coalizão. Fonte: (NUNES, 2015)

1: **for all** $a \in \{1, ...n\}$ **do**
2: $f_1[a_i] \leftarrow a_i$
3: $f_2[a_i] \leftarrow v(a_i)$
4: **for all** $s \in \{2, ...n\}$ **do**
5: **for all** c $\subseteq$ A, com $|c|$ = s **do**
6: $f_2[c] \leftarrow \max\{f_2[c'] + f_2[c''] : c' \subset c, c'' = c - c', 1 \leqslant |c'| \leqslant \frac{|c|}{2}, (|c''| \leqslant (n-s)\, ou\, |c'| + |c''| = n\}$
7: **if** $f_2[c] \leqslant v$(c) **then**
8: $f_1[c] \leftarrow$ c* (onde c* é o maior c" do passo anterior)
9: **else**
10: $f_1[c] \leftarrow c$
11: CS* $\leftarrow$ {a}
12: **for all** c $\subseteq$ CS* **do**
13: **if** $f_2[a_i] \neq$ c **then**
14: CS* $\leftarrow$ (CS* - {c} $\cup$ ($f_1[c]$)

Partição de Inteiros$^{+/-}$ (*Integer Partition*$^{+/-}$ ou simplesmente IP$^{+/-}$) é uma modificação do algoritmo IP que leva em conta as externalidades (RAHWAN et al., 2012).

Alocação de tarefas via formação de coalizões (*Task allocation via coalition formation*

- TAVCF) é um algoritmo heurístico, *anytime*, guloso e polinomial que produz resultados que estão perto do ótimo, comprovado por uma proporção logarítmica proposto por (SHEHORY; KRAUS, 1998) e assume que não há autoridade central que distribui as tarefas entre os agentes, portanto os agentes trocam mensagens entre si e não tem visão geral do problema. O algoritmo consiste em duas etapas: (i) é calculado o estado inicial de todas as coalizões possíveis; (ii) iterativamente recalcula-se o valor da coalizão e os agentes decidem qual coalizão preferem formar. Redução do espaço é realizada quando se fala de ”coalizões possíveis”, pois há preferência por coalizões com baixo número de agentes justificada pelos custos de comunicação entre os agentes e os altos custos computacionais de atividades. Portanto, os agentes visam a diminuição de custos com comunicação e computação desnecessárias. A escolha de uma coalizão por um agente é dada em cada iteração do algoritmo, que decide qual coalizão é sua preferida, e a configuração da coalizão é gradualmente alcançada. Ao final do processo, o valor da coalizão é calculado, cada agente terá uma lista de coalizões, seus valores e custos que foram calculados. Sobreposição de coalizão é permitida e no caso de ocorrerem, os agentes podem contribuir para os resultados totais várias vezes. Esta diferença provoca uma escolha diferente dos agentes para cada coalizão a ser formada.

Algoritmo Genético Baseado em Ordem (*Order-Based Genetic Algorithm* - OBGA) não dá garantias de otimalidade e tem sua iteração sempre em três passos: (i) avaliação: Avalia a população corrente, (ii) seleção: Seleciona os membros repetidamente com reposição e baseia-se em sua avaliação para gerar uma nova população e (iii) recombinação: Constrói novos membros a partir dos membros já selecionados por meio de modificações de seus atributos (mutação) ou trocando os membros (crossover) (SEN et al., 2000).

Arrefecimento Simulado para Formação de Coalizões em Multi-agentes (*Simulated Annealing for Multi-agent Coalition Formation* - SAMCF) é um algoritmo de busca local, tenta fugir dos máximos locais fazendo saltos aleatórios para uma outra solução, neste cenário a probabilidade de realizar um movimento ruim diminui com o passar do tempo (conforme diminui a temperatura). Apesar de as chances de encontrar a solução ótima seja considerável, quando a temperatura baixa lentamente, não há garantia de encontrá-la. Basicamente o algoritmo inicia em uma posição aleatória e salta para um vizinho (cada vizinho é uma estrutura de coalizão), se encontrar uma coalizão melhor armazena esta como melhor, se não ele ainda assim pode armazená-la com probabilidade relativa a temperatura (evitando assim, máximos locais) (KEINäNEN, 2009).

Grafo de Sinergia Ponderado (*Weighted Synergy Graph* - WSG) introduz a noção de sinergia em grafo que utilizam peso nas arestas, que permite um conjunto de agentes avaliem

a formação da equipe (LIEMHETCHARAT; VELOSO, 2014); dado um conjunto de agentes e uma tarefa para a qual são necessários diferentes papeis, o grafo de sinergia é usado para aproximar a atribuição de função ideal para a equipe. O algoritmo gera um grafo de sinergia ponderado aleatório com uma estimativa das capacidades dos agentes usando a estrutura do grafo que utilizam arestas com peso e um conjunto de observações então, o algoritmo de aprendizagem melhora iterativamente o grafo utilizando a estimativa atual do gráfico ponderado de sinergia.

O Grafo de Sinergia Ponderado (WSG) tem abordagem heurística para formação de coalizão (e não estruturas de coalizão). Primeiramente, ilustrado pela Figura 3-a, gera-se uma coalizão, um grafo não direcionado com peso, que forma-se com a iteração dos agentes ao realizar uma única tarefa (com outra tarefa, as iterações seriam diferentes e os pesos também, gerando assim um novo grafo) (LIEMHETCHARAT; VELOSO, 2014). Os pesos representam a dificuldade ou custo de dois agentes trabalharem juntos. Após gerar a coalizão, é calculada a compatibilidade (ϕ) entre os agentes em uma coalizão que é dada pelo menor caminho entre dois agentes. As arestas não utilizadas são removidas, já que desnecessárias, simplificando o grafo. A heurística é calculada como: $\phi_{fraction}(d) = 1/d$ ou $\phi_{decay} exp(-dln2/h)$. O grafo passa ao estado ilustrado pela Figura 3-b (em (a) o grafo gerado a partir da iteração entre os agentes ao realizarem uma atividade, em (b) o grafo contendo apenas o menor caminho entre os vértices e em (c) a proficiência do agente ao realizar determinada tarefa).

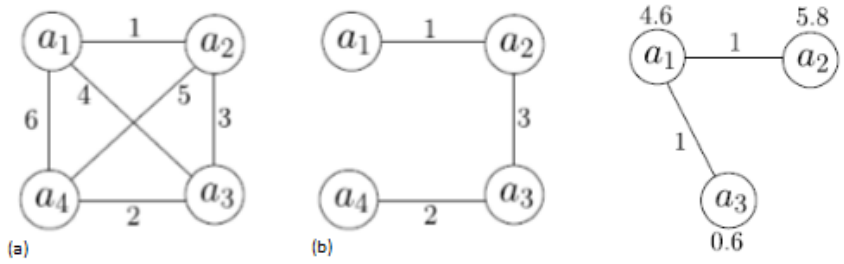

Figura 3: Grafo com peso representando a iteração entre os agentes.
Fonte: (LIEMHETCHARAT; VELOSO, 2014)

Em seguida é calculada a proficiência (μ) de um agente ao desempenhar uma determinada tarefa, e é vista como a mensuração da contribuição do agente para a coalizão ao desempenhar a tarefa. Esse valor que mensuração a contribuição do agente é dado pelo problema. Na Figura 3-c, o agente a_1 trabalha igualmente bem com a_2 e a_3 mas o desempenho da coalizão formada por $\{a_1, a_2\}$ é melhor que $\{a_1, a_3\}$ dado que $\mu_2 > \mu_3$. A a proficiência do agente ao realizar várias vezes a mesma tarefa não será a mesma, já que o agente age em um ambiente não determinístico e seu desempenho pode variar nas inúmeras vezes que exerce essa mesma

tarefa. Para resolver esse problema de variabilidade, é incorporado ao modelo uma variável normalmente distribuída σ, não utilizando um valor simples. Então, cada agente é associado a uma variável aleatória C_i que é a sua capacidade não determinística, dado por C_i $\mathcal{N}(\mu_i, \sigma_i^2)$, onde μ é o valor mais frequente e σ é a variância desse valor.

O maior ponto torna-se: Como saber se $\mathcal{N}(\mu_c, \sigma_c^2) > \mathcal{N}(\mu_{c'}, \sigma_{c'}^2)$? Sendo que σ_c^2 e $\sigma_{c'}^2$ são variáveis de uma distribuição normal e μ_c e $\mu_{c'}$ são os picos de maior ocorrência da proficiência do agente. A resposta é que depende do δ utilizado para "margem de erro"esperada. O δ é a probabilidade de uma coalizão atingir uma utilidade $\mathscr{U}$ enquanto nenhuma outra pode. Exemplificando, se $\delta < 1/2$ então uma coalizão de grande risco e grande recompensa é preferida a uma que tenha pouca probabilidade de obter um grande desempenho: tem-se assim, $\mathcal{N}(30.1, 10.3) > \mathcal{N}(17.8, 0.8)$.

A tabela 1 mostra a comparação entre os algoritmos e suas eficiências e abordagens.

Tabela 1: Comparação entre os algoritmos exatos citados no texto

Algoritmos	Optimalidade	Complexidade	Externalidade	anytime	autor	intuição
CSS1	S	$\mathcal{O}(3^n)$	N	S	(SANDHOLM et al., 1999)	Percorrer os dois últimos níveis para dar um limite inferior.
DP	S	$\mathcal{O}(3^n)$	N	N	(SANDHOLM et al., 1999)	Programação Dinâmica.
ODP	S	$\mathcal{O}(3^n)$	N	N	(MICHALAK et al., 2016)	Evita $2/3$ das operações de comparação do DP.
IDP	S	$\mathcal{O}(3^n)$	N	S	(RAHWAN; JENNINGS, 2008)	Melhoria sobre o ODP.
IP	S	$\mathcal{O}(n^n)$	N	S	(RAHWAN; JENNINGS, 2007)	Utilização de técnica *branch- and-bound* otimizando a busca e eliminando espaços infrutíferos.
IP$^{+/-}$	S	$\mathcal{O}(n^n)$	S	S	(RAHWAN et al., 2012)	Modificação do algoritmo IP para considerar externalidades positivas e negativas.
ODP-IP	S	$\mathcal{O}(3^n)$	N	S	(RAHWAN; JENNINGS, 2008; MICHALAK et al., 2016)	Combinação dos algoritmos ODP e IP combinando assim programação dinâmica com característica anytime.
TAVCF	N	$\mathcal{O}(n^k)$	-	S	(SHEHORY; KRAUS, 1998)	Algoritmo heurístico com restrição no tamanho da coalizão, sendo k a quantidade de agentes na maior coalizão.
OBGA	N	-	N	S	(SEN et al., 2000)	Algoritmo genético.
SAMCF	N	-	-	S	(KEINäNEN, 2009)	-
WSG	N	-	-	S	(LIEMHETCHARAT; VELOSO, 2014)	Utiliza mecanismo de aprendizagem para iterativamente melhorar um grafo de sinergia que é usado para aproximar a atribuição de função ideal para a equipe.

Adaptado de (NUNES, 2015)

Detecção de comunidades e partição de grafos são similares, e se os algoritmos de propagação de rótulos são bons para detecção de comunidades, então eles também podem ser utilizados para particionamento de grafos com duas diferenças: (i) a detecção de comunidades não necessita de uma especificação do tamanho da comunidade; (ii) podem detectar comunidades sobrepostas (os de partição procuram por partições não sobrepostas (UGANDER, 2014)).

A propagação de rótulos foi, também, estudada em (RAGHAVAN et al., 2007), que mostra a grande gama de definições ou tentativas de definir comunidade como clique, k-clique, grau, k-clubs e outros, e como detecta-las com algoritmos de *flooding*, propagação de rótulos, corte mínimo, centralidade baseada em arestas e mais. A proposta de (RAGHAVAN et al., 2007) é sobrepor várias execuções de um algoritmos de propagação de rótulos e uni-los em uma única resposta fazendo com que as comunidades que mais se destacam sejam consolidadas na solução final, ele demostra que já na quinta iteração de um algoritmo de propagação foram rotulados corretamente cerca de 95% dos vértices tocados (sendo um bom início para prosseguir com a detecção através dos algoritmos de propagação de rótulos).

2.2.2.1 JUSTIFICATIVA DA ABORDAGEM

No modelo tradicional, no qual qualquer subconjunto de agentes é considerado uma coalizão possível, o número total de coalizões é exponencial ao número de agentes: $2^n - 1$, e que a geração destas podem ser diminuídas quando mapeadas como um grafo e utilizando as conexões deste grafo como restrições. Mas diminuir a entrada através de um grafo de restrições não é suficiente, pois o número de coalizões factíveis (subgrafos conexos) podem ser exponenciais mesmo em grafos esparsos como é mostrado em (NUNES, 2015) que, em seu trabalho, buscou apresentar uma análise sobre a utilização da teoria dos grafos para buscar a redução do número de coalizões factíveis mapeando-as como cliques e não somo subrafos conexos.

Todo grafo, inclusive os esparsos, contém uma árvore geradora e portanto podem ter um número exponencial de subgrafos induzidos conexos, pois o simples fato do grafo ser esparso não torna o problema menos complexo, mas utilizar uma restrição sobre as coalizões que considera somente factível quando é uma clique, potencialmente colabora (NUNES, 2015)

2.3 PROPAGAÇÃO DE RÓTULOS BALANCEADA

A modificação de um algoritmo de propagação de rótulos para realizar particionamentos em grafos gigantes de forma igualitária (em partes iguais) enquanto maximiza arestas locais (arestas que iniciam e terminam no mesmo particionamento) foi proposta em (UGAN-

DER, 2014). O algoritmo foi dividido em duas partes: Primeiro dividiu-se geograficamente os vértices do grafo social em cortes, e depois de realizar essa agregação inicial, utilizou o algoritmo modificado de propagação de rótulos somente nos cortes e não no grafo inteiro, tornando possível executar esta tarefa. O algoritmo foi formulado como um problema de otimização e é iterativo, sendo que cada iteração resolve um programa linear. Os algoritmos de propagação de rótulos não dão garantias de desempenho, e a modificação feita também não. Em um algoritmo de propagação de rótulos os vértices não rotulados iterativamente adotam o rótulo da maioria dos seus vizinhos até a convergência.

Em seu trabalho, (UGANDER, 2014), valeu-se de uma modificação do algoritmo de propagação de rótulos para otimizar um serviço utilizado pelo Facebook (um grafo com milhões de vértices e arestas) chamado *People You May Know Service* (Pessoas que Você Pode Conhecer), esse serviço é responsável por mostrar dicas de pessoas com as quais se conectar. Esse serviço mostra para um determinado usuário (vértice) u e para cada amigo de um amigo w (vértices a distância dois de u) um vetor x_{uw} baseado na estrutura local entre u e w. Quando esse serviço é requisitado, para cada $w \in N(u)$, é enviada uma requisição para cada máquina m_w (que contém as informações do vértice w - cada máquina é considerada uma partição). O objetivo da propagação balanceada de rótulos é fazer a partição de vértices em máquinas de tal forma que, sempre que possível, $m_u = m_w \forall w \in N(u)$.

A estratégia utilizada foi especificar o algoritmo guloso como um problema de maximização submetido a restrição de um limite superior (T) e inferior (S), tal que $S_i \leq |V_i| \leq T_i, \forall i$. Basicamente o algoritmo faz os seguintes passos iterativamente:

- Determinar para onde cada vértice prefere mover-se, e quanto cada vértice ganharia com esta transferência;
- Classificar os ganhos de cada vértice para cada par no corte e construir uma equação linear;
- Resolver a equação linear, que determina quantos vértices devem ser movidos, entre cada par de fragmento;
- Mover esses vértices.

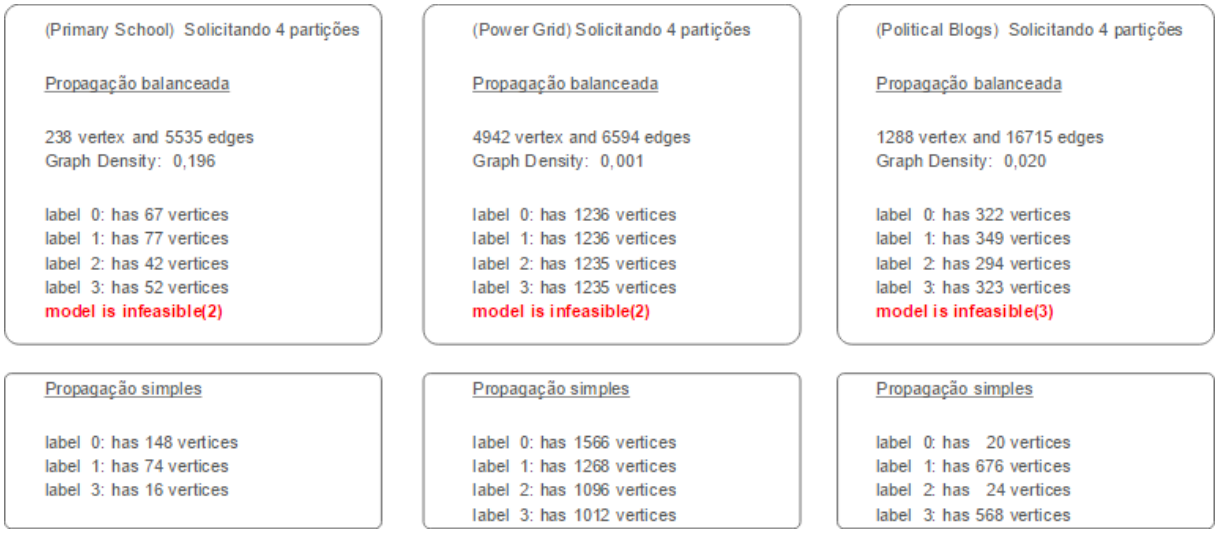

Figura 4: Comparação entre a propagação balanceada e a comum
Fonte: Elaboração própria.

noend 6 Algoritmo de propagação de rótulos comum adaptado de (UGANDER, 2014)

Require: G = (V,E)

Ensure: $P = \{V_1, V_2, ..., V_n\}$

1: Inicialize P com —V— conjuntos que contém um único vértice
2: **while** P for modificado **do**
3: **for** $u \in V$ **do**
4: $L(u) \leftarrow$ novo rótulo

Quando comparado com uma propagação de rótulos normal, a única diferença é que em vez de mover todos os vértices que teriam sua partição maximizada, o algoritmo para, resolve um programa linear, e então prossegue movendo tantos vértices quanto possível sem prejudicar o equilíbrio. Tal como acontece com a propagação de rótulos comum, a maior parte do trabalho consiste em determinar para onde os vértices preferem mover-se.

A Figura 4 compara a propagação balanceada (superior) e a comum (inferior) para três grafos distintos. No experimento criado para elaborar a Figura 4 foram rodados, para três grafos distintos, um algoritmo de propagação de rótulos comum, levando em consideração apenas a vizinhança simples, e uma adaptação do algoritmo de propagação balanceada de (UGANDER, 2014). Como mostrado pela imagem, a propagação comum extrapola os limites máximos e mínimos das partições, enquanto o algoritmo balanceado as mantem.

A figura 5 ilustra o resultado final da propagação comum: em (a) o estado inicial dos rótulos, em (b) o resultado da propagação balanceada e em (c) o resultado da propagação comum. É possível notar a diferença entre ambos os métodos já que a balanceada tentou dividir

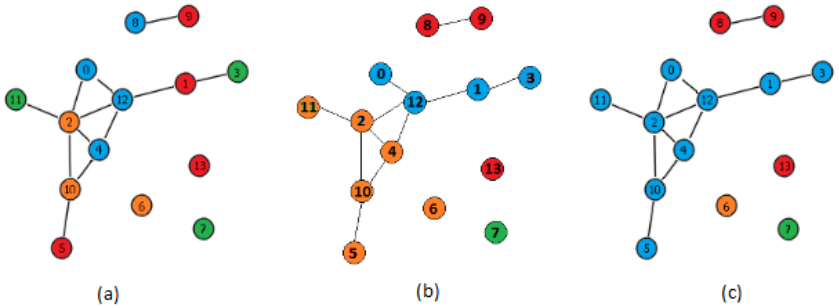

Figura 5: Comparação do resultado final de propagação entre a balanceada e a comum
Fonte: Elaboração própria.

o componente conexo maior em dois, já a propagação comum rotulou (corretamente) todos os vértices possíveis a partir dos seus vizinhos.

Quando o algoritmo executa sob uma inicialização aleatória, nota-se que as melhorias locais feitas pelo algoritmo tendem a não descobrir estruturas geográficas grandes. A inicialização deste influencia bastante no resultado final e em sua qualidade, para melhorar essa inicialização foi utilizada informações sobre a localização do vértice, já que para cada usuário é atribuída uma cidade, isso agregou uma grande melhoria ao resultado final do processo.

Para segregar geograficamente os vértices o algoritmo primeiro acha a cidade com o maior custo ainda não atribuído, essa cidade é selecionada como centro do novo corte, todas as outras cidades com um custo diferente de zero são ordenadas de acordo com sua distância com a cidade central. Para assegurar que o algoritmo não ultrapasse os limites geográficos foi introduzido para todas as cidades no mesmo pais uma distância negativa. Começando com a cidade central, toda a lista ordenada é processada e cada cidade é atribuída a um corte que está em construção. Quando o algoritmo muda de país, este dá as cidades desse pais uma nova distância negativa, relativa a nova cidade central de maior custo. Quando não há mais espaço no corte, uma fração dos vértices dessa cidade é removido e subtraído do valor total da cidade (esses vértices remanescentes entrarão em outro corte). O resultado inclui vários cortes centrados sobre uma cidade que contém todos os vértices dentro de um certo raio da cidade central e equilibrado no custo. Depois dessa agregação inicial, o algoritmo passa a operar apenas nesses cortes e não mais no grafo todo, tornando possível executar a atribuição de cada corte a uma máquina (partição). Os experimentos mostraram que usando somente a inicialização geográfica, sem a propagação, 52.7% das arestas locais (número de arestas que são atribuídas a um mesmo corte de uma partição), e quando uma iteração da propagação é utilizada, esse valor vai para 71.5%, enquanto a média da inicialização aleatória fica em 44.8%. Tal diferença é apresentada na Figura 6 que mostra a comparação entre a inicialização aleatória e a geográfica no que diz respeito a arestas locais, ao lado direito a proporção de arestas locais

após cada iterações, e ao lado esquerdo a quantidade de vértices movidos em cada iteração. Para prevenir muitas mudanças nas duas primeiras iterações, estipulou-se que somente os vértices que tivessem um ganho de vizinhos maior que 1 fossem alterados, isso foi grafado na legenda da figura com o * na primeira e segunda iteração.

A complexidade do algoritmo não está na quantidade de vértices nem arestas, mas é realçada pela quantidade de partições (coalizões). Em um teste prático realizado em uma mesma configuração de hardware, dividir o grafo do Live Journal com 8.4 milhões de vértices e 42.9 milhões de arestas não direcionadas (foi convertido de direcionado para não direcionado) em 20, 40 e 100 partições levou 3min, 8min e 88min.

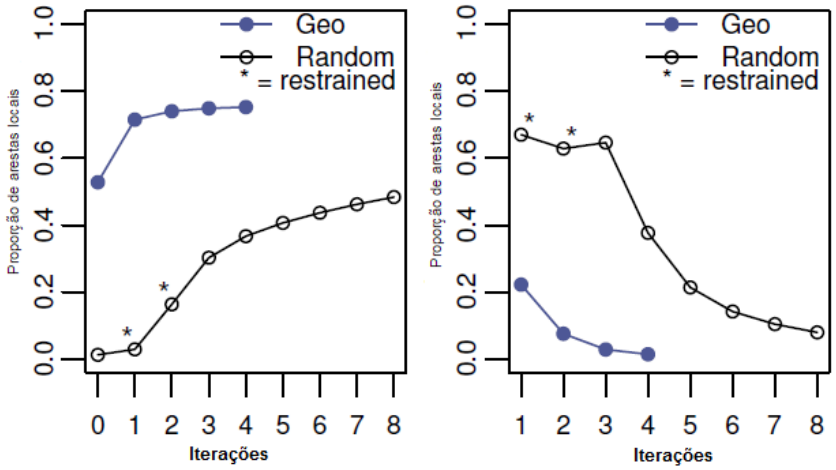

Figura 6: Comparação entre inicialização aleatória e a geográfica.
Fonte: (UGANDER, 2014)

2.4 DISCUSSÃO DO CAPÍTULO

Neste capítulo foram vistas as soluções que tratam da geração de estruturas de coalizão, que foram variadas, desde enumeração exaustiva que tem como limitador a própria quantidade de vértices, até algoritmos heurísticos que trabalham com cliques, subgrafos conexos, programação linear, genéticos... cada um com sua abordagem, limites e conquistas. A importância deste capítulo é mostrar os estudos sobre esse tema na atualidade, deste sendo possível distinguir caminhos para possíveis soluções de escalabilidade e evitar utilização de métodos já tidos como ineficientes, pois como estudado em (NUNES, 2015) somente mudar a estrutura do grafo (para esparsos) não é uma solução por completo já que quando procurado por subgrafos conexos levaria ao mesmo problema exponencial.

3 PROPOSTA DE MODIFICAÇÃO DE UM ALGORITMO DE PARTICIONAMENTO PARA GERAR ESTRUTURAS DE COALIZÃO

Nesta seção apresenta-se a proposta desta dissertação de mestrado: a modificação do algoritmo de (UGANDER, 2014), seu novo comportamento e funcionamento. Também são apresentadas outras formas de valoração que direcionam o contágio e alteram o valor final da estrutura.

3.1 ALGORITMO DE PROPAGAÇÃO PARA GERAÇÃO DE ESTRUTURAS DE COALIZÃO (APGEC)

O algoritmos de (UGANDER, 2014), para particionamento, tem como base três passos: No primeiro é feita a propagação de rótulos comum, na qual cada vértice participa do bloco que mais aparece na sua vizinhança (de distância um); Na segunda é feita a execução de um programa linear que restringe a movimentação dos vértices, dos blocos que estão para as que mais aparecem em sua vizinhança, com o intuito de deixar as partições equilibradas no que diz respeito a quantidade de vértices em cada um deles (o objetivo é um particionamento igualitário e minimizar as arestas que saem de um partição para outra); No último passo os vértices se movimentam de acordo com o permitido pelo passo anterior.

A primeira modificação para a criação do algoritmo proposto nessa dissertação é a propagação de rótulos, que não se propagará da forma comum (pela quantidade de vizinhos) mas por um jogo no qual cada coalizão c tem um valor dado por:

$$vc(c) = (d * \rho) + (\frac{\sum_{v \in c} val(v)}{|c| * \max_{v \in c} val(v))} * (1 - \rho)) \quad (1)$$

Na Equação 1, d é a densidade da coalizão c, ρ é o peso para as inter-relações e $val(v)$ o valor de cada vértice da coalizão c. O vértice adquirirá o rótulo da coalizão que dê a ele maior ganho (que maximize o valor de $vc(c)$).

O comportamento da alteração do contágio pode ser visto na Figura 7 que embora não tenha sido criada com base na equação acima, dá a visão de como o método de contágio pode

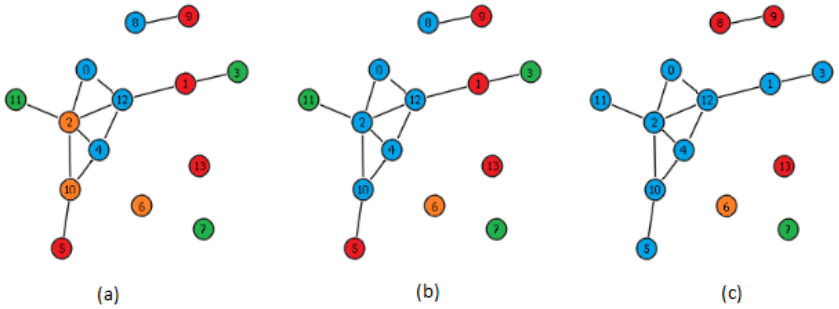

Figura 7: Resultado final ao modificar o método de contágio no algoritmo de propagação de rótulos.
(a) estado inicial, (b) resultado da propagação pelo Coeficiente de Agrupamento, (c) resultado da propagação comum. Fonte: Elaboração própria.

influenciar no resultado final do particionamento. Em (a) é mostrado o estado inicial do grafo, em (b) o resultado da propagação quando o contágio é feito por Coeficiente de Agrupamento, em (c) o resultado da propagação quando o contágio é feito por quantidade de vizinhos (propagação comum).

A função característica do vértice (para o cálculo da Equação 1) é realizada pela adição deste, e suas arestas, em cada uma das coalizões a qual ele tem acesso (vizinhos de grau um). Essa valoração implica que vértices com o mesmo nível de habilidade se juntem e vértices com habilidades distantes se afastem, o que é interpretado como a dificuldade de interação entre pessoas com habilidades distantes.

A segunda modificação é a remoção de todo o programa linear, que é oneroso de manter, já que em cada iteração uma nova gama de regras seria criada e executada. No lugar do programa linear foi utilizada a comparação simples entre o valor da estrutura gerada pela não movimentação do vértice (permanência na coalizão na qual se encontra) e o valor gerado pela sua movimentação (permitir a movimentação do vértice para a nova coalizão que dará a ele maior ganho). Essa segunda etapa é feita pelo cálculo do valor da estrutura antes da movimentação e após a movimentação para cada vértice que está na fila para movimentar-se. Esse cálculo é dado por:

$$v(cs) = \frac{\sum_{c \in CS} v(c)}{|CS|}. \tag{2}$$

Dada as modificações acima tem-se um algoritmo heurístico que recebe três parâmetros:

1. Um grafo G não direcionado que será particionado. O grafo como um todo é visto como uma estrutura de coalizão (grafo particionado e valorado).

2. Um valor *MAX_COALITIONS* com a quantidade máxima de coalizões dentro de uma estrutura. Esse parâmetro serve como restrição, já que a complexidade computacional do algoritmo está diretamente ligada a quantidade de escolhas que o vértice tem (a quantas outras coalizões ele está ligado). Diminuir esse valor é uma forma de acelerar o processo de busca por uma coalizão melhor, mas pode levar a baixos valores de ganho pois os vértices não teriam escolhas melhores a fazer. Diminuir esse valor é forçar um vértice a participar de uma coalizão que não daria maior ganho para o vértice.

3. O peso ρ dado ao relacionamento entre os vértices. Esse parâmetro é uma forma de privilegiar a habilidade ou o relacionamento dos agentes em uma coalizão. Ele é um valor real no intervalor [0, 1].

o primeiro é um grafo G não direcionado que será particionado, o segundo um valor *MAX_COALITIONS* com a quantidade máxima de coalizões dentro de uma estrutura e o último é o peso, ρ, dado ao relacionamento entre os vértices. Esse algoritmo é dividido em duas etapas: A primeira é a propagação de rótulos que tem uma visão gulosa do ponto de vista do vértice. A segunda é uma restrição à movimentação que limita a movimentação do vértice para não diminuir o ganho geral da estrutura.

A primeira etapa (propagação) o vértice é colocado em uma coalizão que dará melhor possibilidade de executar uma tarefa (maior valor dado pela Equação 1). Para tal, o vértice, e suas arestas, são colocados em cada uma das coalizões a qual ele tem acesso (coalizões vizinhas de grau um) e a Equação 1 é executada para cada uma dessas coalizões vizinhas. Se houver alguma coalizão que seja melhor que a atual ele irá para uma fila de mudança.

Na segunda etapa (restrição) o valor da estrutura é calculado para cada vértice na fila de mudança logo, é calculado pela Equação 2 o valor da estrutura antes da movimentação desse vértice e após a sua movimentação. Seu comportamento básico pode ser visto pelo Algoritmo 7. A movimentação do vértice é permitida ou não baseado no incremento do valor da estrutura. Os vértices não sabem quem está na lista de mudança e a dinâmica de processamento na lista pode alterar os valores futuros dos vértices, fazendo com que o valor esperado de ganho pelo vértice não seja mais o correto.

Esse processo de contágio (etapa 1) e movimentação (etapa 2) é realizado cinco vezes, que é número de iterações para que 95% dos vértices sejam rotulados corretamente (RAGHAVAN et al., 2007) (se estivéssemos utilizando o padrão de propagação pela maioria dos rótulos vizinhos) mas esta métrica serve de ponto inicial para este estudo.

A densidade foi escolhida como propriedade pela sua capacidade de representar uma

clique nos grafos, uma representação da coesão social. Cliques são comumente utilizados para representar comunidades ou coalizões e acredita-se ser o cerne do problema da geração de estruturas de coalisão, como mostrado por (NUNES, 2015).

noend 7 Algoritmo de Propagação para Geração de Estruturas de Coalizão (APGEC) simplificado

Require: G = (V, E), MAX_COALITIONS, ρ

Ensure: $P = \{c_1, ..., c_m\}$

1: Inicialize os vértices para as k coalizões
2: **for** 1 até 5 **do**
3: $V \leftarrow V$ embaralhado
4: L(V) $\forall v \in V, L(v) \neq \emptyset$ {Rotina de propagacao de rotulos}
5: M(V) $\forall v' \in V, CS \cup L(v) < CS \cup L(v'), L(v) \leftarrow L(v')$ {Rotina de movimentacao dos vertices}

Como pode ser visto na função de valoração representada pela Equação 1, ela será maximizada quando existir uma clique fazendo desta uma função superaditiva. Mas em um grafo do mundo real, a possibilidade deste ser completo (onde todos os vértices têm ligação com todos os outros) pode ser descartada, por isso a função não será superaditiva em grafos do mundo real.

O algoritmo prevê um limite superior teórico, um valor ao qual nenhuma coalizão ou estrutura consegue ultrapassar, este é um número real dado pelo intervalo $[0, 1]$. Os valores apresentados nos experimentos seguem esses limites, sendo 1 o maior valor possível de uma estrutura (não necessariamente alcançável).

O Algoritmo 8 contém a lógica dos dois passos mencionados anteriormente (Algoritmo 7), requer um grafo $G = (V, E)$, não direcionado, com valores nos vértices (que significa a habilidade), os pesos das arestas não é considerado e tem como saída um grafo rotulado em que cada rótulo é uma coalizão, sendo uma coalizão uma partição do grafo e este uma estrutura. O primeiro passo é embaralhar a ordem em que os vértices estão, depois é criado um conjunto vazio (M) que representa os vértices que pretendem mudar de coalizão, então é verificado o qual coalizão maximiza o ganho de cada vértice de acordo com as coalizões que este tem acesso (são identificadas as coalizões vizinhas do vértice através da função $N(v)$ e depois em quais rótulos esses vizinhos estão através da função $L(N)$) que é calculado pela função de valoração de coalizões, se esta função retornar que o rótulo em que o vértice está não é o melhor para ele então este é adicionado ao conjunto M de mudança. Em seguida é realizado para cada vértice dentro do conjunto M o cálculo de valoração da estrutura, que se retornar um valor melhor para a estru-

tura permite que o vértice troque de rótulo. Depois que o vértice entra na lista de mudança, ele é avaliado na partição na qual se encontra e na partição que deseja estar, se o valor da estrutura aumentar é permitida sua movimentação.

A Figura 8 mostra o passo a passo da movimentação dos vértices do grafo, este é um exemplo de como o algoritmo funciona na prática, são apresentados os valores de cada iteração para a estrutura embora não sejam dados os valores individuais dos vértices. Em (a) o estado inicial do grafo # 01 Clube de Karate, onde todos os vértices pertencem a uma coalizão distinta, este representa o estado inicial do algoritmo quando MAX_COALITION é igual a $|V|$, aqui o valor da estrutura é 0,5 e se encontram 34 coalizões. Em (b) é o resultado da primeira iteração do algoritmo com base na Equação 1, na qual cada vértice recebeu a melhor coalizão para si e se movimentou baseado na Equação 2. Essa primeira iteração faz com que o valor da estrutura seja 0,622 e sejam formadas 25 coalizões. Em (c) mais uma iteração é realizada e o valor da estrutura chega em 0,695 e agora com 23 coalizões. Em (d) é executada a terceira iteração, o valor da estrutura é 0,732 e a quantidade de coalizões é 16. Em (e) a penúltima iteração o valor da estrutura é 0,803 com 12 coalizões. E na última, e quinta iteração, com 12 coalizões ainda, o valor da estrutura é 0,806.

noend 8 Algoritmo de Propagação para Geração de Estruturas de Coalizão (APGEC)

Require: G = (V, E), MAX_COALITIONS, ρ

Ensure: $P = \{P_1, P_2, P_3, ..., P_n\}$

1: Inicialize os vértices para as MAX_COALITIONS coalizões
2: **for** 1 até 5 **do**
3: $V \leftarrow Vembaralhado$
4: $M \leftarrow \{\}$
5: **while** $v \in V$ **do**
6: $maxVal \leftarrow 0$
7: $chooseLabel \leftarrow L(v)$
8: $initialLabel \leftarrow L(v)$
9: $N \leftarrow N(v)$
10: $H \leftarrow L(N)$
11: **while** $h \in H$ **do**
12: $L(v) \leftarrow h$
13: $C \leftarrow$ Pega-se todos os vértices marcados com o mesmo label de v
14: $vc(C) \leftarrow vc(C, \rho)$
15: **if** $vc(C) > maxVal$ **then**
16: $maxVal \leftarrow vc(C, \rho)$
17: $chooseLabel \leftarrow h$
18: $L(v) \leftarrow initialLabel$ {Fim da propagação de rótulos}
19: **if** $initialLabel != chooseLabel$ **then**
20: $v' \leftarrow v$
21: $L(v') \leftarrow h$
22: $M \leftarrow v'$
{Inicio da movimentação}
23: **while** $m \in M$ **do**
24: $P \leftarrow getPartitions(G)$
25: $valueOld \leftarrow vc(P)$
26: $G' \leftarrow G/v \bigcup v'$
27: $P' \leftarrow getPartitions(G')$
28: $valueNew \leftarrow vc(P')$
29: **if** $valueNew > valueOld$ **then**
30: $L(v) \leftarrow L(v')$

A propagação de rótulos modificada é adicionada ao algoritmo (linhas 3 até 18) para criar um processo de contágio e acelerar o cálculo dos valores das coalizões. A restrição de movimento (linhas 23 até 30) também foi utilizada como limitador em outros trabalhos como em (UGANDER, 2014) e aqui, ao invés de utilizar programação linear, é utilizado uma comparação de resultados.

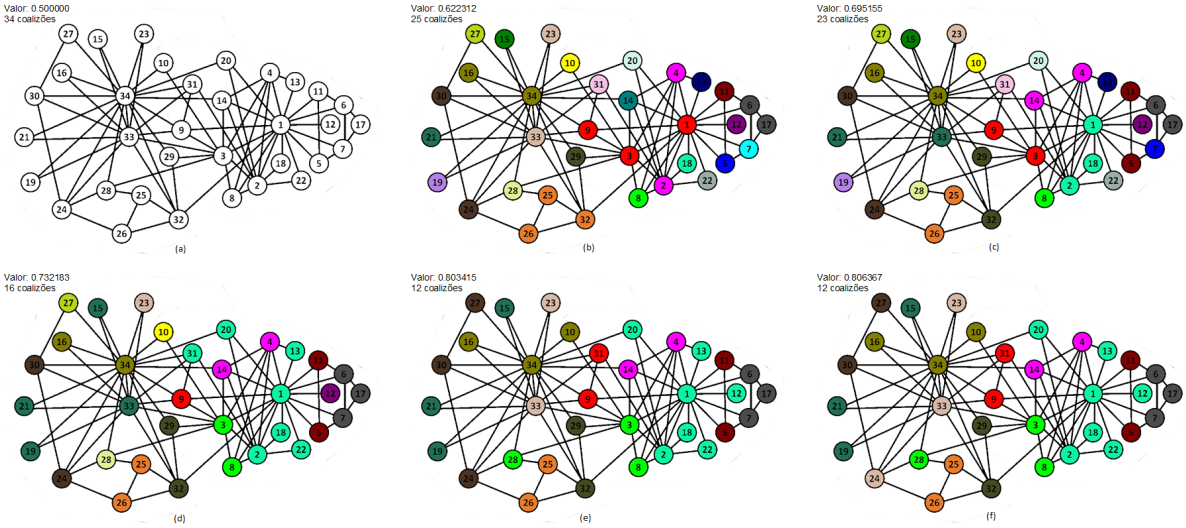

Figura 8: Passos da propagação modificada com o contágio direcionado pela Equação 1

Em (a) o estado inicial, em (b), (c), (d) e (e) os estados intermediários e em (f) o estado final da propagação modificada.

Outras formas de valoração podem ser utilizadas como:

1. Dado um grafo não direcionado G = V, E, com pesos positivos para cada aresta em E, é definido um jogo no qual cada coalizão c tem um valor $v(c) = \sum_{\{i,j\} \in E : i,j \in c} v(i,j)$, denominada *Edge Sum Coalition Valuation Function* (Função de Valoração pela Soma das Arestas) (VOICE et al., 2012). Esta função tem por intuição que tipicamente, um par de vértices pode ser associado com um peso indicando seus potenciais de eficiência mutuas ao exercer de uma tarefa. Os pesos podem ser positivos (eficiência) ou negativos (ineficiência) representando as divergências de relacionamento entre os vértices os quais tem relação direta com a eficiência ou não da coalizão como um todo. O valor da coalizão é calculado pelo total dos pesos das arestas entre seus participantes (intra-cluster) e tem sua complexidade estudada em (DENG; PAPADIMITRIOU, 1994) para diferentes métodos de computa-la. Esse tipo de valoração não leva em consideração externalidades.

2. Pode-se modificar a função acima para: $vc(c) = |E^+(c)| + |\bar{E}^-|$, onde $E^+(c) = \{\{i,j\} = +|i,j \in c\}$ que resulta em um conjunto de arestas intra-cluster positivas e $\bar{E}^-(c) = \{\{i,j\} = -|i \in c, j \notin c\}$ que resulta em um conjunto de arestas inter-cluster negativas com uma terminação em c, esta função é nomeada como *Correlation Coalition Valuation Function* (Função de Valoração por Correlação de Coalizão) (VOICE et al., 2012). Portanto o valor de uma coalizão é dado pela soma das arestas positivas internas a ela e negativas que tem uma terminação na coalizão. Esse tipo de valoração não leva em consideração externalidades.

3. Também proposta por (VOICE et al., 2012) é a quantificação dos vizinhos que promove uma aproximação maior entre seus membros com outras coalizões, a coalizão portanto admite membros que tem vizinhos em comum com outras e é dado pela formula: $vc(c) = \sum_{i \in c} n_i(c)$, onde $n_i(c)$ é o número de pares de vértices $\{\tau, \iota\}$ ambos pertencentes a V tal que $\tau \in c$ e $\iota \notin c$ e $\{i, \tau\} \in E$ e $\{i, \iota\} \in E$, para uma 3-cliques, onde dois vértices pertencem a coalizão e outro não, essa valoração recebe o nome de *Coordination Coalition Valuation Function* (Função de Valoração por Coordenação de Coalizões). Esse tipo de valoração não leva em consideração externalidades.

4. A valoração proposta por (BRANDES et al., 2008) chamada Função de Valoração por Modularidade, sendo modularidade um índice de qualidade para *clustering* (redes com maior modularidade tem conexões densas entre seus vértices, mas conexões esparsas entre comunidades diferentes), é dado por: $v(C) = \frac{|E(c)|}{|E|} - (\frac{|E(c)|+|\bar{E}(c)|}{2|E|})^2$.

A seguir são descritos os experimentos, os dados utilizados, sua coleta e procedimentos de transformação sobre os dados, o modelo adotado para execução dos experimentos, coleta de dados e normalização. Por fim é apresentada o método utilizado para comparar os resultados e investigar suas causas e consequências.

4 EXPERIMENTOS E RESULTADOS

Nesta pesquisa busca-se uma otimização nos algoritmos de formação de estruturas de coalizões em redes complexas e, para isto, o método utilizado para a comprovação dos resultados será o empírico que é uma pesquisa científica sistemática e controlada por meio de experimentos que podem ser replicados (WAZLAWICK, 2009). Ao final do processo será elaborada uma tabela comparativa de desempenho entre os algoritmos existentes (algoritmos exatos e heurísticos) obtidos por uma revisão bibliográfica e dados do algoritmo proposto neste trabalho. Foram, também, definidos que algoritmos serão estudados; sua origem; os grafos sobre os quais eles serão aplicados e a origem destes grafos; os materiais utilizados para elaboração ou auxílio na confecção dos algoritmos e heurísticas, assim como os resultados obtidos por esses algoritmos; e, os materiais ou ambientes utilizados para executar os códigos e computar as propriedades.

Este trabalho foi guiado por uma pesquisa bibliográfica e com a experimental, pois proveram conhecimento sobre os algoritmos já criados, seus avanços, seus problemas e ajudaram a localizar suas principais falhas. Os algoritmos estudados foram aqueles listados por (NUNES, 2015) e (RASMUSSEN; LARSEN, 2011) que são exatos e heurísticos. Estes foram escolhidos por representarem um progresso cronológico na evolução dos algoritmos. As pesquisas foram relevantes para reconhecer a importância da identificação de comunidades como meio de gerar estruturas de coalizão. Vários testes foram feitos neste contexto para encontrar as estruturas a partir de comunidades.

O algoritmo de (UGANDER, 2014) foi a principal referência para a elaboração do novo algoritmos que tem três pontos chave: O primeiro é a forma de contágio, que normalmente se dá apenas pela vizinhança, este item foi revisto substituindo por uma função que maximiza o valor do objetivo do agente nas coalizões (rótulos) vizinhas. O segundo é a função de valoração da coalizão ou estrutura de coalizão que privilegia a aglomeração de agentes com nível de habilidade aproximada. O último é a capacidade de dimensionar o tamanho das coalizões na estrutura, fazendo com que este seja uma restrição e consequentemente acelere o processo de cálculo.

Tabela 2: Grafos utilizados para o estudo e seus tipos.

ID	Nome	vértices	arestas	tipo
# 01	Karate	34	78	Social
# 02	Dolphin	63	159	Biológica
# 03	Les Miserables	78	254	Social
# 04	Books about US Politics	106	441	Econômica
# 05	Copperfield Word Adjacencies	113	425	Linguagem
# 06	American College Football	116	613	Esportiva
# 07	Airlines	235	1,297	Infraestrutura
# 08	Contact Networks in a Primary School	238	5,539	Social
# 09	Neural Network (C Elegans)	298	2,148	Biológica
# 10	Codeminer	724	1,015	Infraestrutura
# 11	Political Blogs	1,288	16,715	Política
# 12	Diseasome	1,419	2,738	Biológica
# 13	Coauthorships in network science	1,480	2,742	Social
# 14	Power Grid	4,942	6,594	Infraestrutura
# 15	Wikipedia voting	7.115	100.762	Social

Fonte: Adaptado de (NEWMAN, 2013; LESKOVEC; KREVL, 2014)

4.0.1 EXPERIMENTO 1

Este experimento tem como objetivo comparar o desempenho do algoritmo proposto com o exato ODP-IP (RAHWAN et al., 2013), disponibilizado em Java (RAHWAN, 2014) e utilizado exatamente como distribuído.

Foi utilizada a pesquisa bibliográfica para localização dos problemas, análise do estado da arte e avanços dados nos últimos anos no que se refere a Geração de Estruturas de Coalizão.

Este experimento foi executado em um uma máquina Dell Latitude, com 16GB DDR3 de RAM e processador quadi-core i5 2520M de 2,5GHz. Foi utilizado o Microsoft Excel (do pacote Microsoft Office 2013) para criar os gráficos e os cálculos de proporção entre os valores encontrados do exato e proposto. Também foi utilizado o software Eclipse Neon 4.6.0 para compilar o programa ODP-IP e o proposto.

A comparação foi realizada na amostra de 17 redes completas, gerando-se um grafo K_n, com n sendo o número de vértices do grafo completo K, indo de 5 a 21. O limite para a execução do algoritmo ODP-IP é K_{30} (um grafo completo com 30 vértices) devido a limitações da linguagem e implementação. Ambos os algoritmos (ODP-IP e proposto) foram executados sob a mesma rede (grafo completo) e sob a mesma valoração de coalizões para cada uma das 17 redes completas.

Cada vértice pertence a uma coalizão com o parâmetro MAX_COALITIONS = $|V|$ e

ρ não foi utilizado já que a função de valoração vem do próprio algoritmo exato. O algoritmo exato ODP-IP é aplicado em cada grafo completo, gerando o Conjunto Potência de coalizões possíveis e as valorando através de uma distribuição normal, depois procura o valor da melhor estrutura. O algoritmo heurístico é aplicado sobre o mesmo K_i, com a mesma valoração e então busca a estrutura de maior valor. Cada rede foi valorada, calculada a estrutura ótima pelo ODP-IP e calculada a melhor estrutura encontrada pelo algoritmo proposto dez mil vezes. Para cada execução foram anotados os valores encontrados para: valor da estrutura encontrado após sua finalização, o tempo para encontrar a solução (não contando o tempo de leitura de arquivos ou gravar nos arquivos dos resultados). Como o algoritmos ODP-IP é ótimo, a comparação foi realizada, em porcentagem, do quão distante o algoritmos proposto está desta solução ótima, representado por: $\frac{val_{proposto}}{val_{ODP-IP}}$. Para facilitar e automatizar o processo, um terceiro programa foi criado para executar o exato e em seguida o proposto com os mesmos valores, este é representado pelo Algoritmo 9.

noend 9 Algoritmos automatizador de execução

```
1:  for k = 1 to 5 do
2:      Instanciar ODP-IP
3:      getNumOfAgents ← k
4:      opção StoreCoalitionValuesToFile ← checked
5:      opção PrintDetailedResultsOfIPToFiles ← checked
6:      for 1 to 10.000 do
7:          Executar ODP-IP com as configurações citadas
8:          Salvar resultado em disco
9:          Executar Algoritmo de Propagação para Geração de Estruturas de Coalizão
10:         Salvar resultado em disco
```

Para realizar o primeiro experimento primeiramente o código do ODP-IP foi executado (classe SolverRandomProblems) com a configuração abaixo:

1. ”Solve the CSG problem based on the user’s input”marcado;
2. ”Normal Distribution”marcado;
3. ”Run ODP-IP (RAHWAN, 2014)”marcado;
4. ”Store coalition values in a file”marcado;
5. ”Print the interim results of IP to an output file”marcado;

6. ”Number of agents”configurado inicialmente como 5 e a cada ciclo de dez mil execuções é incrementado em 1 até 21.

Ao executar o ODP-IP com as configurações acima são gerados arquivos com os resultados. Destes, apenas dois são considerados: O primeiro é o arquivo ”kAgents_Normal_1.txt”, onde k é o número de agentes escolhido. Este arquivo contém a distribuição de valores de cada coalizão. O segundo arquivo é o ”valueOfBestCSFoundByIP.txt”que contém o valor da estrutura ótima e o tempo em milissegundos para encontrar essa solução. Esse programa foi executado dez mil vezes para cada número de agentes de 5 a 21 e seus arquivos salvos. Após cada execução do ODP-IP, e com os arquivos salvos, o algoritmo proposto foi executado extraindo as informações dos arquivos salvos do ODP-IP. O primeiro passo é utilizar as informações do arquivo ”kAgents_Normal_1.txt”, onde k é o número de agentes escolhido, para saber o valor das coalizões. Então o algoritmo proposto é executado. Ele também salva os resultados de valor encontrado e tempo decorrido em milissegundos, sem contar o tempo de leitura e gravação das informações.

Destes arquivos foram gerados dois gráficos, o primeiro contendo o melhor, o pior e as médias de valores encontrado pelo algoritmo proposto comparado com o exato (valor ótimo), portanto: $\frac{val_{proposto}}{val_{ODP-IP}}$, Figura 10. Também foi traçada uma linha de tendência que se aproximasse do comportamento da média. O segundo gráfico contém a comparação da média de tempo do exato contra da média de tempo do heurístico mostrado na Figura 9.

4.0.1.1 RESULTADOS DO PRIMEIRO EXPERIMENTO

Os resultados obtidos no primeiro experimento que é a comparação entre o algoritmo heurístico criado e o exato ODP-IP, mostrado pela Figura 9, mostram que o algoritmo heurístico superou a quantidade de vértices em um grafo que nos algoritmos exatos são uma restrição. Enquanto o ODP-IP chega a 30 vértices, o heurístico apresentado pode calcular grafos muito maiores, com diferentes qualidades de respostas de acordo com a restrição no número máximo de coalizões em uma estrutura. No comparativo depois de 10 mil execuções, pode ser visto que quanto maior o grafo maior a distância da solução ótima. Tanto a maior diferença quanto a menor também ficam mais distantes da solução ótima como mostrado pela Figura 10, não há dados suficientes para dizer se este declínio é linear ou não. A progressão de tempo conforme a quantidade de vértices aumenta, mostrada na Figura 9, pode-se notar que o ODP-IP é mais rápido até k_{15}, após o heurístico passa a ser mais rápido. A progressão decrescente da qualidade da resposta entregue pelo algoritmo heurístico, mostrada na Figura 10, pode-se notar que a qualidade de resposta se mantem perfeita até k_{12}, após esse tamanho a qualidade decai, mas não

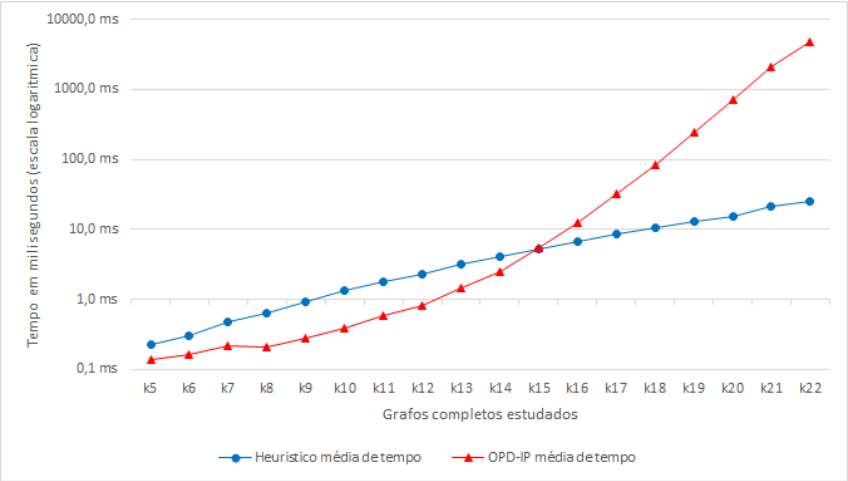

Figura 9: Comparação dos tempos entre o odp-ip e o algoritmo heurístico proposto
Fonte: Elaboração própria.

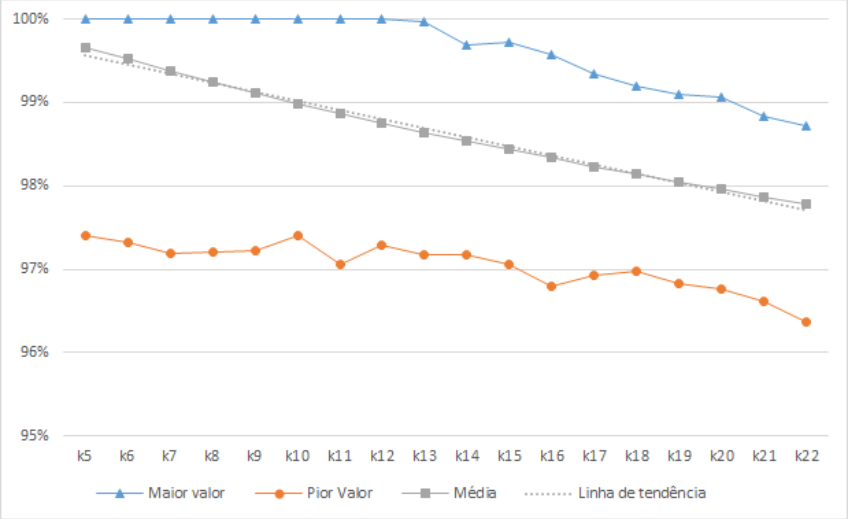

Figura 10: Comparação da qualidade de resposta do algoritmo heurístico proposto com o exato ODP-IP
Fonte: Elaboração própria.

é possível determinar seu comportamento pela amostragem.

4.0.2 EXPERIMENTO 2

Este experimento tem como objetivo comparar o desempenho e limites do algoritmo proposto.

Foi utilizada pesquisa experimental para verificar limites de tempos do algoritmo proposto e verificar se é possível encontrar um número que fornece um valor bom para a coalizão e também o deixe rápido a ponto de poder calcular estruturas em grafos maiores.

Esse experimento foi executado tanto em uma máquina com Windows quanto com Linux, em cada sub-experimento é descrita a configuração de hardware utilizado. Foi utilizada uma ferramenta online chamada SageMath [1] que trabalha especificamente com grafos, utilizada para converter todos os grafos em um formato único. Também foi utilizado o software Eclipse Neon 4.6.0 para compilar o algoritmo proposto. Foi utilizado o Microsoft Excel (do pacote Microsoft Office 2013) para criar os gráficos

Este experimento foi realizado em uma amostra de 14 redes, trazidas do mundo real, listada pela Tabela 2, ordenada pela quantidade de vértices, as quais podem ser encontradas em (NEWMAN, 2013) ou (LESKOVEC; KREVL, 2014). Essas redes foram escolhidas por terem diversidade tanto no número de arestas quanto de vértices (a menor com 34 vértices e a maior com 4.942), as arestas destes grafos podem ser direcionadas ou não direcionadas, alguns conexos e outras não. Não foram estudadas as estruturas interna destas redes (bipartido, planar, arvore...) como limitador do escopo desta pesquisa.

Todos os grafos direcionados foram convertidos a não direcionados a fim de se adequarem a limitação do escopo deste estudo. A valoração dos vértices foi dada por uma distribuição normal, sendo a média 3 e o desvio padrão de 1/3, em caso de valor menor que zero o valor foi substituído por zero, também para limitar o estudo. Essa valoração dos vértices ocorreu apenas uma vez para cada rede e este valor perdurou até o final deste experimento.

Este experimento foi subdividido em outros três para uma melhor análise.

4.0.2.1 EXPERIMENTO 2.1

O objetivo deste sub-experimento é averiguar o comportamento do algoritmo ao limitar ou relaxar a quantidade máxima de coalizões em uma estrutura (modificar o parâmetro $MAX\_COALITIONS$).

Este sub-experimento atribui a ρ o valor 0,5 para cada uma das 15 redes da Tabela 2 e é executado uma vez para cada configuração de $MAX\_COALITIONS$ (iniciando em 5 e dobrando até 320) e. Destas execuções são gradados os valores obtidos em arquivos os resultados: valor final da estrutura, o valor inicial da estrutura, tempo total de execução (em segundos) e quantidade de chamadas à função de valoração (tanto da estrutura quando das coalizões). O valor da coalizão foi calculado pelo próprio algoritmo proposto através da Equação 1 e 2. Este experimento foi executado em um uma máquina Linux server (Ubuntu - 3.19.0-25-generic) com 2G de RAM, com 4 núcleos de 2GHz cada.

[1] *software open source* (software de utilização livre) de matemática licenciado sob a GPL, disponível em http://www.sagemath.org)

Destes arquivos foram gerados quatro gráficos, o primeiro contendo a progressão de ganho da estrutura ao final do processo (valor final da estrutura). O segundo a progressão de ganho da estrutura ao iniciar o processo (valor inicial da estrutura). O terceiro a progressão da tomada de tempo nos diferentes grafos analisados. O último gráfico contém a quantidade de chamadas à função de valoração (função de valoração de coalizões ou de estruturas) para os diferentes grafos.

O processo adotado neste experimento pode ser visto pelo Algoritmo 10.

noend 10 Algoritmos do processo do sub-experimento 2.1

1: $\rho \leftarrow 0,5$
2: $MAX\_COALITIONS \leftarrow 5$
3: **for** each $G \in Tabela\ 2$ **do**
4: **for** $MAX\_COALITIONS <= 320$ **do**
5: Executar Algoritmo de Propagação para Geração de Estruturas de Coalizão
6: Salvar resultado em disco
7: $MAX\_COALITIONS \leftarrow MAX\_COALITIONS * 2$

4.0.2.2 EXPERIMENTO 2.2

O objetivo deste sub-experimento é verificar as modificações no resultado do algoritmo ao atribuir importância (peso) para a habilidade ou para o inter-relacionamento.

Este sub-experimento foi executado com a configuração de $MAX\ COALITIONS = 20$, para cada uma das 14 redes da Tabela 2, foi anotado o valor final da estrutura para ρ iniciando em zero e incrementado em 0,1 até chegar em 1. O valor da coalizão foi calculado pelo próprio algoritmo proposto através da Equação 1 e 2. Este experimento foi executado em um uma máquina Dell Latitude, com 16GB DDR3 de RAM e processador quadi-core i5 2520M de 2,5GHz.

Seus passos são descritos pelo Algoritmo 11 e os dados dos arquivos foram condensados na Tabela 4.

noend 11 Algoritmos do processo do sub-experimento 2.2

1: $\rho \leftarrow 0$
2: $MAX\_COALITIONS \leftarrow 20$
3: **for** each $G \in Tabela\ 2$ **do**
4: **for** $\rho <= 1$ **do**
5: Executar Algoritmo de Propagação para Geração de Estruturas de Coalizão
6: Salvar resultado em disco
7: $\rho \leftarrow \rho + 0,1$

4.0.2.3 EXPERIMENTO 2.3

O objetivo deste sub-experimento é verificar as modificações no resultado do algoritmo ao atribuir importância (peso) para a habilidade ou para o inter-relacionamento sem a restrição de tamanho máximo de coalizões em uma estrutura.

Este sub-experimento foi executado com a configuração de *MAX COALITIONS* igual a quantidade de vértices de cada grafo, fazendo com que cada vértice iniciasse em uma coalizão distinta, para cada uma das 14 redes da Tabela 2, foi anotado o valor final da estrutura para ρ iniciando em zero e incrementando em 0,1 até chegar em 1. O valor da coalizão foi calculado pelo próprio algoritmo proposto através da Equação 1 e 2. Este experimento foi executado em um uma máquina Dell Latitude, com 16GB DDR3 de RAM e processador quadi-core i5 2520M de 2,5GHz.

Seu procedimento é descrito pelo Algoritmo 12 e os dados dos arquivos foram resumidos na Tabela 3.

noend 12 Algoritmos do processo do sub-experimento 2.3

1: $\rho \leftarrow 0$
2: **for** each $G \in Tabela\ 2$ **do**
3: $MAX\_COALITIONS \leftarrow |vertices(G)|$
4: **for** $\rho <= 1$ **do**
5: Executar Algoritmo de Propagação para Geração de Estruturas de Coalizão
6: Salvar resultado em disco
7: $\rho \leftarrow \rho + 0,1$

4.0.2.4 RESULTADOS DO SEGUNDO EXPERIMENTO

No segundo experimento os tempos variaram conforme o número de coalizões na estrutura. Outro fator determinante no tempo de processamento são os graus dos vértices, pois grafos de tamanhos muito próximos obtiveram tempos de processamento distantes apenas pela quantidade de arestas dos grafos, como pode ser visto entre as linhas # 07 Airlines e # 08 Contact Networks in a Primary School. A diferença de tempo de processamento para grafos de mesma quantidade de vértices deve-se a como o algoritmo foi construído, já que são levantadas todas as possibilidades de movimentação para todas as coalizões vizinhas e são calculados seus valores. A quantidade de chamadas à função de valoração também incrementa ao aumentar a quantidade de vértices ou a quantidade de coalizões em uma estrutura, ilustrado pela Figura 12.

Os resultados obtidos no sub-experimento exp2.1 ao avaliar o ganho demonstra um ganho entre o valor inicial de cada estrutura (valor inicial ao avaliar a estrutura quando os agentes ainda não fizeram escolha alguma de movimentação) ilustrado pela Figura 13 e o valor final da estrutura (quando a movimentação dos agentes já foi realizada e o processo finalizou) ilustrado pela Figura 14. O tempo para a execução do processo pode ser visto pela Figura 11 que ilustra a diferença de tempo entre o processamento de um grafo pequeno com 34 vértices e um com pouco mais de 7 mil vértices.

Ainda no segundo experimento, embora seja possível controlar como as coalizões são formadas (dando preferência ao inter-relacionamento ou a habilidade), nos testes realizados, as coalizões foram valoradas com uma função que privilegiava a habilidade, fazendo com que os relacionamentos fossem prejudicados. Este resultado pode ser confirmado pela Tabela 4 que mostra o melhor valor sempre quando o relacionamento é anulado (peso zero para relacionamento). Algumas redes foram imunes a variável peso, com menos de 10% de diferença entre o maior valor e o menor, enquanto outros tiveram grandes diferenças chegando a 68%. Quando o mesmo experimento é executado mas sem a restrição de número máximo de coalizões em uma estrutura, Tabela 3, nota-se uma melhora nos valores encontrados e uma melhor distribuição desses valores. Se comparado com o experimento anterior pode-se ver que no primeiro existe um gradiente de cor que vai do melhor (canto superior esquerdo) até o pior (canto inferior direito) causado pela limitação, esse mesmo gradiente não é visto quando a restrição é relaxada e este passa a ser quase horizontal, se a restrição fosse removida totalmente, seria completamente vertical.

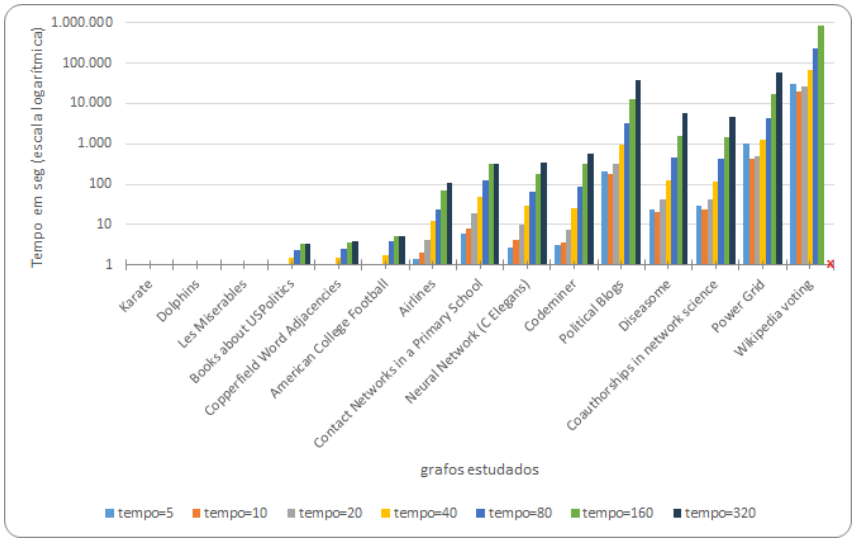

Figura 11: Comportamento quanto a tempo do algoritmo com relação ao número máximo de coalizões da estrutura.

Fonte: Elaboração própria.

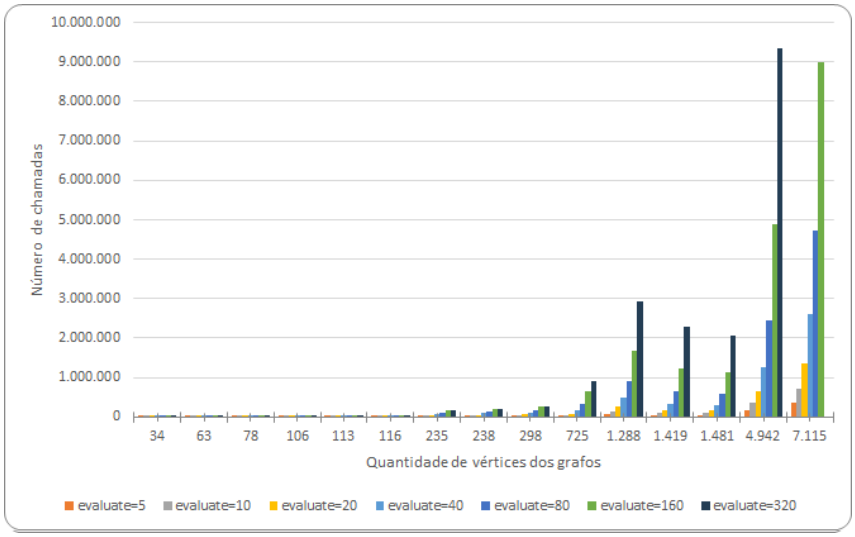

Figura 12: Quantidade de chamadas à função de valoração

Fonte: Elaboração própria.

Tabela 3: Resultado para diferentes configurações de peso com MAX_COALITIONS = $|V|$

				Peso dado relacionamento dos agentes (consequentemente o complemento é dado a proficiência)										
Nome	**#Vertice**	**Arestas**	**densidade**	**0,0**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
Karate	34	78	0,139	1,00000	0,92349	0,86643	0,82650	0,83671	0,85074	0,85313	0,85552	0,85792	0,77121	0,79286
Dolphins	63	159	0,081	1,00000	0,93165	0,89107	0,88257	0,87724	0,87389	0,86792	0,82874	0,82035	0,81196	0,83294
Les Miserables	78	254	0,085	1,00000	0,91969	0,85694	0,80596	0,82426	0,82426	0,80945	0,80399	0,78192	0,77503	0,83150
Books about USPolitics	106	441	0,079	0,99524	0,93344	0,90630	0,89946	0,90101	0,90836	0,90400	0,90076	0,90449	0,91300	0,87431
Copperfield Word Adjacencies	113	425	0,067	0,99115	0,92176	0,88684	0,86388	0,85828	0,85239	0,85163	0,84590	0,84310	0,82922	0,89007
American College Football	116	613	0,092	0,99565	0,94433	0,92987	0,93597	0,93394	0,93984	0,94879	0,95438	0,96113	0,96789	0,97000
Airlines	235	1297	0,047	1,00000	0,92298	0,86229	0,85340	0,83181	0,81633	0,78760	0,77648	0,76585	0,74077	0,81544
Contact Networks in a Primary School	238	5539	0,196	0,99576	0,97371	0,97499	0,97669	0,98002	0,98335	0,98627	0,98970	0,99313	0,99657	0,99914
Neural Network (C Elegans)	298	2148	0,049	0,99662	0,93790	0,91561	0,90511	0,89942	0,89979	0,89902	0,90410	0,90332	0,90444	0,91070
Codeminer	725	1014	0,004	0,99931	0,91305	0,84197	0,77709	0,71223	0,65497	0,72389	0,71112	0,69608	0,66677	0,32878
Political Blogs	1288	16715	0,020						0,79312	0,77192		0,72169		0,71834
Diseasome	1419	2738	0,003	0,99718		0,84853		0,78664	0,79066		0,76869		0,74773	
Coauthorships in network science	1481	2741	0,003											
Power Grid	4942	6594	0,001											

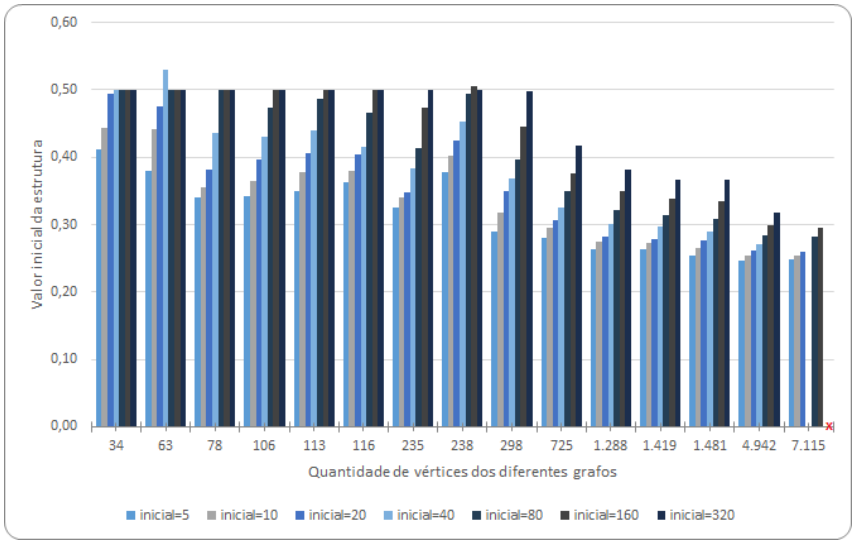

Figura 13: Progressão de ganho da estrutura ao iniciar o processo
Fonte: Elaboração própria.

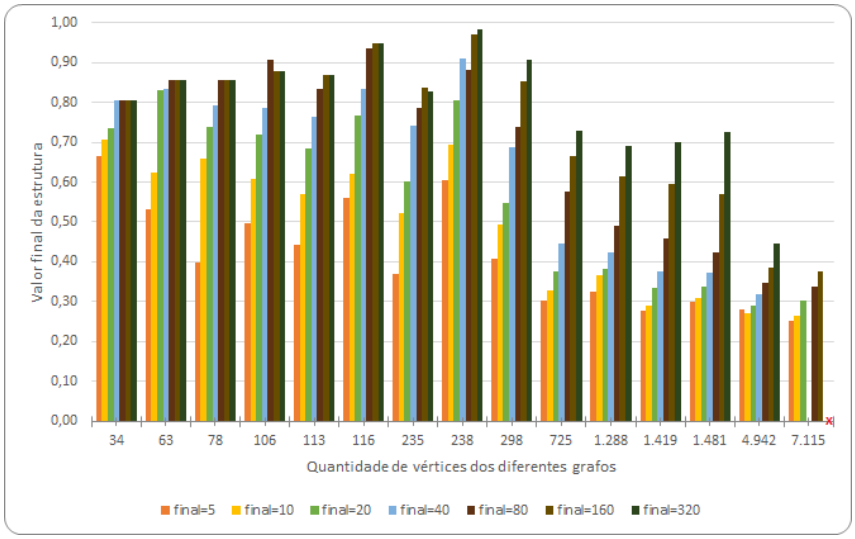

Figura 14: Progressão de ganho da estrutura ao final do processo
Fonte: Elaboração própria.

Tabela 4: Resultado para diferentes configurações de peso

				Peso dado relacionamento dos agentes (consequentemente o complemento é dado a proficiência)										
Nome	**#Vertice**	**Arestas**	**densidade**	**0,0**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
Karate	34	78	0,139	0,98616	0,90191	0,85050	0,81729	0,78489	0,77627	0,75460	0,71539	0,69694	0,67763	0,63333
Dolphins	63	159	0,081	0,94824	0,88454	0,84434	0,80378	0,80997	0,76737	0,75195	0,72215	0,71375	0,72916	0,71217
Les Miserables	78	254	0,085	0,90810	0,84823	0,82496	0,83101	0,81582	0,79067	0,78658	0,73671	0,70435	0,51899	0,56734
Books about USPolitics	106	441	0,079	0,90779	0,84652	0,80151	0,78242	0,69508	0,69879	0,71221	0,66393	0,58259	0,51670	0,42872
Copperfield Word Adjacencies	113	425	0,067	0,89640	0,84167	0,77064	0,70130	0,67704	0,65855	0,64105	0,60075	0,51598	0,53491	0,46532
American College Football	116	613	0,092	0,91700	0,85999	0,82883	0,81033	0,78407	0,79223	0,72819	0,74786	0,65078	0,68143	0,55264
Airlines	235	1297	0,047	0,88882	0,79813	0,76574	0,71409	0,63543	0,53707	0,54175	0,56479	0,49911	0,32833	0,31922
Contact Networks in a Primary School	238	5539	0,196	0,93889	0,87726	0,84448	0,78159	0,80571	0,76506	0,80749	0,86189	0,85614	0,79757	0,91967
Neural Network (C Elegans)	298	2148	0,049	0,85811	0,77813	0,70872	0,60626	0,59903	0,54007	0,52655	0,54570	0,45869	0,39706	0,41541
Codeminer	725	1014	0,004	0,74917	0,65837	0,56817	0,51201	0,43340	0,37040	0,30207	0,24917	0,19572	0,15457	0,08588
Political Blogs	1288	16715	0,020	0,68225	0,61240	0,53691	0,50016	0,43614	0,36582	0,32015	0,23134	0,19023	0,10600	0,09722
Diseasome	1419	2738	0,003	0,64331	0,57687	0,51747	0,45095	0,39015	0,32767	0,26781	0,21418	0,16018	0,10139	0,05113
Coauthorships in network science	1481	2741	0,003	0,66898	0,61216	0,55223	0,47629	0,41409	0,34964	0,28379	0,21947	0,16102	0,09779	0,04460
Power Grid	4942	6594	0,001	0,59935	0,53973	0,47982	0,41899	0,35972	0,29941	0,23881	0,18070	0,12080	0,06439	0,00781

4.0.3 EXPERIMENTO 3

Este experimento tem como objetivo comparar o desempenho do algoritmo proposto com o heurístico CFSS (BISTAFFA et al., 2017), disponibilizado em Java (BISTAFFA, 2016) e utilizado exatamente como distribuído.

Foi utilizada a pesquisa bibliográfica para localização dos problemas, análise do estado da arte e avanços dados nos últimos anos no que tange Geração de Estruturas de Coalizão.

Este experimento foi executado em uma máquina Linux server (Ubuntu - 3.19.0-25-generic) com 2G de RAM, com 4 núcleos de 2GHz cada. Foi utilizado o Microsoft Excel (do pacote Microsoft Office 2013) para criar os gráficos e os cálculos de proporção entre os valores encontrados do CFSS e proposto. Também foi utilizado o software Eclipse Neon 4.6.0 para gerar o arquivo de entrada do programa CFSS e compilar o algoritmo proposto.

A função de valoração é dada pelo próprio autor do CFSS, a Soma das Arestas com Custo de Coordenação: $v(C) = \sum_{e \in edges(C)} w(e) - K(C)$, onde $w(e)$ é o peso da aresta, $K(C)$ é o custo de coordenação que neste experimento seguiu como: $K(C) = |C|^{1,8}$. Foi executado o algoritmo proposto para cada configuração de MAX_COALITIONS 20 vezes e foi extraída a média de tempo e a média de valor, esta média de tempo é entregue como limite de tempo para a execução do CFSS, também 20 vezes. O processo é descrito pelo Algoritmo 13.

noend 13 Algoritmos do processo do experimento 3

1: $\rho \leftarrow 0,5$
2: $MAX\_COALITIONS \leftarrow 5$
3: **for** each $G \in Tabela\ 2$ **do**
4: $E \leftarrow arestas(G)$
5: **for** each $e \in E$ **do**
6: $e \leftarrow 5$
7: **for** $MAX\_COALITIONS <= 320$ **do**
8: **for** 1 to 20 **do**
9: Executar Algoritmo de Propagação para Geração de Estruturas de Coalizão
10: Salvar resultado em disco
11: $t \leftarrow \frac{\sum tempoexecucao}{20}$
12: Gerar o arquivo de entrada para o CFSS do grafo G
13: **for** 1 to 20 **do**
14: Executar Algoritmo CFSS com limite de tempo t
15: Salvar resultado em disco
16: Gerar os gráficos de comparação
17: $MAX\_COALITIONS \leftarrow MAX\_COALITIONS * 2$

A comparação foi realizada na amostra de 14 redes da Tabela 2. Isso quer dizer que o grafo # 15 *Wikipedia voting* não foi comparado devido ao seu tempo de processamento. Foi utilizada a função de valoração *Edge Sum With Coordination Cost* proposta como função para calcular a melhor coalizão e estrutura de coalizão por (BISTAFFA et al., 2017). Ambos os algoritmos (CFSS e proposto) foram executados sob a mesma rede e sob a mesma valoração de coalizões para cada uma das 14 redes. Cada rede foi valorada, calculada a estrutura pelo CFSS e pelo algoritmo proposto vinte vezes. De cada execução foram anotados os valores encontrados para: valor da estrutura encontrado após sua finalização, o tempo para encontrar a solução (sem computar o tempo de leitura de arquivos ou gravar nos arquivos dos resultados). Como o algoritmos CFSS é heurístico e não teve tempo suficiente para encontrar a solução ótima, a comparação foi realizada de forma direta e em porcentagem ($\frac{val_{proposto}}{val_{CFSS}}$).

Para essa comparação é executado o algoritmo proposto 20 vezes são anotados os valores para média de tempo de execução em segundos, a média do valor final, a média para quantidade de coalizões para a melhor estrutura encontrada, a média para a quantidade de comparações realizadas para cada configuração $MAX\_COALITIONS$. Então os grafos da Tabela 2 são convertidos para o formato de entrada do CFSS de acordo com instruções do próprio

Tabela 5: Comparação dos valores obtidos entre os algoritmos heurísticos

ID	Tempo	CFSS	Propagação	Proporção
# 01	00:00:10	25,34	61,71	41,07%
# 02	00:00:33	9,74	171,57	05,68%
# 03	00:00:43	76,72	502,52	15,27%
# 04	00:02:16	58,13	632,20	09,19%
# 05	00:03:44	56,16	186,76	30,07%
# 06	00:04:03	958,70	872,24	109,91%
# 07	02:18:44	0,00	1.067,13	00,00%
# 08	02:49:30	8.325,35	11.721,87	71,02%
# 09	05:58:01	0,00	1.009,09	00,00%
# 10	35:35:05	0,00	292,32	00,00%

algoritmo e executado com limite de tempo da média tomada pelo algoritmos proposto. A saída da execução o CFSS, com o arquivo de entrada, foi armazenado em arquivo. A saída contém o valor da estrutura encontrada (apenas se positiva), o tempo de execução do algoritmo e a quantidade de nós analisados.

Destes arquivos foi gerada uma tabela para comparação de tempo entre os dois algoritmos.

4.0.3.1 RESULTADOS DO TERCEIRO EXPERIMENTO

No terceiro experimento o algoritmo proposto ganha força ao dar um resultado em um tempo determinado e melhora esse valor conforme mais coalizões são permitidas em uma estrutura, essa restrição também demonstra uma grande otimização no algoritmo. Embora o algoritmo CFSS possa entregar resultados muito bons ele ainda cai no mesmo problema que os exatos, que é a busca quase exaustiva enquanto o tempo permitir. A comparação entre o algoritmo proposto e o CFSS é mostrado na Tabela 6 e 5, nela é apresentado o valor encontrado pelos dois algoritmos no tempo decorrido para as redes com menos de mil vértices, MAX_COALITIONS=$|V|$ e peso de aresta igualitário 5. A proporção anotada indica o valor encontrado pelo CFSS dividido pelo valor encontrado pelo algoritmo proposto. O valor zero para o CFSS em algumas redes significa que ele não conseguiu achar um valor positivo no tempo imposto (formato hh:mm:ss). Mesmo quando a quantidade de coalizões permitidas aumenta para grafos muito grandes (o que causa um incremento no tempo de execução) o CFSS não alcançou o algoritmo proposto, tendo sua vitória apenas em # 06 American College Football, superando o proposto em 10%.

Tabela 6: Comparação dos valores obtidos entre os algoritmos heurísticos para grafos com mais de mil vértices e max_coalitions = 320

ID	Tempo	Propagação	CFSS
# 11	11:46:12	10.009,61	0,00
# 12	1:28:44	101,64	0,00
# 13	0:50:47	2.932,90	2.599,43
# 14			

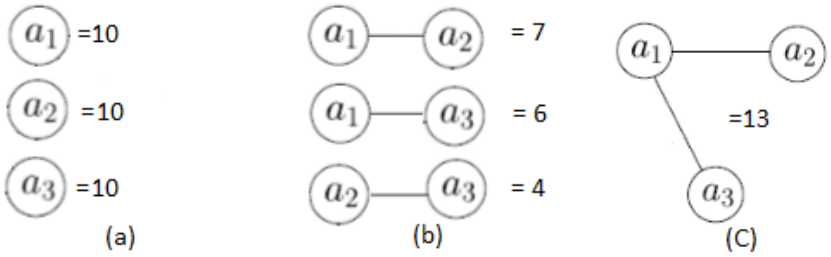

Figura 15: Estrutura gerando valor a coalizão.
Fonte: Elaboração própria.

São observadas duas grandes restrições no processo de encontrar a estrutura de coalizão: a primeira é a restrição MAX_COALITIONS; a segunda é o processo de propagação de rótulos. Com elas, foi possível o cálculo de estruturas com 5 mil agentes em pouco mais de 5h (com restrição de MAX_COALITIONS = 160). Essas duas restrições têm o mesmo problema: não permitem ao algoritmo encontrar a solução ótima. A primeira por limitar a movimentação dos agentes fazendo com que, na maioria das vezes, o número de coalizões na estrutura ótima seja maior que a estipulada ou esteja tão próximo da ideal que os agentes não conseguem se movimentar livremente até encontrar a estrutura perfeita. A segunda por não permitir certos tipos de estruturas, já que o processo de contágio é feito dois a dois e essas estruturas não são alcançadas como mostrado pela Figura 15. Em (a) pode ser visto o valor unitário de cada um dos três agentes; em (b) é visto seus valores combinados dois a dois; em (c) seus valores combinados três a três. O objetivo é mostrar que para chegar de (a) até (c) deve-se passar por (b), mas este diminui o valor da coalizão e não é permitida, assim a melhor coalizão nunca será atingida.

A Figura 16 mostra a quantidade de nós abertos pelo CFSS em comparação a quantidade de chamadas a função de valoração (valoração de coalizões + valoração das estruturas) realizada pelo Algoritmo de Propagação para Formação de Estrutura de Coalizão proposto neste estudo. Pode ser vista a quantidade de nós abertos (e avaliados) pelo CFSS durante o tempo determinado. Enquanto isso, o APFEC ao realizar muito menos operações.

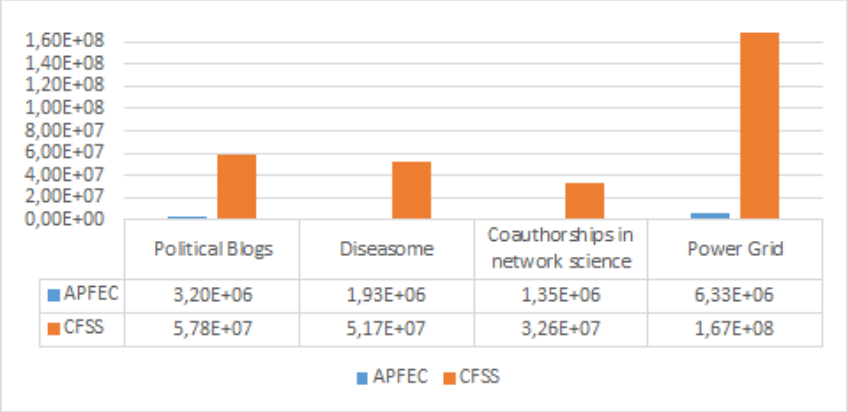

Figura 16: Comparação de nos abertos (cfss) com chamadas à função de valoração (apfec) para os grafos com mais de mil vértices.

Fonte: Elaboração própria.

4.1 DISCUSSÃO DO CAPÍTULO

Neste capítulo foram apresentados os experimentos e suas configurações e os resultados obtido. Sendo o primeiro experimento a comparação do algoritmo proposto (APGEC) com o estado da arte dos algoritmos exatos (ODP-IP), no segundo experimento foi analisado o próprio algoritmo proposto e no terceiro ele foi comparado com o estado da arte dos algoritmos heurísticos (CFSS). No capítulo seguinte serão extraídas as conclusões sobre esses resultados e possíveis trabalhos futuros para solução de alguns dos problemas encontrados como o problema dos mínimos locais e das coalizões não conexas.

5 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação propôs uma alternativa heurística que pode ser aplicada a classes gerais de grafos reais. Os resultados empíricos mostram que o algoritmo desenvolvido supera o algoritmo ODP-IP (o estado da arte quanto a algoritmos exatos para geração de estruturas de coalizão) no que diz respeito ao tamanho do grafo a ser computado, sendo que, para o maior grafo estudado, o k_{21}, o pior valor encontrado após 10 mil execuções foi de 96.36% do valor ótimo (Figura 9), e somente foi superado pelo CFSS em uma rede, e nesta foi superado em 10%. Neste momento é possível responder as perguntas criadas no início deste estudo:

- É possível, através de um mecanismo de propagação de rótulos, tornar escalável a geração de estruturas de coalizão para grafos com milhares de vértices? A resposta a essa pergunta é sim. Especialmente, o algoritmo heurístico desenvolvido foi capaz de calcular estruturas de coalizão em grafos com 5 mil vértices e 6.500 arestas. Além disso, a adoção de propagação de rótulos como heurística proveu uma forma eficaz de acelerar o processo de cálculo. Em termos gerais, o algoritmo desenvolvido conseguiu bons resultados quanto ao valor final da estrutura (comparado com o valor máximo teórico possível), com garantias experimentais e *anytime*.

- O método de contágio pode ser caracterizado por um jogo onde os agentes querem maximizar seus ganhos? Sim, neste caso a função de valoração utilizada visa maximizar o ganho do agente aproximando-o de agentes com a mesma habilidade. Essa função de valoração também pode ser trocada, como mostrado nos experimentos, para proporcionar comportamentos distintos na formação das coalizões.

- Para evitar diminuir o valor da estrutura, ela poderia intervir no rótulo do agente? Sim, o comportamento do algoritmo pode deixar ou não a movimentação dos agentes ocorrer de acordo com o valor da estrutura.

Um dos produtos deste estudo foi a publicação na *Brazilian Conference on Intelligent Systems* (Conferência Brasileira em Sistemas Inteligentes) de um artigo sobre esta dissertação. O BRACIS'17 ocorreu em Uberlândia/MG de 2 a 5 de outubro de 2017.

Como sugestão de trabalho futuro pode-se citar o aprimoramento deste algoritmo para que evitasse mínimos locais (comportamento da Figura 15) com a adição do comportamento de Têmpera Simulada. Outra modificação possível é a retro análise, sempre que um novo ciclo de propagação é realizado, para evitar a quebra da coalizão, já que nesse momento um ciclo de propagação pode isolar agentes e tornar a coalizão um grafo desconexo. Também pode-se executar este algoritmo para grafos maiores de 5 mil vértices avaliando seu comportamento a fim de verificar seus limites ou provar seus limites e restrições.

REFERÊNCIAS

AZIZ, H.; KEIJZER, B. de. Complexity of coalition structure generation. In: **The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1**. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011. (AAMAS '11), p. 191–198. ISBN 0-9826571-5-3, 978-0-9826571-5-7. Disponível em: <http://dl.acm.org/citation.cfm?id=2030470.2030498>.

BACHRACH, Y. et al. Optimal coalition structure generation in cooperative graph games. In: **Proceedings of 27th Conference on Artificial Intelligence (AAAI)**. Washington - USA: [s.n.], 2013. p. 81–87.

BACHRACH, Y. et al. Cooperative weakest link games. In: **Proceedings of 13th Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)**. Paris - France: [s.n.], 2014. p. 589–596.

BISTAFFA, F. **CFSS implementation - edge sum with coordination cost**. 2016. Disponível em: https://github.com/filippobistaffa/CFSS. Acesso em 12 de abril de 2017.

BISTAFFA, F. et al. Algorithms for graph-constrained coalition formation in the real world. **ACM Trans. Intell. Syst. Technol.**, ACM, v. 8, n. 4, p. 60:1–60:24, fev. 2017.

BJöRKLUND, A.; HUSFELDT, T.; KOIVISTO, M. Set partitioning via inclusion-exclusion. **SIAM Journal on Computing**, v. 39, n. 2, p. 546–563, 2009. Disponível em: <http://dx.doi.org/10.1137/070683933>.

BRANDES, U. et al. On modularity clustering. **IEEE Transactions on Knowledge and Data Engineering**, v. 20, n. 2, p. 172–188, Feb 2008. ISSN 1041-4347.

CERQUIDES, J. et al. A tutorial on optimization for multi-agent systems. **The Computer Journal**, v. 57, n. 6, p. 799–824, 2014. Disponível em: <http://comjnl.oxfordjournals.org/content/57/6/799.abstract>.

CHEVALEYRE, Y. et al. A short introduction to computational social choice. In: **Proceedings of 33rd Conference on Current Trends in Theory and Practice of Computer Science**. Harrachov - Czech Republic: [s.n.], 2007. p. 51–69.

DEAN, T.; BODDY, M. An analysis of time-dependent planning. In: **Proceedings of 7th Conference on Artificial Intelligence (AAAI)**. Minnesota - USA: [s.n.], 1988. p. 49–54.

DENG, X.; PAPADIMITRIOU, C. H. On the complexity of cooperative solution concepts. **Mathematics of Operations Research**, INFORMS, v. 19, n. 2, p. 257–266, 1994. ISSN 0364765X, 15265471. Disponível em: <https://www.jstor.org/stable/3690220>.

EASLEY, D.; KLEINBERG, J. **Networks, Crowds, and Markets: Reasoning About a Highly Connected World**. NY, US: Cambridge University Press, 2010. ISBN 9781139490306.

GLOVER, F. W.; KOCHENBERGER, G. A. (Ed.). **Handbook of Metaheuristics**. NY, US: Springer US, 2003. ISBN 978-0-306-48056-0.

GOLUMBIC, M. C. **Algorithmic Graph Theory and Perfect Graphs, 2nd Edition**. Amsterdam, NL: North Holland, 2014.

HAFALIR, I. E. Efficiency in coalition games with externalities. **Games and Economic Behavior**, v. 61, n. 2, p. 242 – 258, 2007. ISSN 0899-8256. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0899825607000164>.

HOEFER, M.; VAZ, D.; WAGNER, L. Hedonic coalition formation in networks. In: **Proceedings of 29th Conference on Artificial Intelligence (AAAI)**. Austin - USA: [s.n.], 2015. p. 929–935. Disponível em: <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9617>.

HOFFMAN, K.; PADBERG, M. Set covering, packing and partitioning problems. In: FLOUDAS, C. A.; PARDALOS, P. M. (Ed.). **Encyclopedia of Optimization**. NY, US: Springer US, 2009. p. 3482–3486. ISBN 978-0-387-74758-3.

HU, X. et al. Role-based label propagation algorithm for community detection. **CoRR**, abs/1601.06307, 2016. Disponível em: <http://arxiv.org/abs/1601.06307>.

KEINäNEN, H. Simulated annealing for multi-agent coalition formation. In: **Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications**. Berlin, Heidelberg: Springer-Verlag, 2009. (KES-AMSTA '09), p. 30–39. ISBN 978-3-642-01664-6.

LESKOVEC, J.; HORVITZ, E. Planetary-scale views on a large instant-messaging network. In: **Proceedings of 17th International World Wide Web Conference (WWW2008)**. Pequim - China: [s.n.], 2008. p. 915–924.

LESKOVEC, J.; KREVL, A. **SNAP Datasets: Stanford Large Network Dataset Collection**. jun. 2014. Disponível em: http://snap.stanford.edu/data. Acesso em 03 de maio de 2017.

LIEMHETCHARAT, S.; VELOSO, M. Weighted synergy graphs for effective team formation with heterogeneous ad hoc agents. **Artificial Intelligence**, v. 208, p. 41 – 65, 2014. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370213001306>.

MESBAHI, M.; EGERSTEDT, M. **Graph Theoretic Methods in Multiagent Networks**. New Jersey, US: Princeton University Press, 2010.

MICHALAK, T. et al. A hybrid exact algorithm for complete set partitioning. **Artificial Intelligence**, v. 230, p. 14 – 50, 2016. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370215001423>.

MOON, J.; MOSER, L. On cliques in graphs. **Israel Journal of Mathematics**, Springer-Verlag, v. 3, n. 1, p. 23–28, 1965. ISSN 0021-2172. Disponível em: <http://dx.doi.org/10.1007/BF02760024>.

NEWMAN, M. **Network data**. 2013. Disponível em: http://www-personal.umich.edu/ mejn/netdata. Acesso em 3 de maio de 2017.

NUNES, A. A. **Restringindo o espaço de busca na geração de estruturas de coalizão utilizando grafos**. Dissertação (Dissertação de Mestrado) — Pós-Graduação em Computação Aplicada - Universidade Tecnológica Federal do Paraná, Curitiba - PR, Agosto 2015.

PELETEIRO, A.; BURGUILLO, J. C.; CHONG, S. Y. Exploring indirect reciprocity in complex networks using coalitions and rewiring. In: **Proceedings of 13th Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)**. Paris - France: [s.n.], 2014. p. 669–676.

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. **Physical Review E**, v. 76, n. 3, 2007.

RAHWAN, T. **ODP-IP implementation**. 2014. Disponível em: https://github.com/trahwan/ODP-IP. Acesso em 19 de dezembro de 2016.

RAHWAN, T.; JENNINGS, N. An improved dynamic programming algorithm for coalition structure generation. In: **Proceedings of 7th Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS)**. Estoril - Portugal: [s.n.], 2008. p. 1417–1420.

RAHWAN, T.; JENNINGS, N. R. An algorithm for distributing coalitional value calculations among cooperating agents. **Artificial Intelligence Journal**, v. 171, n. 8-9, p. 535–567, 2007. Disponível em: <http://eprints.soton.ac.uk/263720/>.

RAHWAN, T.; JENNINGS, N. R. Coalition structure generation: Dynamic programming meets anytime optimisation. In: **Proceedings of 23rd Conference on Artificial Intelligence (AAAI)**. Illinois - USA: [s.n.], 2008. p. 156–161.

RAHWAN, T. et al. Coalition structure generation in multi-agent systems with positive and negative externalities. In: **Proceedings of 21rd International Joint Conference on Artificial Intelligence (IJCAI)**. California, US: [s.n.], 2009. p. 257–263.

RAHWAN, T. et al. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. **Artificial Intelligence**, v. 186, p. 95 – 122, 2012. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370212000288>.

RAHWAN, T. et al. **An Exact Algorithm for Coalition Structure Generation and Complete Set Partitioning**. [S.l.], 2013. 50 p.

RAHWAN, T. et al. An anytime algorithm for optimal coalition structure generation. **J. Artif. Int. Res.**, AI Access Foundation, USA, v. 34, n. 1, p. 521–567, abr. 2009. ISSN 1076-9757. Disponível em: <https://www.jair.org/media/2695/live-2695-4374-jair.pdf>.

RAM, B. **Discrete Mathematics**. Cambridge, UK: Pearson, 2010. Disponível em: <http://www.amazon.com/gp/product/B00G4YDRQ0>.

RASMUSSEN, M. S.; LARSEN, J. **Optimisation-Based Solution Methods for Set Partitioning Models**. Tese (Doutorado), 2011.

RILEY, L. et al. Distributing coalition value calculations to coalition members. In: **Proceedings of 29th Conference on Artificial Intelligence (AAAI)**. Austin - USA: [s.n.], 2015. p. 2117–2123. Disponível em: <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9693>.

SANDHOLM, T. et al. Coalition structure generation with worst case guarantees. **Artificial Intelligence**, v. 111, n. 1–2, p. 209–238, 1999. ISSN 0004-3702. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0004370299000363>.

SEN, S.; SEN, I.; DUTTA, P. S. Searching for optimal coalition structures. In: **Proceedings of the Fourth International Conference on Multiagent Systems**. [S.l.]: IEEE, 2000. p. 286–292.

SHAPLEY, L. S. Cores of convex games. **International Journal of Game Theory**, Physica-Verlag, v. 1, n. 1, p. 11–26, 1971. ISSN 0020-7276. Disponível em: <http://dx.doi.org/10.1007/BF01753431>.

SHEHORY, O.; KRAUS, S. Methods for task allocation via agent coalition formation. **Artif. Intell.**, Elsevier Science Publishers Ltd., Essex, UK, v. 101, n. 1-2, p. 165–200, maio 1998. ISSN 0004-3702. Disponível em: <http://dx.doi.org/10.1016/S0004-3702(98)00045-9>.

TRUDEAU, R. J. **Introduction to Graph Theory 2nd Edition**. NY, US: Dover Publications, 2013.

UGANDER, J.; BACKSTROM, L. Balanced label propagation for partitioning massive graphs. In: **Proceedings of the Sixth ACM International Conference on Web Search and Data Mining**. New York, NY, USA: ACM, 2013. (WSDM '13), p. 507–516. ISBN 978-1-4503-1869-3. Disponível em: <http://doi.acm.org/10.1145/2433396.2433461>.

UGANDER, J. H. O. **Computational Perspectives on Large-Scale Social Networks**. Tese (Doutorado) — Cornell University, NY - USA, June 2014.

VOICE, T.; POLUKAROV, M.; JENNINGS, N. R. Coalition structure generation over graphs. **Journal of Artificial Intelligence Research (JAIR)**, v. 45, p. 165 – 195, 2012. ISSN 1076-9757.

WANG, X. F.; CHEN, G. Complex networks: small-world, scale-free and beyond. **IEEE, Circuits and Systems Magazine**, v. 3, n. 1, p. 6–20, 2003. ISSN 1531-636X.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of 'small-world' networks. **Nature**, v. 393, n. 6684, p. 440–442, Jun 1998. ISSN 0028-0836. Disponível em: <http://dx.doi.org/10.1038/30918>.

WAZLAWICK, R. **Metodologia de Pesquisa para Ciência da Computação**. RJ, BR: Elsevier, 2009. ISBN 9788535266436. Disponível em: <https://books.google.com.br/books?id=ZLbCKKWQ6OQC>.

YEH, D. Y. A dynamic programming approach to the complete set partitioning problem. **BIT Numerical Mathematics**, Kluwer Academic Publishers, v. 26, n. 4, p. 467–474, 1986. ISSN 0006-3835. Disponível em: <http://dx.doi.org/10.1007/BF01935053>.