UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

JUAN CARLOS TOLEDO BAPTISTA

RENDEZVOUS SIMÉTRICO EM GRAFOS

Rio de Janeiro
Novembro de 2017

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
INSTITUTO TÉRCIO PACITTI DE APLICAÇÕES E PESQUISAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

JUAN CARLOS TOLEDO BAPTISTA

RENDEZVOUS SIMÉTRICO EM GRAFOS

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Orientador: Vinícius Gusmão Pereira de Sá

Rio de Janeiro
Novembro de 2017

CBIB Baptista, Juan Carlos Toledo

Rendezvous simétrico em grafos / Juan Carlos Toledo Baptista. – Novembro de 2017.

67 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática, Rio de Janeiro, Novembro de 2017.

Orientador: Vinícius Gusmão Pereira de Sá.

.

1. Rendezvous de Robôs. 2. Algoritmos Randomizados. – Teses. I. Sá, Vinícius Gusmão Pereira de (Orient.). II. Universidade Federal do Rio de Janeiro, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Programa de Pós-Graduação em Informática. III. Título

CDD

JUAN CARLOS TOLEDO BAPTISTA

Rendezvous simétrico em grafos

Dissertação de Mestrado submetida ao Corpo Docente do Departamento de Ciência da Computação do Instituto de Matemática, e Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do título de Mestre em Informática.

Aprovado em: Rio de Janeiro, 22 de NOVEMBRO de 2017.

Prof. Dr. Vinícius Gusmão Pereira de Sá (Orientador)

Prof. Dr. Jayme Luiz Szwarcfiter

Prof. Dr. João Antônio Recio da Paixão

Profa. Dra. Loana Tito Nogueira

AGRADECIMENTOS

Agradeço, antes de tudo, a Deus - *em quem estão escondidos todos os tesouros da sabedoria e da ciência (cl2:3)* - pelo sustento e pelo refúgio em todos os momentos. Agradeço à minha esposa, Cristine, pelo carinho, companheirismo e suporte que me deram forças para seguir até o fim sem desistir. Agradeço também à minha família e aos amigos que me apoiaram e me incentivaram ao longo desse período. Ao meu orientador Vinícius Gusmão pelos conselhos, pela atenção e confiança, que foram fundamentais para a conclusão deste trabalho. Aos professores do Programa de Pós-Graduação em Informática/UFRJ por todo o conhecimento transmitido e pela solicitude. Agradeço também à CAPES pelo apoio financeiro.

RESUMO

Baptista, Juan Carlos Toledo. ***Rendezvous* simétrico em grafos**. Novembro de 2017. 57 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Novembro de 2017.

Dois agentes, partindo de pontos distintos e desconhecendo a localização um do outro, precisam se encontrar. Este cenário geral é conhecido como o *Problema do Rendezvous de Robôs*. Esta dissertação aborda algumas das principais variantes dentre aquelas em que os robôs se movem ao longo de certas famílias de grafos. Após serem revisitadas algumas soluções conhecidas, é oferecida uma solução randomizada simples e original para uma das variantes mais populares do problema, que é comumente referida como a variante dos *robôs de paraquedas*. Nesta variante, o espaço considerado é unidimensional (um *caminho* infinito), e as posições iniciais dos agentes recebem marcadores que podem ajudá-los a se orientar durante a busca. É também exigido que os robôs executem exatamente o *mesmo algoritmo*, isto é, trata-se de uma variante *simétrica*. A solução proposta apresenta melhor desempenho, em média, do que a melhor solução conhecida até então. Esta dissertação introduz, ainda, uma variante até então não considerada na literatura, a saber, aquela em que os agentes precisam retornar a suas posições iniciais, e para a qual o ganho em desempenho obtido pelo uso da randomização torna-se ainda mais acentuado.

Palavras-chave: Rendezvous de Robôs, Algoritmos Randomizados.

ABSTRACT

Baptista, Juan Carlos Toledo. ***Rendezvous* simétrico em grafos**. Novembro de 2017. 57 f. Dissertação (Mestrado em Informática) - PPGI, Instituto de Matemática, Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Novembro de 2017.

Two agents, starting from distinct points and unaware of the position of one another, must meet. This general setting is known as the *Robot Rendezvous Problem*. This M.Sc. thesis focuses on some of the main variations of the problem among those in which the agents move along given graph families. After revisiting existing solutions, we present a novel, randomized solution to one of its most popular variations, often referred to as the *parachuting robots*. In such a variation, the space is unidimensional (an infinite path graph), and the starting positions receive markers, which may play an important role in the search algorithm; it is also required that the two agents run exactly the same algortithm, i.e., the variation is *symmetric*. The proposed solution presents a better time complexity, on average, than the best solution known thus far. This text also introduces a variation, not yet considered in the literature, where the agents must return to their starting points after they have met, and where the performance improvement attained by the use of randomization is even more pronounced.

Keywords: Robots Rendezvous, Randomized Algorithms.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Dois agentes móveis, partindo de pontos distintos em algum ambiente, e desconhecendo a localização um do outro, precisam se encontrar. Este cenário geral é bem estudado na literatura e conhecido como o *Problema de Rendezvous*. Problemas deste tipo despertam interesse pela facilidade de compreensão de seu objetivo e pela intuitiva conexão com situações reais. Por exemplo: quando duas pessoas vão juntas a um shopping e em algum momento se perdem uma da outra, qual decisão elas deveriam tomar para se encontrar o mais rápido possível, supondo que não possam se comunicar? Talvez uma permanecer parada enquanto a outra procura por todo o shopping seja a melhor estratégia; ou possivelmente haja uma estratégia que favoreça o encontro mais rapidamente. Um dos primeiros cenários sugeridos na literatura como problema de rendezvous, conhecido como problema do astronauta, traz uma boa ilustração deste tipo de paradigma: dois astronautas pousam sobre um planeta desconhecido, sem nenhuma noção das posições um do outro nem de direção ou orientação em relação a algum eixo. A comunicação entre eles não é possível, porém sabem que o planeta é pequeno o suficiente para que toda sua superfície possa ser explorada várias vezes, se necessário. Possuindo velocidades idênticas, como os astronautas devem se movimentar de maneira que o tempo esperado de encontro entre eles seja minimizado? Dependendo do contexto, os agentes representam não só pessoas, mas vários tipos de entidades físicas (como robôs autônomos) ou virtuais (como agentes de software explorando uma rede, ou personagens de jogos explorando um cenário) (PELC (2012); ALPERN; GAL (2006); ALPERN (2002a)).

Diversos fatores precisam ser considerados na modelagem deste tipo de problema, fatores que impactam não só sua descrição, mas também a escolha da estratégia a ser utilizada pelos agentes. O principal deles é o ambiente percorrido

pelos agentes. Uma grande variedade de tipos de ambientes foi amplamente investigada nos trabalhos sobre rendezvous. Esses ambientes podem ser classificados em dois grandes grupos: terrenos físicos, normalmente modelados através de cenários geométricos, tais como linhas (finitas ou não) (BEVERIDGE; OZSOYELLER; ISLER (2011); ANDERSON; ESSEGAIER (1995); GAL (1999); BASTON; GAL (1998)) ou planos (BAEZA-YATES; CULBERSON; RAWLINS (1993); ANDERSON; FEKETE (2001); THOMAS; HULME (1997)); e redes de comunicação, onde diversas ferramentas vindas da Teoria dos Grafos auxiliam na criação e na otimização dos algoritmos (PELC (2012); KRANAKIS; KRIZANC; RAJSBAUM (2006); KOWALSKI; MALINOWSKI (2006)).

Uma outra possível classificação para os problemas de rendezvous é dada em relação à simetria das estratégias que devem ser adotadas pelos agentes. O problema é dito assimétrico quando é permitido aos agentes executarem algoritmos de busca independentes um do outro. Um exemplo deste caso é aquele em que um deles permanece parado enquanto o outro o busca pelo ambiente. Nas versões simétricas, os agentes devem executar a mesma estratégia de busca. Esta versão é mais apropriada para o contexto de robôs ou agentes de software que precisam ser programados de forma idêntica (BEVERIDGE; OZSOYELLER; ISLER (2011)).

Outro fator importante a ser considerado nos problemas de *rendezvous* é o determinismo da estratégia utilizada. Durante a execução do algoritmo de busca, os agentes devem tomar decisões sobre direção, velocidade ou quantidade de passos. Essas decisões podem ser tomadas todas de forma determinística ou pode-se lançar mão de alguma escolha aleatória durante o processo. Mais precisamente, em qualquer cenário determinístico, as posições iniciais dos agentes são escolhidas por um adversário que modela uma situação de pior caso, e cada decisão do agente é baseada unicamente em sua própria situação atual, podendo envolver algum tipo de identificador (se houver) e a parte do ambiente já explorada por ele. Já no cenário

randomizado, as posições iniciais dos agentes são escolhidas aleatoriamente, e suas decisões podem mudar aleatoriamente durante o percurso. O cálculo de tempo dessas estratégias também é diferente: enquanto no cenário determinístico, o tempo é dado pelo pior caso, no randomizado é considerado o valor esperado de cada caso. Em ambos os cenários, o interesse geralmente é reduzir o custo (ou custo esperado) do pior caso.

A sincronia entre os movimentos dos agentes também é uma característica relevante na elaboração de estratégias de rendezvous. Modelos síncronos normalmente assumem que as transições entre vértices são feitas com um passo dos agentes, e seus passos são coordenados por ciclos fixos de relógio (DESSMARK et al. (2006); KOWALSKI; MALINOWSKI (2006); TA-SHMA; ZWICK (2007)). Já em modelos assíncronos não existe o conceito de um ciclo de relógio que possa ser seguido pelos agentes, possibilitando movimentos assíncronos e até mesmo diferentes velocidades entre os agentes (CZYZOWICZ; PELC; LABOUREL (2012); DE MARCO et al. (2006)). Uma generalização natural para o problema de rendezvous é conhecida como *gathering*, onde, ao invés de apenas dois, estuda-se o cenário onde múltiplos agentes precisam se reunir num mesmo local de algum ambiente (ALPERN (2002b); ANDERSON; WEBER (1990); ISRAELI; JALFON (1990); SINHA; GHOSE (2006); DIMAROGONAS; KYRIAKOPOULOS (2007)).

Neste trabalho, focamos nas variantes simétricas do problema de *rendezvous* em grafos. Estudamos diversas modelagens encontradas na literatura e apresentamos seus principais resultados. Estudamos em especial um cenário unidimensional, que é modelado como um caminho infinito, em que os agentes deixam marcas em suas posições iniciais. Analisamos a solução determinística tradicional para este cenário e, a seguir, propomos uma solução randomizada cujo valor esperado para o tempo de encontro dos agentes é menor do que o da solução determinística na maioria dos casos. Mais formalmente, mostramos que, se a distância inicial d entre os agentes

(medida em passos) é escolhida aleatoriamente do conjunto $\{1, 2, \ldots, D\}$, então a probabilidade de o algoritmo randomizado ser mais rápido que o determinístico é maior que 0.5 para todo $D \in \mathbb{N}$. Por fim, consideramos uma extensão do problema em que os agentes, após terem se encontrado, devem retornar ao seu ponto de partida. Comprovamos que, para esta variação, a estratégia randomizada possui um desempenho superior para qualquer distância inicial $d \in \mathbb{N}$.

2 MODELOS E PARÂMETROS

Além das classificações gerais para problemas de rendezvous vistas no capítulo anterior, uma série de parâmetros dos modelos são relevantes para a elaboração e estratégias de resolução, e até mesmo para a viabilidade do encontro entre os agentes. Neste capítulo veremos características que devem estar bem definidas quando se trata da descrição de um modelo de rendezvous, algumas referentes ao grafo e outras aos próprios agentes.

2.1 Sobre o ambiente

Todos os modelos discutidos nesta dissertação se baseiam *grafos não direcionados*, capturando a ideia requerida na maioria das aplicações, de que os agentes podem se movimentar em ambos os sentidos das arestas. É assumido também que os *grafos são conexos*, condição necessária para a viabilidade do encontro (agentes com pontos de partida em diferentes componentes conexas obviamente nunca irão se encontrar).

Uma outra pressuposição que deve ser notada é a de que os *grafos são anônimos*, ou seja, vértices não possuem rótulos que possam ser identificados pelos agentes. A razão para isso é que, caso fosse possível aos agentes fazer distinção dos vértices por meio de rótulos, a estratégia de encontro mais eficiente seria marcar um ponto de encontro (por exemplo, o vértice de menor rótulo), reduzindo a tarefa a um problema de exploração em grafos. Em decorrência do anonimato do grafo, uma outra configuração faz-se necessária em alguns modelos: os vértices devem possuir *números de porta*, isto é, um vértice de grau d deve possuir os rótulos $0, 1, 2, ..., d-1$

correspondentes a cada uma de suas arestas incidentes. Se os agentes não pudessem identificar as "portas" de um vértices, não seriam capazes de explorar todos vizinhos de vértices de grau maior que 2, já que, ao visitar tais vértices, não saberiam identificar quais vizinhos já foram visitados anteriormente, pois suas "portas" seriam todas iguais. Entretanto, os números de porta de um vértice não possuem nenhuma consistência em relação a outros vértices no sentido de identificá-los na topologia do grafo, servem apenas identificar localmente em um vértice quais vizinhos já foram explorados. A figura 2.1 mostra um exemplos de um grafo com números de porta.

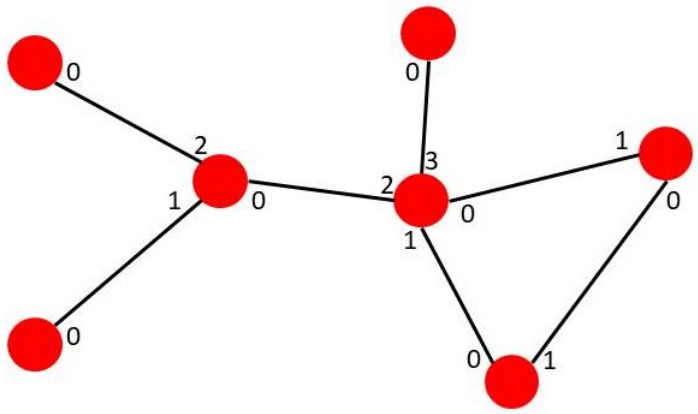


Figura 2.1: Exemplo: números de porta

2.2 Sobre os agentes

Passaremos, agora, às características referentes aos agentes. Diversos trabalhos atribuem rótulos únicos aos agentes, onde cada agente sabe apenas seu próprio rótulo, com a finalidade de parametrizar a estratégia utilizada para o encontro. Embora eles devam adotar o mesmo algoritmo, suas decisões são baseadas em seus rótulos. O objetivo dos rótulos é quebrar a simetria dos agentes em relação às suas posições iniciais. Em outros modelos, quando os agentes não podem ser diferenciados, estes são chamados de *agentes anônimos* e, nesse caso, as estratégias de busca

precisam utilizar outras técnicas para quebrar a simetria.

As informações conhecidas pelos agentes sobre o ambiente devem ser levadas em consideração no algoritmo de rendezvous. Informações como a topologia do grafo, o número de vértices e noção de orientação (no caso de grafo ciclo ou caminho) são exemplos de dados prévios que podem tornar a solução mais simples ou mais complexa de ser encontrada, ou mesmo otimizada.

Em geral, o objetivo dos problemas de rendezvous é que os agentes se encontrem o mais rápido possível, medida essa normalmente dada pela quantidade de passos dos agentes ou quantidade de ciclos de relógio. Porém, alguns autores se interessaram não apenas no menor de tempo necessário para o encontro, mas também na menor quantidade de memória necessária para que o encontro seja possível.

A caracterização do encontro dos agentes pode variar de acordo com o modelo desejado. No caso de ambientes virtuais, por exemplo, os agentes precisam estar no mesmo vértice ao mesmo tempo para que o encontro ocorra, ou seja, se dois agentes estiverem cruzando uma aresta ao mesmo tempo em sentidos opostos, não caracterizaria um encontro entre eles. Porém se pensarmos em situações físicas, como robôs sobre um terreno, faz sentido que os agentes possam se encontrar no meio de uma aresta. Alguns modelos encontrados na literatura assumem essa possibilidade.

Diversos trabalhos exploram o fato de que os agentes podem iniciar com um *delay* em relação um ao outro, isto é, eles podem iniciar a busca em tempos distintos. Quando isso ocorre, é assumido que os agentes são criados no momento de sua partida. Isso implica que na seguinte situação: suponha que o agente A tenha o vértice v como ponto de partida, mas ele inicia seus movimentos após o agente B já ter iniciado. Caso o agente B visite o vértice v antes do agente A ter sido criado, não é caracterizado o encontro. No cenário em que essa diferença no tempo inicial

não é permitida, o algoritmo pode ser elaborado assumindo que ambos os agentes já foram criados e que estão se movimentando sobre o ambiente na mesma quantidade de tempo.

Além das características vistas até aqui, veremos que uma série de modelos possuem restrições bastante específicas, o que torna o conjunto de cenários possíveis é bem extenso e até mesmo difícil de ser classificado por seus parâmetros. Nos Capítulos 3 e 4 serão apresentados os principais cenários de rendezvous encontrados na literatura, bem como alguns de seus algoritmos e resultados. No Capítulo 5, contribuímos com um algoritmo para o modelo específico de um grafo linear infinito onde os agentes deixam marcas em seus pontos de partida.

3 ALGORITMOS DETERMINÍSTICOS

Neste capítulo revisamos os principais modelos determinísticos de rendezvous em grafos e alguns de seus algoritmos encontrados na literatura. Veremos que diversas restrições específicas da topologia do grafo e da capacidade dos agentes dão origem a uma grande quantidade de modelos distintos e, por consequência, de estratégias para resolução.

O principal desafio dos algoritmos determinísticos para rendezvous em grafos é a quebra da simetria das posições dos agentes. Alguns tipos de grafos possuem uma simetria difícil de ser quebrada, como por exemplo, o grafo ciclo: todos os vértices possuem um vizinho no sentido horário e outro no sentido anti-horário. Se os agentes decidirem sempre pelo mesmo sentido, a distância entre eles nunca diminuirá, e o encontro não poderá ocorrer. Alguns recursos são utilizados para contornar este problema, como por exemplo o uso de rótulos únicos que diferenciam os agentes. Neste caso, os agentes executam a mesma estratégia parametrizada por seus rótulos. Como os rótulos são necessariamente distintos, isso garante que a simetria será quebrada em algum momento da execução do algoritmo. É importante ressaltar que os agentes conhecem apenas seu próprio rótulo, caso contrário, a estratégia poderia se reduzir a um dos agentes (o de menor rótulo, por exemplo) ficar parado enquanto o outro o procura.

Uma outra forma de quebrar a simetria é a possibilidade dos agentes deixarem marcas nos vértices. As marcas deixadas por um agente podem ser percebidas pelo outro, possibilitando assim que as decisões sobre seus movimentos sejam alteradas de acordo com a percepção ou não de marcas no vértice atualmente visitado. Por fim, uma terceira ferramenta usada para a quebra de simetria é a exploração

da assimetria do próprio grafo. Apesar dos vértices não possuirem identificadores, alguns vértices podem possuir características topológicas que os destacam no grafo. Caso essas características possam ser percebidas pelos agentes, estes podem utilizá-las como ponto de encontro, ou reduzir os possíveis pontos de encontro. Além disso, assimetrias das posições iniciais dos agentes em relação ao grafo como um todo também podem ajudar na construção de algoritmos determinísticos.

Nas próximas subseções veremos os principais modelos encontrados na literatura que utilizam os recursos de quebra de simetria citados anteriormente.

3.1 Agentes com Rótulos

Em DESSMARK et al. (2006), os autores apresentam algoritmos determinísticos para algumas classes de grafos buscando reduzir custo de *rendezvous*, dado pelo número de passos dos agente. O cenário modelado nesse trabalho considera que os agentes se movem em passos síncronos, ou seja, obedecem ciclos de relógio fixos, podendo ambos começar ao mesmo tempo ou haver uma diferença de tempo entre o primeiro passo de cada agente. Com o objetivo de quebrar a simetria, os agentes possuem rótulos, que são as representações binárias de dois inteiros distintos, e cada agente sabe apenas o seu próprio rótulo (se ambos os agentes soubessem ambos os rótulos, a solução se reduziria a um problema de busca simples em grafo, com o agente de menor rótulo esperando, enquanto o outro procura). Tais rótulos são utilizados como parâmetros para as estratégias, garantindo que os agentes tomem decisões que quebram a simetria em algum momento do algoritmo, permitindo assim o encontro.

3.1.1 *Rendevous* em árvores

Neste cenário, os agentes sabem apenas que o ambiente a ser percorrido é uma árvore, porém não têm nenhuma outra informação sobre sua topologia ou seu tamanho. A estratégia proposta lança mão do conceito de vértices centrais: aqueles que minimizam a maior distância para os outros vértices do grafo. Árvores possuem, no mínimo um e no máximo dois nós centrais. No segundo caso os nós centrais sempre estarão ligados por uma aresta (chamada de aresta central) (KNUTH (1997)). A estratégia consiste em explorar o grafo através de uma busca em profundidade procurando o vértice (ou aresta) central, onde ocorrerá o encontro. No caso da árvore possuir apenas um vértice central, o agente que encontrar este vértice primeiro permanece parado, esperando até que o outro o encontre. No caso da aresta central, após o encontro da mesma, o agente executa um procedimento que garante o encontro em um grafo K2 que será descrito a seguir.

O procedimento *Rótulo-Estendido* consiste em, a cada passo, decidir se o agente executa o comando *ande* (atravessa a aresta) ou *fique* (permanece no mesmo vértice) baseado numa extensão do rótulo L descrita a seguir: tome a representação binária de L e construa um rótulo L^* iniciando com os bits 10 e escrevendo duas vezes cada bit de L, então repita L^* indefinidamente, formando uma sequência binária infinita (veja um exemplo na figura 3.1). O i-ésimo bit desta sequência corresponderá ao passo i do agente, se o bit for 1, o agente anda, se o bit for 0, o agente fica. O procedimento é descrito mais formalmente pelo algoritmo 1.

Assumindo, sem perda de generalidade, que o agente B inicia os movimentos em um tempo τ após o agente A, a prova de complexidade do procedimento *Rótulo-Estendido* é dividida em 4 possibilidades, onde a contagem de passos é dada pelos passos do agente B:

Caso 1: τ é ímpar – Neste caso, ou o agente A *fica* no segundo passo, ou o encontro ocorre em um dos dois primeiros passos. Caso o agente A *fique* no segundo passo, ele ficará também no terceiro (já que o segundo passo seria um de índice ímpar para o agente A). Ocorre que, o agente B necessariamente *anda* no terceiro passo, que corresponde ao bit mais significativo de L_B, que é 1. Logo, o encontro ocorre, no máximo, no passo 3.

Caso 2: τ é par e não é divisível por $2\lfloor \log L_A \rfloor + 4$ – Nesse caso, os movimentos do agente A serão decididos pelo mesmo bit de L_A. Portanto, as ações dos dois agentes serão diferentes em um dos dois primeiros passos, provocando o encontro.

Caso 3: τ é par e divisível por $2\lfloor \log L_A \rfloor + 4$, e $\lfloor \log L_A \rfloor = \lfloor \log L_B \rfloor$ – Nesse caso, pelo menos um bit é diferente em ambos os rótulos. Seja b a posição deste bit. No passo $2b + 1$ os agentes tomarão decisões diferentes e se encontrarão. Portanto, o encontro é alcançado no passo $2\lfloor \log L_A \rfloor + 3$.

Caso 4: τ é par e divisível por $2\lfloor \log L_A \rfloor + 4$, e $\lfloor \log L_A \rfloor \neq \lfloor \log L_B \rfloor$ – Neste caso as decisões do agente de menor rótulo serão diferentes nos passos $2\lfloor \log l \rfloor + 5$ e $2\lfloor \log l \rfloor + 6$, enquanto o outro agente tomará a mesma decisão nesses passos. Logo, o encontro acontecerá no passo $2\lfloor \log l \rfloor + 6$, onde l é o tamanho do agente de menor rótulo.

Dessa forma, fica provado que o procedimento *Rótulo-Estendido* alcança o encontro em tempo $O(2\lfloor \log l \rfloor + 6)$. A estratégia geral para rendezvous em árvores proposto por DESSMARK et al. (2006) é descrita pelo algoritmo 2. Como a exploração de uma árvore de n vértices pode ser feita através de uma busca em profundidade em tempo $O(n)$, ficou estabelecido que o algoritmo geral para rendezvous em árvores de n vértices garante o encontro dos agentes em $O(n + \log l)$ passos. Além disso, também foi mostrado que existem árvores em que esta complexidade

é a melhor possível para qualquer algoritmo de rendezvous, mesmo considerando o cenário em que os agentes iniciam a busca no mesmo instante de tempo.

Algoritmo 1: Procedimento Rótudo-Estendido para Rendezvous em um K2

– No passo 1, *ande*.
– No passo 2, *fique*.
– Nos passos $3 \leq i \leq 2\lfloor \log L \rfloor + 4$, *ande* se o bit $\lceil (i-2)/2 \rceil$ de L é igual 1, caso contrário, *fique*.
– Nos passos $i > 2\lfloor \log L \rfloor + 4$, comporte-se como no passo

$$1 + (i - 1 \mod 2\lfloor \log L \rfloor + 4)$$

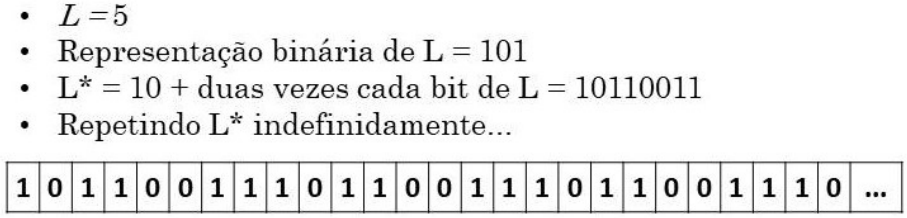


Figura 3.1: Exemplo do procedimento rótulo-extendido

Algoritmo 2: Algoritmo Determinístico para Rendezvous em Árvores

início
  Explore a árvore procurando um vértice ou aresta central
  se *a árvore possuir um único vértice central v* **então**
    vá para v e pare
  senão
    vá para a aresta central
    execute o procedimento *Rótulo-Estendido*
  fim se
fim

3.1.2 *Rendezvous* em ciclos

Neste cenário, os agentes conhecem que a estrutura do grafo é um ciclo, porém desconhecem o seu tamanho. Foram apresentados algoritmos diferentes para

os casos em que os agentes iniciam a busca ao mesmo tempo e em tempos arbitrários.

3.1.2.1 Partida Simultânea

Na situação em que os agentes partem do ponto inicial no mesmo instante de tempo, foi proposto um algoritmo ótimo em relação ao número de passos. Com o objetivo de simplificar a apresentação, foram mostrados dois algoritmos que funcionam apenas sob determinadas restrições e posteriormente um algoritmo geral, formado pela união dos dois primeiros, que considera todos os casos.

O primeiro algoritmo funciona sob a condição dos rótulos dos agentes serem semelhantes, mais precisamente, $\lfloor \log \log L_1 \rfloor = \lfloor \log \log L_2 \rfloor$. Seja o rótulo L_i^* consistindo do rótulo L_i e completado com zeros à esquerda para que o tamanho do rótulo seja $2^{\lfloor \log \log L_i \rfloor + 1}$. Por exemplo, se $L = 15$ (1111), então $L^* = 1111$, enquanto se $L = 16$ (10000), então $L^* = 00010000$. O agente decide se mover ou permanecer parado baseado nos bits de L^*. O algoritmo possui estágios numerados, onde cada estágio consiste em $2^{\lfloor \log \log L_i \rfloor + 1}$ fases, correspondentes a cada bit de L^*.

Algoritmo 3: Rendezvous Determinístico em Grafo Ciclo - Rótulos Semelhantes

início
  para $s \leftarrow 1, 2, 3...$ **faça**
    para *cada bit* b *de* L^* **faça**
      se $b = 1$ **então**
        - ande 2^{s-1} passos a partir do vértice inicial em uma direção arbitrária.
        - ande 2^s na direção oposta
        - volte para o vértice inicial
      senão
        permaneça parado por 2^{s+1} unidades de tempo
      fim se
    fim para
  fim para
fim

Note que, no estágio s, o agente explora todos os vértices a uma distância de até 2^{s-1} do seu vértice inicial. O encontro ocorre no primeiro estágio em que 2^{s-1} for maior que a distância inicial e quando a fase b corresponder a bits diferentes em L_1^* e L_2^*. Os autores mostram que o algoritmo acima provê o encontro dos agentes em $O(D \log l)$ passos, onde D é a distância inicial entre os agentes.

O segundo algoritmo funciona sob restrição complementar a do primeiro, ou seja, $\lfloor \log \log L_1 \rfloor \neq \lfloor \log \log L_2 \rfloor$. Cada estágio s do algoritmo possui s fases. Seja $b_i = 2 + \lfloor \log \log L_i \rfloor$, então o agente i executa os seguintes passos:

Algoritmo 4: Rendezvous Determinístico em Grafo Ciclo - Rótulos Diferentes

```
início
    b_i ← 2 + ⌊log log L_i⌋
    para s ← 1, 2, 3... faça
        para p = 1 até s faça
            se s < b_i ou p ≠ s − b_i + 1 então
                permaneça parado
            senão
                - ande 2^{p−1} passos a partir do vértice inicial em uma direção
                  arbitrária.
                - ande 2^p na direção oposta
                - volte para o vértice inicial
            fim se
        fim para
    fim para
fim
```

No estágio $s \geq b_i$, o agente i visita todos os vértices a uma distância máxima 2^{s-b_i}. Portando, o encontro ocorrerá no estágio em que s é o menor inteiro tal que $2^{s-b_i} \geq D$ e i é o índice do agente que possui o menor rótulo. Foi mostrado que este algoritmo também alcança o encontro em $O(D \log l)$ passos.

A estratégia para resolver o caso geral (para rótulos de quaisquer tamanhos) é uma combinação dos dois algoritmos apresentados anteriormente. A ideia é atribuir fatias de tempo que alternem as execuções dos algoritmos. Em cada início de fase dos algoritmos, o agente se encontra em seu vértice inicial. A próxima fase do algoritmo da vez é executada caso haja tempo para ela na fatia de tempo atual. Caso contrário, o agente deve esperar no vértice inicial o início da próxima fatia (onde será executado o outro algoritmo). Só na fatia seguinte, que o algoritmo voltará a ser executado da fase onde parou.

A fatia de tempo t consistirá de 2^{t+1} passos. É importante notar que as fases que serão executadas em uma fatia de tempo nunca terão mais passos que o número

total de passos dessa fatia. Como uma fase do algoritmo terá, no máximo, o dobro do número de passos da fase anterior, temos que uma fração constante de cada fatia de tempo é de fato utilizada pelo algoritmo geral. Temos também que exatamente um dos algoritmos terá sua condição cumprida pelos rótulos, e este alcançará o encontro dos agentes em $O(D \log n)$, enquanto o outro executará não mais que duas vezes mais passos no total.

3.1.2.2 Partida Arbitrária

Os algoritmos apresentados na subseção anterior exploram a sincronia entre as fases executadas pelos agentes, portanto não podem ser utilizados no cenário em que estes podem não iniciar a busca simultaneamente, mas com um delay τ entre os tempos iniciais. DESSMARK et al. (2006) propuseram, então, um algoritmo que explora somente a diferença entre os rótulos para alcançar a quebra da simetria. A ideia é que os agentes alternem entre ciclos crescentes de andar e parar, de forma que o tamanho desses ciclos do agente de maior rótulo supere o de menor. Para o agente com rótulo L, o algoritmo é como se segue:

Algoritmo 5: Rendezvous Determinístico em Grafo Ciclo - Partida Arbitrária - Dessmark

início
  para $k = 1, 2, ...$ **faça**
    - ande kL passos em uma direção qualquer;
    - permaneça parado por kL passos
  fim para
fim

Os autores demonstram que esta estratégia provê o encontro entre os agentes em $O(l\tau + ln^2)$ passos.

A seguir será descrito um algoritmo apresentado por KOWALSKI; MALI-

NOWSKI (2006), que garante o encontro entre os agentes em $O(n \log l)$ passos, um desempenho superior ao apresentado por DESSMARK et al. (2006).

Primeiro, é definida uma função $f(L, r)$ para um rótulo L e um inteiro r, como uma sequência binária obtida substituindo cada 0 na representação binária de L por por $0^{4^r}1^{4^r}$ e cada 1 por $1^{4^r}0^{4^r}$. Cada agente escolhe arbitrariamente uma das duas direções para iniciar a busca, então executa o algoritmo 6.

Algoritmo 6: Rendezvous Determinístico em Grafo Ciclo - Partida Arbitrária - Kowalski

```
início
    cont ← 0
    d ← 1
    para estagio ← 1, 2, ... faça
        T ← f(L, estagio)
        para i ← 1, ..., |T| faça
            se T[i] = 1 então
                ande para o próximo vértice mantendo a direção atual
            senão
                permaneça parado por um passo
            fim se
            se cont = d então
                mude a direção atual
                d ← 2d
            fim se
        fim para
    fim para
fim
```

3.1.3 *Rendezvous* em grafos arbitrários

3.1.3.1 Partida Simultânea

Para este cenário, foi proposto uma generalização do algoritmo para ciclos, que lança mão do mesmo padrão de atividade e passividade dos agentes, intercalando entre os algoritmos para rótulos semelhantes e diferentes. A única diferença neste caso é que, ao invés de buscas lineares crescentes para alcançar os vértices mais próximos, os agentes farão uma buscas em largura pelo grafo e voltarão ao ponto inicial, a cada ciclo. Com isso, o encontro ocorrerá quando a busca alcançar os D vértices mais próximos, demandando um tempo de $O(D\Delta^D)$ passos. Assim os autores mostram que essa estratégia provê o encontro em $O(D\Delta^D \log l)$ passos no total.

3.1.3.2 Partida Arbitrária

Para a situação em que os agentes partem em tempos distintos, a uma distância de tempo τ um do outro, DESSMARK et al. (2006) apresentaram uma estratégia onde são usados objetos combinatórios cuja existência é provada pelo método probabilístico (embora o algoritmo seja determinístico). Cada agente encontra, independentemente um do outro, o mesmo objeto combinatório com determinadas propriedades que são usadas no algoritmo para alcançar o encontro. Esse cálculo pode ser realizado usando uma busca exaustiva, porém o tempo computacional dos agentes não contribuem com o custo total do algoritmo, que é calculado pelos movimentos dos agentes. Os autores argumentam que o cálculo para encontrar estes objetos seriam feitos numa fase de pré-processamento, anterior à busca propriamente dita.

Considerando apenas os passos dos agentes, este algoritmo alcança o encontro em tempo polinomial em n, τ e $\log l$, mais precisamente $O(n^5\sqrt{\tau \log l} \log n + n^{10} \log^2 n \log l)$. Além disso, os autores também mostram limites inferiores que dependem de n e de l para diferentes tipos de grafos, porém nenhum desses limites envolviam o delay τ entre as partidas dos agentes. Então, o trabalho é concluído com a seguinte questão: exite algum algoritmo determinístico para rendezvous em grafos arbitrários cuja complexidade seja independente de τ?

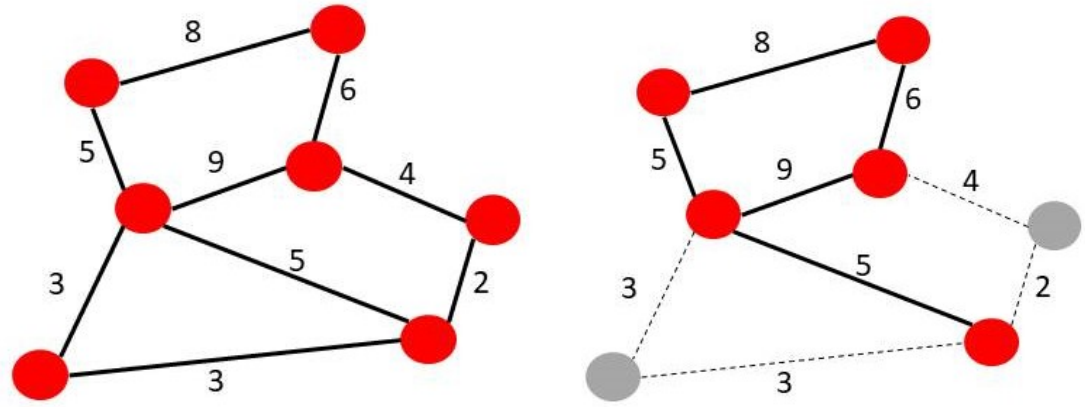
Tal algoritmo foi apresentado por KOWALSKI; MALINOWSKI (2006), utilizando a mesma ideia do algoritmo de DESSMARK et al. (2006) em relação ao cálculo de objetos combinatórios que são encontrados pelos agentes. Sua complexidade é de $O(\log^3 l + n^{15} \log^{12} n)$, e portanto, independente de τ.

3.2 Agentes Anônimos

Nesta seção veremos alguns cenários de rendezvous onde os agentes são anônimos, ou seja, não existem rótulos que garantam uma distinção entre cada agente e, portanto, os algoritmos determinísticos não recebem parâmetros distintos como entrada. Neste caso, os algoritmos devem explorar outros fatores que garantam a quebra da simetria dos movimentos, como a assimetria das posições iniciais no grafo ou a capacidade dos agentes de fazerem marcas nos vértices.

3.2.1 Grafos restritos

Grande parte dos modelos estudados em problemas de rendezvous em grafos consideram que os agentes têm acesso ao mesmo conjunto de vértices e arestas. FARRUGIA et al. (2015) estudaram cenários interessantes onde agentes anônimos



(a) Grafo original (b) Subgrafo alcançável por agente de peso 5

Figura 3.2: Modelo com limite de peso nas arestas

possuem propriedades que, combinadas com as características topológicas do grafo, impedem que estes agentes acessem determinadas arestas ou vértices. Seja $G = (V, E)$ o ambiente em questão, são definidos $G_A = (V_A, E_A)$ e $G_B = (V_B, E_B)$, onde $V_A, V_B \subseteq V$ e $E_A, E_B \subseteq E$, os *subgrafos alcançáveis* pelos agentes A e B, respectivamente. É fácil notar que neste tipo cenário o encontro nem sempre é viável, já que o conjunto de vértices acessíveis por ambos os agentes pode ser vazio, por exemplo.

Num primeiro cenário, restrições nas arestas são modeladas como limites de peso que cada aresta suporta, ou seja, um agente só pode atravessar arestas que suportam seu peso. Isso produz uma ordenação natural das arestas em relação ao seu peso máximo, logo $E_A \subseteq E_B$ ou vice-versa. Por definição do modelo, os vértices também possuem uma ordenação. A figura 3.2a) mostra um exemplo de um grafo G e a figura 3.2b) realça um subgrafo alcançável de G para um agente com peso 5. Os autores apresentam um algoritmo ótimo que garante o encontro em $O(k_A + k_B)$ passos (caso o encontro seja viável), onde k_X, $X \in A, B$, é o número de vértices de G_X.

Já num cenário seguinte, é assumido apenas que $V_A \subseteq V_B$ (sem perda de generalidade), isto é, as restrições são impostas apenas nos vértices, porém ainda levando em consideração a ordenação dos mesmos. A relação entre as arestas nos conjuntos E_A e E_B não é especificada. Para este modelo é mostrado um algoritmo que, caso o encontro entre os agentes seja viável, é alcançado em $O((k_A + K_B)\log(k_A + k_B))$ passos.

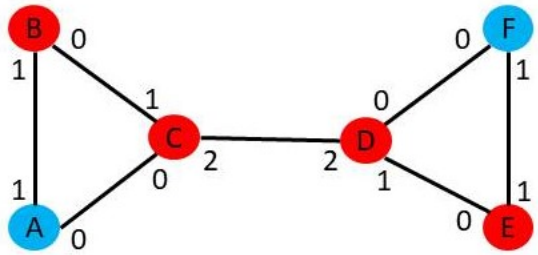
Por fim, é estudado o caso em que os subgrafos alcançáveis dos agentes não possuem nenhuma relação entre si, chamado de rendezvous cego. Uma única restrição trivial é imposta: $V_A \cap V_B \neq 0$, ou seja, ao menos um vértice deve ser acessível por ambos os agentes. Os autores mostram que neste modelo não é possível garantir o encontro dos agentes. Para resolver este problema, foram adicionadas novas informações ao modelo de rendezvous cego através de rótulos explícitos nos vértices do grafo, de forma que um vértice possui o mesmo rótulo identificado pelos agentes em ambos os grafos alcançáveis. Assim, foi possível estabelecer um algoritmo que garante o encontro em tempo $\min\{O((k_A + k_B)^3 \log\log n), O((k_A + k_B)^2 \log n)\}$.

3.2.2 Otimizando memória

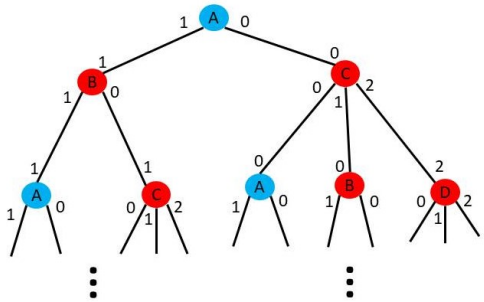
Problemas de rendezvous possuem forte relação com problemas de exploração de grafos. Por isso, alguns trabalhos abordaram o problema de rendezvous de robôs sob uma perspectiva normalmente utilizada em problemas de exploração: ao invés da redução do número de passos dos robôs, o objetivo é reduzir a quantidade de memória utilizada pelos robôs para armazenarem informações sobre o grafo. Os trabalhos considerados até aqui assumem que os agentes possuem uma quantidade infinita de memória, já os vistos neste tópico admitem limitações quanto ao número de bits necessários para guardar as informações.

Em CZYZOWICZ; KOSOWSKI; PELC (2012) são construídos agentes idênticos equipados com $O(\log n)$ bits de memória capazes se encontrar em grafos de até n vértices, podendo iniciar a busca em tempos diferentes um do outro. Os autores provam que não é possível alcançar o encontro caso os pontos de partida dos agentes sejam simétricos. A ideia de simetria é formalizada usando o conceito de *visão* de um vértice, introduzido por YAMASHITA; KAMEDA (1996). A visão de um vértice v é uma árvore infinita $\mathcal{V}(v)$ que segue a seguinte definição recursiva: a raiz de $\mathcal{V}(v)$ é x_0 e corresponde a v. Para cada vizinho v_i de v, existe um vizinho correspondente x_i para x_0 tal que o número de porta em v para a aresta v, v_i é o mesmo número de porta em x_0 para a aresta x_0, x_i, e o número de porta em v_i para a aresta v, v_i é o mesmo número de porta em x_i para a aresta x_0, x_i. Seguindo esta definição, os vértices u e v são ditos simétricos se $\mathcal{V}(u) = \mathcal{V}(v)$. Na figura 3.3a) vemos um exemplo onde os vértices A e F são simétricos, possuindo ambos a mesma visão (quanto aos números de porta) representados nas figuras 3.3b) e 3.3c), respectivamente. É importante ressaltar que os rótulos foram inseridos apenas para facilitar a compreensão do leitor, não sendo percebidos pelos agentes.

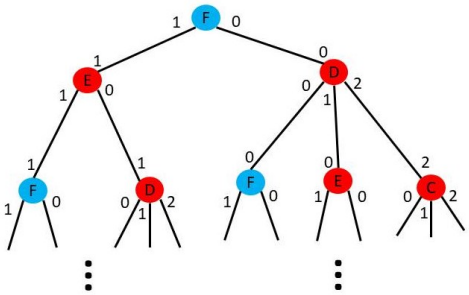
Já em FRAIGNIAUD; PELC (2013), os autores possuem o mesmo objetivo de otimizar a quantidade de memória, porém com o foco na exploração de árvores. A principal contribuição deste trabalho é o impacto que o delay entre os tempos iniciais dos agentes produz na quantidade de memória necessária para que o encontro seja possível. É provado que, para um delay arbitrário, o limite inferior de bits de memória é de $\Omega(\log n)$, mesmo em um grafo caminho de tamanho n. Em contraste, para partidas simultâneas, o tamanho de memória que viabiliza o encontro depende de n e do número de folhas f. Os autores mostram agentes com $O(\log f + \log \log n)$ bits memória capazes de se encontrar em árvores com n vértices e f folhas. Para a classe de árvores com $O(\log n)$ folhas, é revelado uma diferença exponencial de memória mínima necessária entre os cenários com delay arbitrário e partidas simultâneas.



(a) Vértices simétricos



(b) Visão do vértice A



(c) Visão do vértice F

Figura 3.3: Exemplo de simetria de vértices

CZYZOWICZ; KOSOWSKI; PELC (2014) estudaram a relação entre a quantidade de bits de memória utilizados pelos agentes e o tempo de encontro entre eles. É mostrado que, para agentes com k bits de memória, é possível alcançar o encontro em tempo $O(n + n^2/k)$ em árvores com n vértices, mesmo com partidas em tempo arbitrariamente distintos. A relação tempo versus memória em árvores também é estudada por BABA et al. (2010), porém em um modelo envolvendo múltiplos agentes.

3.2.2.1 Marcações nos vértices

Conforme mencionamos, uma das técnicas que permitem a quebra de simetria dos movimentos dos agentes em rendezvous determinísticos é a possibilidade do agente deixar marcas nos vértices que possam ser percebidas pelo outro agente. Essas marcações, quando percebidas por um agente, trazem informações adicionais sobre a posição do outro agente, possibilitando uma mudança em seu percurso que reduza o tempo da busca.

KRANAKIS et al. (2003) estudaram o problema de rendezvous em um grafo ciclo de n vértices, onde os agentes iniciam a uma distância d um do outro e são capazes de fazer uma única marca no grafo (as marcas dos agentes são idênticas). Os agentes devem executar o mesmo algoritmo determinístico partindo ao mesmo tempo de seus vértices iniciais. Diferentemente dos modelos vistos até o momento, este cenário permite que o encontro ocorra tanto nos vértices quanto nas arestas, quando os agentes estão simultaneamente atravessando a mesma aresta em direções opostas.

Neste trabalho os autores exploram diversas variações desse modelo, modificando os parâmetros em relação ao conhecimento dos agentes sobre n, d e a orientação do ciclo. São estabelecidos limites superiores e inferiores de tempo para cada variação, além de estabelecer a quantidade de mínima memória que os agentes devem possuir em cada uma delas. Esses resultados são resumidos na tabela 3.1, devida a KRANAKIS et al. (2003). Quando os limites inferior e superior são os mesmos, a complexidade é exibida apenas uma vez.

Os autores começam mostrando que, mesmo com o uso das marcações, não é possível quebrar a simetria quando $d = n/2$, por isso é assumido em todos os cenários que os agentes sabem que $d < n/2$, porém nem sempre sabem o valor exato

Conhecido?			Limite	Limite	Memória
n	d	orientação	Superior	Inferior	Necessária
sim	sim	sim	$3d/2$	$3d/2$	$O(\log d)$
sim	sim	não	$3d/2$	$5d/2$	$O(\log d)$
sim	não	sim	$(n+d)/2$	$(n+d)/2$	$O(\log n)$
sim	não	não	$(n+d)/2$	$(n+d)/2$	$O(\log n)$
não	sim	sim	$3d/2$	$3d/2$	$O(\log d)$
não	sim	não	$3d/2$	$5d/2$	$O(\log d)$
não	não	sim	$n+d/2$	$n+d/2$	$O(\log n)$
não	não	não	$n+d/2$	$n+d/2$	$O(\log n)$

Tabela 3.1: Limites para algoritmos de rendezvous em grafos ciclo com marcadores

de d.

No caso em que conhecem d, os agente precisam se mover d passos em alguma direção para conhecerem a posição das marcações um do outro (um deles encontrará a marcação, enquanto o outro saberá que a marcação está na direção oposta a uma distância $2d$). Então, pelo menos $d/2$ passos além dos primeiros d são necessários para estabelecer o encontro, totalizando um limite inferior de $3d/2$ passos.

Quando d é desconhecido porém n é conhecido, um dos agentes encontrará a marcação em d passos, enquanto o outro só saberá que a marcação está na direção contrária após $n/2$ passos. Somando aos $d/2$ necessários para se moverem em direção um ao outro, é estabelecido um limite inferior de $\frac{n+d}{2}$ passos. Por fim, quando d e n são desconhecidos, quando um agente encontra uma marcação, não é possível saber se ele se moveu d ou $n-d$ passos até que ele retorne ao seu vértice inicial, levando n unidades de tempo. Logo, quando d e n são desconhecidos, são necessários no mínimo $n + d/2$ passos para o encontro.

Para calcular os limites superiores, os autores apresentam soluções para os diversos cenários considerados. Como pode ser notado pela tabela 3.1, quase todos

os limites superiores alcançam os inferiores, exceto quando d é conhecido, mas a orientação do ciclo é desconhecida. O algoritmo 7 assume que d e a orientação do ciclo são conhecidos (n pode ser conhecido ou não), garantindo o encontro em, no máximo, $3d/2$ passos.

Algoritmo 7: Rendezvous em Grafo Ciclo com Marcadores para d e orientação Conhecidos

```
início
    Marque o vértice inicial
    Ande d passos no sentido horário
    se uma marcação for encontrada então
        continue andando no sentido horário
    senão
        ande no sentido anti-horário
    fim se
fim
```

O algoritmo 7 pode ser modificado para a situação em que a orientação do ciclo é desconhecida. Basta substituir a escolha do sentido inicial por uma escolha arbitrária, invertendo o sentido do agente caso não encontre a marcação nos d primeiros passos. A complexidade, no entanto, aumenta devido à possibilidade dos agentes escolherem direções iniciais opostas, alcançando o encontro em $5d/2$ passos.

O algoritmo 8 funciona para os agentes que desconhecem a distância inicial d, mas sabem o tamanho do ciclo n (podendo ou não saber a orientação). Os autores mostram que, no pior caso, o encontro ocorre em $(n + d)/2$ passos.

No cenário mais difícil, quando d, n e a orientação são desconhecidos dos agentes, o algoritmo 9 garante o encontro em $n + d/2$ passos. Este algoritmo requer $\Theta(\log n)$ bits de memória, pois os agentes precisam contar os vértices para descobrir os valores de d e n. Os autores propuseram, então, um outro algoritmo que requer $\Theta(\log \log n)$ bits de memória, porém a complexidade de tempo é um pouco maior: $O(n \log n / \log \log n)$.

Algoritmo 8: Rendezvous em Grafo Ciclo com Marcadores para d Desconhecido e n Conhecido

início
  Marque o vértice inicial
  Escolha arbitrariamente uma direção e ande $n/2$ passos
  se *uma marcação for encontrada* **então**
    continue andando na mesma direção
  senão
    ande no sentido oposto
  fim se
fim

Algoritmo 9: Rendezvous em Grafo Ciclo com Marcadores para d e n Desconhecidos e orientação Conhecida

início
  Escolha uma direçãao e conte o núumero de passos δ_1 até a primeira marcação
  Continue na mesma direção e conte o número de passos δ_2 até retornar ao ponto inicial.
  se $\delta_1 < \delta_2$ **então**
    continue andando na mesma direção
  senão
    ande no sentido oposto
  fim se
fim

FLOCCHINI et al. (2004) também estudaram o modelo de rendezvous em anéis com marcadores, porém como uma generalização para $k \geq 2$ agentes. Neste trabalho os autores exploram relação entre a complexidade de tempo de encontro e a quantidade de memória utilizada. São apresentados três algoritmos principais: o primeiro com tempo $O(n)$ utilizando $O(k \log n)$ bits de memória; o segundo com tempo $O(kn)$ utilizando $O(\log n)$ bits; e o terceiro com tempo $O(n \log n / \log \log n)$ utilizando $O(k \log \log n)$ bits de memória. Um algoritmo ótimo quanto ao uso de memória em grafos ciclo é mostrado em GĄSIENIEC et al. (2006), considerando o modelo em que os agentes podem se mover em apenas um sentido do ciclo. A solução apresentada utiliza $O(\log k + \log \log n)$ bits de memória. Ainda no contexto de marcações em vértices, KRANAKIS; KRIZANC; MARKOU (2006) estudaram um modelo em que as marcações feitas no vértices não são fixas, ou seja, uma marcação feita em um vértice pode ser recolhida por outro agente e recolocada em outro vértice. Um torus cuja orientação é conhecida pelos agentes é tratado neste modelo. CZYZOWICZ et al. (2008) exploraram o impacto da quantidade de marcas que os agentes são capazes de deixar em um grafo ciclo no tempo total para o encontro.

Notamos que, para grafos de tamanho limitado, a quantidade de memória necessária em todas as soluções propostas é bem pequena, fazendo com que o alvo de otimização mais interessante seja, de fato, o tempo gasto no percurso dos agentes.

4 ALGORITMOS RANDOMIZADOS

4.1 Rendezvous em grafos completos

Um dos problemas de rendezvous em grafos mais antigos é conhecido como o *problema do telefone*, e é descrito a seguir: duas salas separadas possuem n telefones cada uma; estes estão espalhados aleatoriamente no interior da sala, cada um conectado por um fio a um telefone da outra sala, de maneira que formam n pares. Em períodos de tempo discretos, dois agentes (um em cada sala) pegam um telefone e dizem "alô". Qual algoritmo deve ser adotado pelos agentes a fim de que o tempo necessário para estabelecerem uma conexão seja minimizado? Este problema é equivalente a um cenário em que agentes buscam se encontrar em um grafo completo K_n, onde cada vértice corresponde a uma par de telefones conectado. Em uma primeira solução trivial, onde, em cada passo, os agentes escolhem aleatoriamente um dos n vértices para explorar (incluindo permanecer no vértice atual), é fácil verificar que o tempo esperado para que ambos os agentes escolham o mesmo vértice ao mesmo tempo é igual a n.

Os primeiros resultados para este problema foram dados por ANDERSON; WEBER (1990), onde foi proposto a estratégia descrita pelo algoritmo 10. Este algoritmo recebe como entrada um parâmetro θ que será utilizado como probabilidade para as escolhas aleatórias dos agentes. Podemos perceber que o encontro pode ocorrer em duas situações: quando um agente faz a escolha de permanecer parado num vértice, enquanto o outro visita todo o grafo; ou, quando ambos escolhem visitar todo o grafo e existe um vértice coincidente na ordem das permutações escolhidas. O restante do trabalho consiste em provar que, para cada valor de $n \geq 2$, existe uma escolha ótima de θ que minimiza o tempo esperado do algoritmo. Tomando $n \to \infty$,

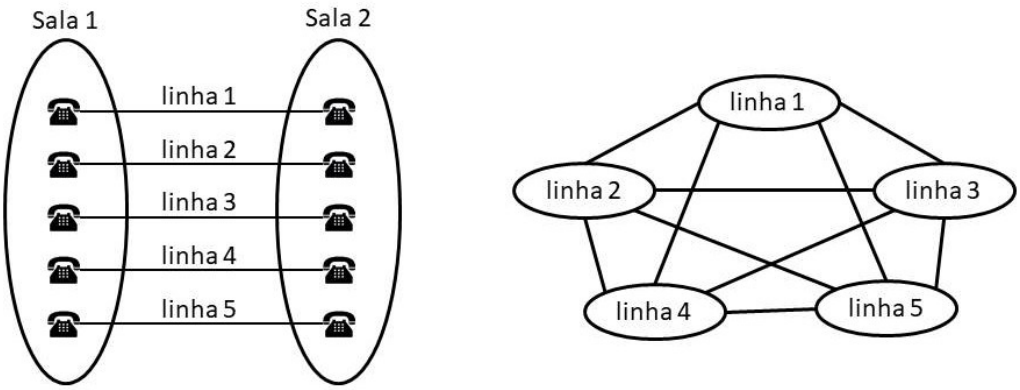


Figura 4.1: Problema do Telefone

é provado que o valor ótimo para θ provoca um tempo esperado de $0.829n$ passos até que o encontro aconteça, ou seja, um limite superior mais baixo que a solução trivial. A tabela 4.1, devida a ANDERSON; WEBER (1990), mostra os valores ótimos de θ para alguns valores de n. Os autores comprovam que esta estratégia não pode ser melhorada para $n = 2$ e 3.

Algoritmo 10: Rendezvous Randomizado em Grafo Completo

início
  Escolha o primeiro vértice a ser visitado com probabilidade uniforme
  se *o encontro não ocorreu* **então**
    repita
      Faça uma das seguintes escolhas:

      1. Com probabilidade θ, permaneça no vértice atual por $n-1$ unidades de tempo;

      2. Com probabilidade $1-\theta$, escolha com probabilidade uniforme uma permutação dos $n-1$ outros vértices (além do atual) e os visite nesta ordem.
    até *os agentes se encontrarem*;
  fim se
fim

n	θ ótimo	T	T/n
2	0.5000	2.0000	1.0000
3	0.3333	2.6667	0.8889
4	0.3220	3.5685	0.8921
5	0.3012	4.3793	0.8759
6	0.2914	5.2133	0.8689
7	0.2842	6.0421	0.8632
8	0.2791	6.8716	0.8590
9	0.2753	7.7008	0.8557
10	0.2722	8.5300	0.8530
50	0.2521	41.6871	0.8338
$n \to \infty$	0.2475		0.8289

Tabela 4.1: Valores ótimos para θ

4.2 Rendezvous em grafos ciclo

Um dos artigos pioneiros nos estudos de rendezvous em grafos, (ALPERN (1995)), considera o cenário de um grafo ciclo de tamanho n conhecido pelos agentes. É dado, então, um algoritmo randomizado de simples descrição e análise: sorteie um dos sentidos com probabilidades iguais e ande $n/2$ passos. É fácil ver que, quando os agentes sortearem sentidos opostos, eles irão se encontrar. Como isso acontece com probabilidade $1/2$, o tempo esperado para o encontro é $O(n)$.

Neste algoritmo, os agentes precisam contar seus passos em função de n, o que exige que eles possuam $O(\log n)$ bits de memória. Em (KRANAKIS; KRIZANC; MORIN (2011)) é estudado este mesmo cenário, porém do ponto de vista da memória requerida. Neste artigo é mostrada a relação entre a quantidade de memória usada pelos agentes e o tempo esperado para o encontro. Nele é concluído que $O(\log \log n)$ bits de memória são suficientes para que o tempo esperado para o encontro neste cenário seja linear.

5 O PROBLEMA DO *RENDEZVOUS* UNIDIMENSIONAL SIMÉTRICO COM MARCADORES

Dois robôs idênticos caem de paraquedas em pontos aleatórios de uma linha infinita e precisam se encontrar. Eles devem abandonar seus paraquedas nos locais onde pousaram e então se reunirem, mas desconhecem sua localização ou distância relativa. A cada ciclo de um relógio universal, cada robô pode andar um passo para a direita, andar um passo para a esquerda ou ficar parado. Os robôs podem ser programados em qualquer linguagem de programação, mas ambos devem executar exatamente o mesmo algoritmo. Qual seria o algoritmo mais eficiente para resolver o problema? Este cenário é um *puzzle* conhecido, utilizado inclusive em entrevistas de grandes empresas de TI, e uma elegante solução determinística é tradicionalmente apresentada como sendo sua melhor resposta (PARACHUTED ROBOTS (2017)).

5.1 Definição e modelo

O Problema do *Rendezvous* Unidimensional Simétrico com marcadores na posição inicial consiste em dois agentes partindo de pontos distintos sobre uma linha infinita, deixando cada um uma marca que identifique suas posições iniciais. Eles devem se encontrar, ambos utilizando a mesma estratégia. Como mencionado, o problema dos robôs paraquedistas é um *puzzle* conhecido que ilustra essa modelagem. Quando um agente encontra o marcador do outro agente, ele passa imediatamente a conhecer de que lado, com relação a si próprio, o outro agente está. Essa informação fundamental é usada para quebrar a simetria dos movimentos, permitindo a elaboração de algoritmos determinísticos que implicam aos agentes não se movimentarem de forma eternamente idêntica. Um desses algoritmos, bastante simples,

consiste em estabelecer um movimento em zigue-zague em torno da posição inicial, com amplitude crescendo geometricamente até que o marcador do outro agente seja localizado e, a partir de então, seguindo sempre na direção em que sabidamente o outro agente estará. Tal estratégia é eficiente para o problema em que um agente móvel busca um ponto fixo na reta, problema que é muitas vezes referido como o *Problema da Vaca Perdida* (BAEZA-YATES; CULBERSON; RAWLINS (1993)). Essa não é, contudo, sequer a melhor solução determinística para o problema em que estamos interessados.

Nas próximas seções revisitaremos a melhor solução conhecida até então para o problema, e mostraremos uma solução randomizada que obtém desempenho ainda melhor. Para efeito de cálculo dos tempo de execução, consideraremos que os agentes têm velocidades idênticas de um "passo por unidade de tempo" e os tempos de encontro serão calculados como uma função do número inicial d de passos que separam os agentes.

Assim como os modelos vistos nos capítulos anteriores, cada passo dos agentes serão modelados como vértices de um grafo, que neste caso, será um grafo linear infinito, ou seja, não existem extremidades que possam ser alcançadas pelos agentes. Apesar deste grafo ser um caso particular de uma árvore, o fato de ser infinito impossibilita a utilização das estratégias visitadas nas seções anteriores. É importante permitirmos que o encontro possa ocorrer sobre as arestas e não somente sobre os vértices, já que a ilustração dos robôs de paraquedas remete a um cenário físico em que não seria interessante permitir que os robôs se cruzem sobre uma aresta sem perceber a presença um do outro.

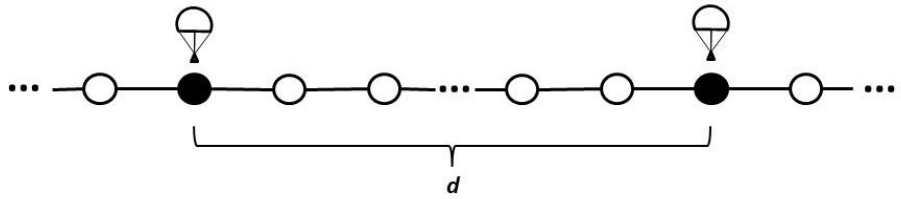


Figura 5.1: Robôs de Paraquedas - Posições Iniciais com Marcação

5.2 A melhor solução conhecida (determinística)

A solução tradicional para o problema (veja, por exemplo, PETER (2016)) consiste em um algoritmo determinístico que é executado em duas etapas:

Algoritmo 11: Rendezvous Unidimensional Determinístico

início
1. **repita**
   - Ande para o vértice da direita
   - Permaneça parado por uma unidade de tempo

até *encontrar a marcação do outro agente*;
2. **repita**
   - Ande para o vértice da direita

até *alcançar o outro agente*;

fim

Os dois agentes executam o mesmo algoritmo. Durante a primeira etapa, eles farão os mesmos movimentos, mantendo inalterada a distância d que os separa. Note, porém, que apenas um deles, digamos o Agente A, encontrará o vértice inicial do outro agente, digamos o Agente B, passando para a segunda etapa do algoritmo. A partir daí, o Agente A persegue o Agente B com velocidade de aproximação igual a um passo a cada duas unidades de tempo, encontrando-o em tempo finito. (veja Figura 5.2).

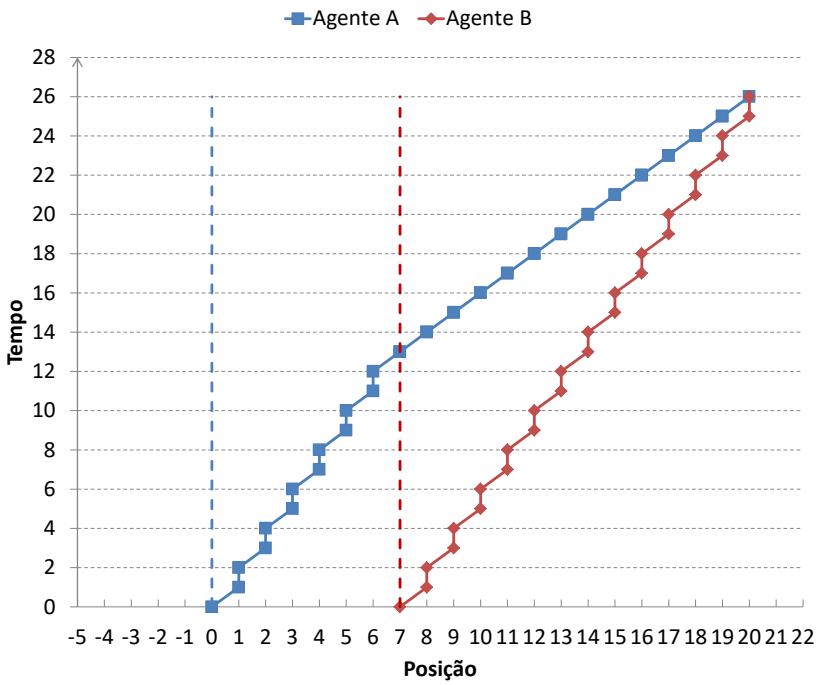


Figura 5.2: Deslocamento dos agentes ao longo do tempo utilizando o algoritmo determinístico

5.2.1 Análise da solução determinística

Na primeira etapa do algoritmo, os agentes andam um passo para a direita a cada duas unidades de tempo. Então, depois de $2d - 2$ ciclos de relógio, eles terão andado uma distância igual a $d - 1$ passos. No ciclo seguinte, os agentes terão percorrido distância d, e o Agente A encontra o vértice marcado pelo Agente B. O tempo total transcorrido até aqui foi portanto de $2d - 1$ unidades de tempo. Inicia-se a etapa da perseguição para o Agente A e, já no ciclo de relógio seguinte, a distância entre os agentes cairá para $d - 1$ (uma vez que o Agente A dará um passo enquanto o Agente B, ainda na primeira etapa do algoritmo, permanecerá parado no mesmo vértice). Deste momento em diante, a cada duas unidades de tempo a distância entre eles se reduzirá em uma unidade. Serão necessárias, portanto, $2(d - 1)$ unidades de tempo para que a distância entre eles caia a zero, totalizando

$1+2(d-1) = 2d-1$ unidades o tempo total de perseguição. Executando, portanto, o algoritmo determinístico descrito acima quando partem de uma distância inicial d um do outro, os agentes levam um tempo total

$$t_{det}(d) = (2d-1) + (2d-1) = 4d-2$$

até se encontrarem.

5.3 Solução proposta (randomizada)

Proporemos agora o seguinte algoritmo randomizado:

Algoritmo 12: Rendezvous Unidimensional Randomizado

início
- Escolha aleatória e uniformemente o sentido inicial do movimento (direita ou esquerda).
- $i \leftarrow 1$

repita
- Ande 2^{i-1} vértices no sentido atual e volte ao vértice inicial
- Inverta o sentido atual
- $i \leftarrow i+1$

até *encontrar a marcação inicial do outro agente*;

repita
- Ande no sentido atual

até *encontrar o outro agente*;

fim

O algoritmo 12 lança mão de uma escolha aleatória do movimento inicial do agente. Em seguida, ele inicia a busca em zigue-zagues (movimentos de ida e volta com alternância de sentido) em torno do próprio vértice inicial, ininterruptamente, com amplitudes crescendo geometricamente, até que a marcação feita pelo outro agente é encontrada. Em seguida, o agente mantém o seu sentido até que ocorra o encontro.

Devido à escolha inicial aleatória dos agentes, existem quatro possíveis cenários em relação ao caminho dos agentes até se encontrarem: em dois deles, os agentes escolhem iniciar a busca seguindo no mesmo sentido; nos outros dois, escolhem sentidos opostos. Quando iniciam no mesmo sentido, um dos agentes, digamos o Agente A, encontrará o marcador inicial do Agente B e permanecerá indo em direção a ele. Quando o Agente B, em seu vai-e-vem, tiver completado seu movimento de ida e estiver voltando a seu ponto inicial, os agentes eventualmente se encontrarão. No caso em que escolhem iniciar para lados opostos, os agentes farão percursos perfeitamente simétricos, isto é, em oposição de fase, e se encontrarão exatamente no vértice (ou aresta) que equidista de suas posições iniciais antes mesmo de terem encontrado o vértice inicial um do outro. É precisamente a possibilidade de ocorrer essa situação de oposição de fase que torna atraente a estratégia randomizada, pois, nesse caso, o encontro se dará de forma bastante rápida, mais do que compensando, em média, o fato de que a estratégia de zigue-zague não os permite se encontrar tão rapidamente no caso complementar (movimentos iniciais no mesmo sentido) quanto a estratégia determinística da perseguição os permitiria. É o que provaremos nas próximas seções.

5.3.1 Análise da solução randomizada

Nesta seção, desenvolveremos os cálculos do tempo esperado até o encontro dos agentes que iniciam a busca a uma distância $d \in \mathbb{N}$ um do outro. Para isso, analisaremos o tempo de encontro, como uma função de d, em cada um dos quatro possíveis cenários advindos da escolha aleatória inicial de cada agente.

Sejam os tempos de encontro $t_{EE}(d)$ e $t_{DD}(d)$ aqueles referentes aos cenários em que os agentes iniciam ambos para o vértice da esquerda (como ilustrado na Figura 5.3) ou ambos para o vértice da direita, respectivamente. Por simetria,

$t_{EE} = t_{DD}$. Seja k o índice da primeira oscilação (movimento de vai-e-vem) cuja amplitude é maior ou igual à distância inicial d. Temos, então,

$$k = \lceil \log_2 d \rceil + 1,$$

e a oscilação k será exatamente aquela em que um dos agentes, digamos o Agente A, encontrará o vértice marcado pelo outro, digamos o Agente B. A oscilação $k-1$ será, portanto, a última que os agentes executarão completamente, retornando aos vértices de origem. Ao fim do movimento de ida da k-ésima oscilação (quando a distância entre eles será ainda igual a d), os agentes passarão a andar em sentidos opostos, pois o Agente A manterá o sentido de seu movimento (indo de encontro ao Agente B), ao passo que o Agente B terá entrado normalmente em seu movimento de retorno à posição de origem (indo de encontro ao Agente A). A partir daquele momento, portanto, os agentes levarão tempo $d/2$ até se encontrarem.

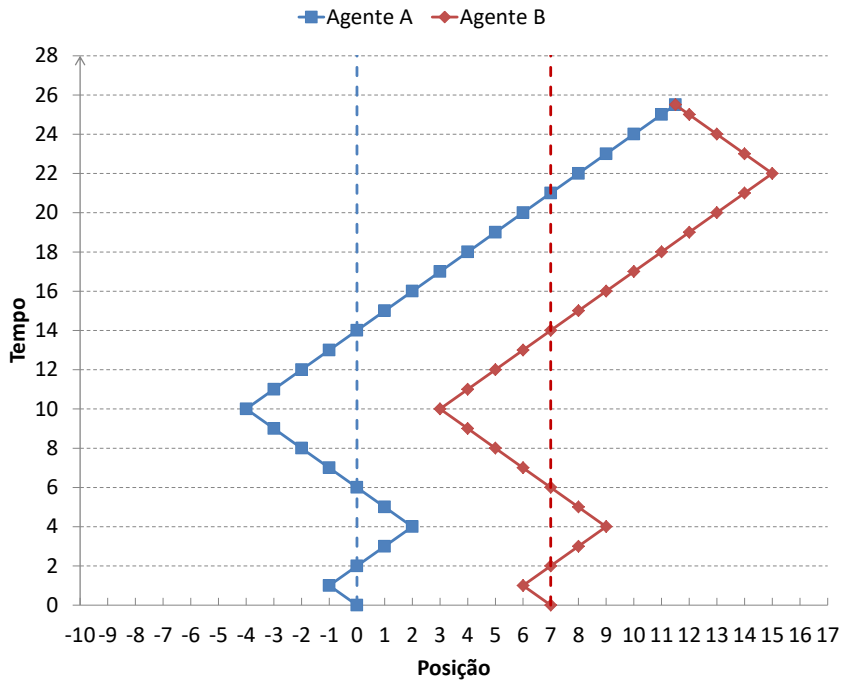


Figura 5.3: Deslocamento dos agentes utilizando o algoritmo randomizado, quando ambos escolhem iniciar no mesmo sentido

Os tempos t_{EE} e t_{DD} são, assim, compostos pelas seguintes parcelas:

- tempo até completar a $(k-1)$-ésima oscilação: $\sum_{i=1}^{k-1} 2^i$;
- duração do movimento de ida da k-ésima oscilação: $2^{\lceil \log_2 d \rceil}$;
- tempo andando em sentidos opostos após o movimento de ida da k-ésima oscilação: $d/2$.

Simplificando a soma das parcelas acima, obtemos

$$\begin{aligned} t_{EE}(d) = t_{DD}(d) &= \left(\sum_{i=1}^{\lceil \log_2 d \rceil} 2^i \right) + 2^{\lceil \log_2 d \rceil} + d/2 \\ &= \left(2^{\lceil \log_2 d \rceil + 1} - 2\right) + 2^{\lceil \log_2 d \rceil} + d/2 \\ &= 3 \cdot 2^{\lceil \log_2 d \rceil} + d/2 - 2. \end{aligned}$$

Seja agora o tempo de encontro t_{DE}, referente ao cenário em que os agentes iniciam a primeira oscilação em sentidos opostos, isto é, indo na direção um do outro (o agente mais à esquerda anda para o vértice da direita, e o agente mais à direita anda para o vértice da esquerda, como ilustrado na Figura 5.4a). Nesse cenário, eles sempre seguirão em direções opostas e, portanto, se encontrarão no vértice (ou aresta) equidistante dos vértices de partida. As oscilações de índice ímpar (a primeira oscilação é aquela de índice 1) são aquelas em que os agentes se aproximam, enquanto as de índice par são aquelas em que eles se afastam um do outro. Portanto, o índice da oscilação em que o encontro ocorre é o primeiro índice ímpar k' tal que $2^{k'-1} \geq d/2$. Isto é, $k' = 1 + \lceil \log_2(d/2) \rceil = \lceil \log_2 d \rceil$, se esse valor é ímpar, ou, caso contrário, esse valor mais uma unidade. Podemos escrever, de forma mais sucinta,

$$k' = \lceil \log_2 d \rceil + (\lceil \log_2 d \rceil + 1) \,\%2,$$

onde a notação "%2" se refere à operação "módulo 2" (o resto da divisão por 2).

O tempo total $t_{DE}(d)$ é dado, portanto, pelo somatório das oscilações anteriores à k'-ésima, mais as $d/2$ unidades de tempo que correspondem ao movimento parcial de ida da própria k'-ésima oscilação (durante o qual os agentes vão de encontro um ao outro a partir de seus pontos iniciais):

$$\begin{aligned} t_{DE} &= \left(\sum_{i=1}^{k'-1} 2^i\right) + d/2 \\ &= \left(2^{\lceil \log_2 d \rceil + (\lceil \log_2 d \rceil + 1)\%2} - 2\right) + d/2 \\ &= 2^{(\lceil \log_2 d \rceil + 1)\%2} \cdot 2^{\lceil \log_2 d \rceil} + d/2 - 2. \end{aligned}$$

Finalmente, seja t_{ED} o tempo de encontro quando os agente iniciam a primeira oscilação se afastando um do outro (Figura 5.4b). Assim como no cenário anterior, eles sempre se encontrarão no ponto central entre os vértices marcados. A diferença, nesse caso, é que eles se aproximam nas oscilações de índice par. Por raciocínio análogo ao do caso anterior, temos que o índice k'' da oscilação em que eles se encontram será dado por

$$k'' = \lceil \log_2 d \rceil + \lceil \log_2 d \rceil \%2,$$

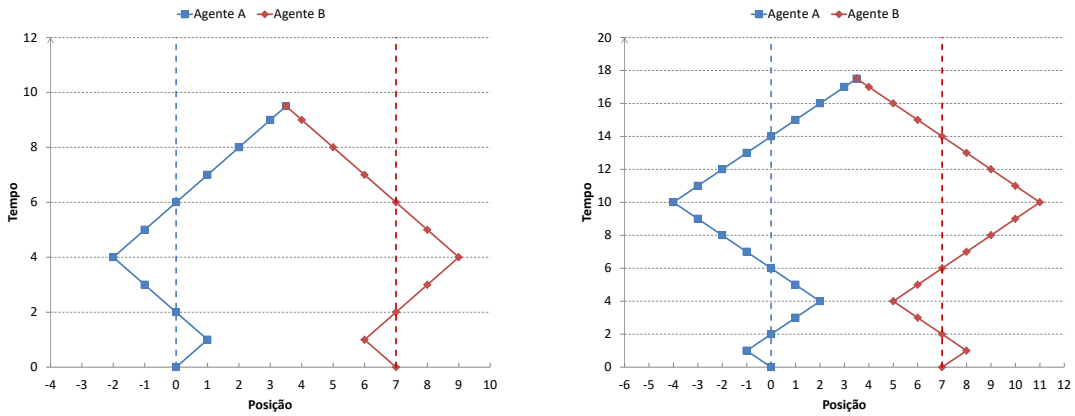
e o tempo total desse cenário é calculado como se segue:

$$\begin{aligned} t_{ED} &= \left(\sum_{i=1}^{k''-1} 2^i\right) + d/2 \\ &= \left(2^{\lceil \log_2 d \rceil + \lceil \log_2 d \rceil \%2} - 2\right) + d/2 \\ &= 2^{\lceil \log_2 d \rceil \%2} \cdot 2^{\lceil \log_2 d \rceil} + d/2 - 2. \end{aligned}$$

Essa expressão é válida para $d \geq 2$, já que para $d = 1$ resultaria em um índice negativo para o limite do somatório, porém é fácil verificar que para $d = 1$, $t_{ED} = 5/2$.

De posse dos tempos para cada uma das possíveis escolhas iniciais dos agentes, temos que o tempo esperado de execução do algoritmo randomizado para uma distância inicial $d \geq 2$ é dado por:

$$\begin{aligned} E[t_{rand}(d)] &= \frac{t_{DD} + t_{EE} + t_{DE} + t_{ED}}{4} \\ &= \frac{t_{DD}}{2} + \frac{t_{DE} + t_{ED}}{4} \\ &= \frac{3 \cdot 2^{\lceil \log_2 d \rceil} + d/2 - 2}{2} + \frac{3 \cdot 2^{\lceil \log_2 d \rceil} + d - 4}{4} \\ &= 3 \cdot 2^{\lceil \log_2 d \rceil - 1} + d/4 - 1 + 3 \cdot 2^{\lceil \log_2 d \rceil - 2} + d/4 - 1 \\ &= 9 \cdot 2^{\lceil \log_2 d \rceil - 2} + d/2 - 2. \end{aligned}$$



(a) Iniciam se aproximando

(b) Iniciam se afastando

Figura 5.4: Deslocamento dos agentes utilizando o algoritmo randomizado, quando escolhem iniciar em direções opostas

5.4 Análise comparativa

Sejam $T_{rand}(D)$ e $T_{det}(D)$ variáveis aleatórias que correspondem aos tempos de execução dos algoritmos randomizado e determinístico, respectivamente, quando a distância inicial d é uma variável aleatória escolhida uniformemente do intervalo

$[1, D]$. Queremos mostrar que o desempenho esperado do algoritmo randomizado é sempre melhor do que o do algoritmo determinístico, qualquer que seja o tamanho D do intervalo considerado. Mais formalmente, queremos mostrar que

$$E[T_{rand}(D)] \leq E[T_{det}(D)],$$

ou ainda, que

$$\sum_{i=1}^{D} t_{rand}(i) \cdot P[d = i] < \sum_{i=1}^{D} t_{det}(i) \cdot P[d = i]$$

para todo $D \in \mathbb{N}$. O operador $E[\cdot]$ aparece aqui indicando a esperança de uma variável aleatória, e $P[\cdot]$ indica a probabilidade de ocorrência do evento indicado.

Como a distribuição de d é uniforme, as probabilidades para todos os possíveis valores de d são idênticas. Portanto, é suficiente compararmos os somatórios dos tempos entre 1 e D. Em outras palavras, queremos mostrar que

$$\sum_{i=1}^{D} t_{rand}(i) < \sum_{i=1}^{D} t_{det}(i). \tag{5.1}$$

Sejam $S_{rand}(D)$ e $S_{det}(D)$ os somatórios dos tempos de execução para d de 1 até D dos algoritmos randomizado e determinístico, respectivamente, correspondendo aos dois termos na desigualdade (5.1). Temos, então, que

$$S_{det}(D) = \sum_{i=1}^{D} t_{det}(i) = \sum_{i=1}^{D} 4i - 2 = \frac{D(2 + 4D - 2)}{2} = 2D^2. \tag{5.2}$$

Para o cálculo de $S_{rand}(D)$, devemos considerar a primeira parcela do somatório separadamente, já que a expressão encontrada para t_{rand} é válida apenas para $d \geq 2$.

$$\begin{aligned} S_{rand}(D) &= \sum_{i=1}^{D} t_{rand}(i) = t_{rand}(1) + \sum_{i=2}^{D} 9 \cdot 2^{\lceil \log_2 d \rceil - 2} + d/2 - 2 \\ &= t_{rand}(1) + 9 \cdot \sum_{i=2}^{D} 2^{\lceil \log_2 d \rceil - 2} + 1/2 \cdot \sum_{i=2}^{D} d - \sum_{i=2}^{D} 2. \end{aligned} \tag{5.3}$$

Mas

$$\begin{aligned}\sum_{i=2}^{D} 2^{\lceil \log_2 d \rceil - 2} &= \sum_{p=1}^{\lceil \log_2 D \rceil} (2^{p-1} \cdot 2^{p-2}) - (2^{\lceil \log_2 D \rceil} - D)2^{\lceil \log_2 D \rceil - 2} \\ &= \frac{(4^{\lceil \log_2 D \rceil} - 1)}{6} - (2^{2\lceil \log_2 D \rceil - 2} - D \cdot 2^{\lceil \log_2 D \rceil - 2}) \\ &= \frac{(2^{2\lceil \log_2 D \rceil} - 1) - 6 \cdot 2^{2\lceil \log_2 D \rceil - 2} - 6 \cdot D \cdot 2^{\lceil \log_2 D \rceil - 2}}{6} \\ &= \frac{2^{\lceil \log_2 D \rceil - 1}(2^{\lceil \log_2 D \rceil + 1} - 3 \cdot D \cdot 2^{\lceil \log_2 D \rceil} + 3D) - 1}{6} \\ &= \frac{2^{\lceil \log_2 D \rceil - 1}(3D - 2^{\lceil \log_2 D \rceil}) - 1}{6},\end{aligned}$$

o que nos permite substituir em (5.3) para obter

$$\begin{aligned}S_{rand}(D) &= \frac{3}{2} + 9\left(\frac{2^{\lceil \log_2 D \rceil - 1}(3D - 2^{\lceil \log_2 D \rceil}) - 1}{6}\right) + \frac{1}{2}\frac{(D+2)(D-1)}{2} - (2D-2) \\ &= \frac{3}{2} + 9D \cdot 2^{\lceil \log_2 D \rceil - 2} - 3 \cdot 2^{2\lceil \log_2 D \rceil - 2} - \frac{3}{2} + \frac{D^2 + D - 2}{4} - 2D + 2 \\ &= 3 \cdot 2^{\lceil \log_2 D \rceil - 2}(3D - 2^{\lceil \log_2 D \rceil}) + \frac{D^2 - 7D + 6}{4} \\ &= \frac{1}{4} \cdot \left(3 \cdot 2^{\lceil \log_2 D \rceil}(3D - 2^{\lceil \log_2 D \rceil}) + D^2 - 7D + 6\right).\end{aligned}$$

Obtidas as expressões para $S_{det}(D)$ e $S_{rand}(D)$, queremos mostrar que a desigualdade $S_{rand}(D) < S_{det}(D)$ é verdadeira para todo $D \in \mathbb{N}$:

$$\begin{aligned}\frac{1}{4} \cdot \left(3 \cdot 2^{\lceil \log_2 D \rceil}(3D - 2^{\lceil \log_2 D \rceil}) + D^2 - 7D + 6\right) &< 2D^2 \\ 3 \cdot 2^{\lceil \log_2 D \rceil}(3D - 2^{\lceil \log_2 D \rceil}) + D^2 - 7D + 6 &< 8D^2 \\ 3 \cdot 2^{\lceil \log_2 D \rceil}(3D - 2^{\lceil \log_2 D \rceil}) - 7D^2 - 7D + 6 &< 0\end{aligned}$$

Fazendo $\omega = 2^{\lceil \log_2 D \rceil}$ e considerando o lado esquerdo da desigualdade acima como uma função $f(\omega)$, ocorre que $f(\omega)$ tem máximo em $\omega = 3D/2$. Portanto,

$f(\omega) \leq f(3D/2)$ para todo ω no domínio da função. Resta mostrar que

$$\begin{aligned} f(3D/2) &< 0 \\ 3 \cdot \frac{3D}{2}(3D - \frac{3D}{2}) - 7D^2 - 7D + 6 &< 0 \\ -\frac{D^2}{4} - 7D + 6 &< 0 \end{aligned}$$

A parte positiva da solução da inequação acima é $D > \alpha$, onde $\alpha \approx 0.8324$, compreendendo portanto todos os inteiros positivos. Dessa forma, $S_{rand} < S_{det}$ é verdadeiro para todo $D \in \mathbb{N}$, e portanto, a estratégia randomizada permite que os agentes se encontrem, em média, mais rapidamente do que a estratégia determinística, como queríamos demonstrar.

Além de apresentar um desempenho esperado melhor, quando a distância inicial é escolhida aleatória e uniformemente do conjunto dos D primeiros naturais, é possível mostrar que o desempenho do algoritmo randomizado supera o do algoritmo determinístico *em mais da metade dos valores* de d no intervalo $[1, D]$, para todo D natural. Em outras palavras, se quisermos *maximizar a probabilidade de escolher a melhor estratégia* a ser executada para um valor qualquer de d escolhido uniformemente do conjunto dos primeiros D naturais, a melhor escolha é a estratégia randomizada. Para garantirmos isso, não basta sabermos, como já sabemos, que, *em média* (considerando-se todas as possíveis distâncias iniciais), a estratégia randomizada leva menos tempo; é preciso mostrar que, para qualquer $D \in \mathbb{N}$, mais da metade das distâncias d tomadas no intervalo $[1, D]$ satisfaz $E[t_{rand}(d)] < t_{det}(d)$, desigualdade essa que se traduz em

$$\begin{aligned} \frac{9}{4} \cdot 2^{\lceil \log_2 d \rceil} + \frac{d}{2} - 2 &< 4d - 2 \\ 2^{\lceil \log_2 d \rceil} &< \frac{7d}{2} \cdot \frac{4}{9} \\ \frac{2^{\lceil \log_2 d \rceil}}{d} &< \frac{14}{9} \end{aligned}$$

Seja $\alpha(d) = 2^{\lceil \log_2 d \rceil}/d$. Da última desigualdade, pode-se concluir que, para uma uma distância inicial d, o algoritmo randomizado terá um melhor desempenho esperado que o algoritmo determinístico quando $\alpha(d) < 14/9$. Por simplicidade, o conjunto de todas as distâncias iniciais em que o algoritmo randomizado tem um melhor desempenho será chamado de D_{rand}, e o das distância iniciais em o algoritmo determinístico possui um melhor desempenho será chamado de D_{det}.

Seja p uma potência de 2. Definimos como F_p o intervalo $[p/2 + 1, p]$. Note que, para qualquer $i \in F_p$, $2^{\lceil \log_2 i \rceil} = p$, e portanto, o valor de α será decrescente dentro do intervalo. Dito isto, importa-nos responder a seguinte questão: qual o menor valor de p, tal que o menor elemento de F_p pertence a D_{det}? É fácil ver que

$$\frac{p}{p/2+1} > 14/9 \Rightarrow 9p > 7p + 14 \Rightarrow p > 7.$$

Como p é uma potência de 2, temos que F_8 é o primeiro intervalo cujo primeiro elemento pertence a D_{det}. A partir deste, todos os próximos intervalos têm um d_{det} como primeiro elemento e um d_{rand} como último (já que o α referente a uma potência de 2 sempre será igual a 1). Por isso, é possível estabelecer um "ponto de corte" dentro de cada intervalo F_p, onde os valores menores que esse ponto pertencerão a D_{det}, e os demais pertencerão a D_{rand} (veja a figura 5.5). Esse ponto de corte é precisamente o valor $9p/14$, pois $\alpha(9p/14) = 14/9$. Esse valor não pode ser um inteiro, portanto $\lfloor 9p/14 \rfloor$ é o maior elemento do intervalo que pertence a D_{det}, e $\lceil 9p/14 \rceil$ é o menor elemento do intervalo que já pertence a D_{rand}.

A i-ésima posição do intervalo F_p contém o valor $p/2 + i$. Logo, a posição do valor $\lfloor 9p/14 \rfloor$ no intervalo é dada por

$$\lfloor 9p/14 \rfloor - p/2 = \lfloor 9p/14 - p/2 \rfloor = \lfloor p/7 \rfloor$$

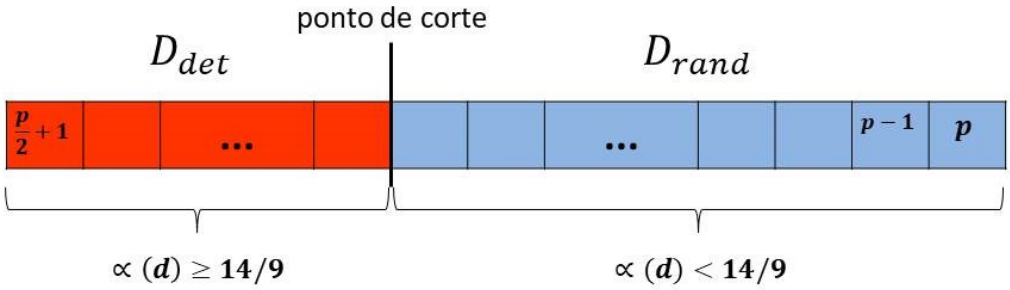


Figura 5.5: Intervalo F_p

Com isso, sabemos que, dos $p/2$ valores do intervalo F_p, exatamente $\lfloor p/7 \rfloor$ pertencem a D_{det}, e os demais $\lceil 5p/14 \rceil$ pertencem a D_{rand}. Analogamente, o intervalo seguinte F_{2p} possui $\lfloor 2p/7 \rfloor$ valores em D_{det} e $\lceil 5p/14 \rceil$ valores em D_{rand}; e assim por diante. Conclui-se, então, por indução, que a quantidade de valores em D_{det} de um intervalo é sempre menor que a quantidade de valores em D_{rand} do intervalo imediatamente anterior. A base da indução é o intervalo F_8, que contém o menor valor natural em D_{det}, como visto anteriormente.

Por fim, segue da afirmação anterior que, para qualquer $D \in \mathbb{N}$, o intervalo $[1, D]$ possui mais valores em D_{rand} que em D_{det}, o que significa que para uma distância inicial escolhida aleatória e uniformemente desse intervalo, a chance de que o tempo esperado do algoritmo randomizado seja inferior ao tempo do algoritmo determinístico é maior do que 50%, como queríamos demonstrar.

5.5 Versão com retorno ao ponto inicial

Suponha agora que os robôs paraquedistas, após se encontrarem, precisem retornar ao vértice de partida. Numa tal versão, é necessário acrescentar o tempo

de retorno entre o ponto de encontro e a posição inicial mais distante. Novamente, calcularemos e compararemos os desempenhos das estratégias determinística e randomizada.

5.5.1 Solução determinística

Sejam 0 e d as posições iniciais dos Agentes A e B, respectivamente (estes valores não são percebidos pelos agentes, apenas nos ajudam na análise do algoritmo). O Agente A encontra o o vértice inicial de B na posição d e em seguida permanece se movendo um vértice para a direita a cada ciclo de relógio durante $2d-1$ unidades de tempo, conforme o tempo de perseguição calculado na Seção 5.2.1. Portanto, o encontro acontece no vértice $3d-1$.

Seja $t'_{det}(d)$ o tempo de execução do algoritmo determinístico para a versão com retorno. Temos, então, que $t'_{det}(d)$ é igual a $t_{det}(d)$ mais o tempo de retorno do Agente A, isto é,

$$t'_{det}(d) = (4d-2) + (3d-1) = 7d-3.$$

5.5.2 Solução randomizada

Para o cálculo do tempo esperado para o algoritmo randomizado é necessário considerar o tempo de retorno dos agentes ao ponto inicial em cada um dos 4 casos descritos na Seção 5.3 referentes às escolhas dos agentes para os sentidos iniciais de seus movimentos.

Os casos mais simples são aqueles em que os agentes escolhem direções opostas, pois o ponto de encontro será exatamente no vértice (ou aresta) central, equi-

distante entre os pontos iniciais, sendo necessário um tempo extra $d/2$ para que os agentes retornem. Sejam $t'_{DE}(d)$ e $t'_{ED}(d)$ os tempos de execução para estes casos, para $d \geq 2$. Segue, então, que

$$t'_{DE}(d) = t_{DE}(d) + d/2 = 2^{(\lceil \log_2 d \rceil + 1)\%2} \cdot 2^{\lceil \log_2 d \rceil} + d - 2;$$
$$t'_{ED}(d) = t_{ED}(d) + d/2 = 2^{(\lceil \log_2 d \rceil)\%2} \cdot 2^{\lceil \log_2 d \rceil} + d - 2.$$

Sejam $t'_{EE}(d)$ e $t'_{DD}(d)$ os tempos de execução para os casos em que os agentes escolhem iniciar os movimentos no mesmo sentido. Nesses casos, o agente que mais se afastará do seu ponto inicial será aquele que encontrar o vértice inicial (digamos o Agente A) marcado pelo outro agente. O algoritmo, então, se encerrará quando este retornar ao seu ponto de origem. Conforme visto na Seção 5.3, a marcação do Agente B será encontrada na oscilação de amplitude $2^{\lceil \log_2 d \rceil}$. Ao final desta, o Agente A se afastará ainda mais de seu ponto inicial uma distância $d/2$, quando ocorrerá o encontro. Portanto,

$$\begin{aligned} t'_{EE}(d) = t'_{DD}(d) &= t_{EE}(d) + d/2 + 2^{\lceil \log_2 d \rceil} \\ &= (3 \cdot 2^{\lceil \log_2 d \rceil} + d/2 - 2) + d/2 + +2^{\lceil \log_2 d \rceil} \\ &= 4 \cdot 2^{\lceil \log_2 d \rceil} + d - 2 \end{aligned}$$

Logo, o tempo esperado do algoritmo randomizado para esta versão, quando a distância inicial entre os agentes é igual a $d \geq 2$, é

$$\begin{aligned} E[t'_{rand}(d)] &= \frac{1}{4}(t'_{DE}(d) + t'_{ED}(d) + t'_{EE}(d) + t'_{DD}(d)) = \frac{1}{4}(t'_{DE}(d) + t'_{ED}(d)) + \frac{1}{2}(t'_{EE}(d)) \\ &= \frac{1}{4}(3 \cdot 2^{\lceil \log_2 d \rceil} + 2d - 4) + \frac{1}{2}(4 \cdot 2^{\lceil \log_2 d \rceil} + d - 2) \\ &= \frac{1}{4}(11 \cdot 2^{\lceil \log_2 d \rceil} + 4d - 8) = \frac{11}{4} \cdot 2^{\lceil \log_2 d \rceil} + d - 2. \end{aligned}$$

5.5.3 Análise comparativa

Vimos que o ponto de encontro fornecido pelo algoritmo determinístico estará sempre a uma distância $3d-1$ de um dos vértices iniciais. Enquanto isso, o algoritmo

randomizado estabelece que o encontro se dará na posição que acrescenta o menor tempo possível à fase de retorno com probabilidade 1/2. Além disso, mesmo nos piores casos da estratégia randomizada, a distância ao ponto de origem mais distante é sempre menor que aquela causada pela estratégia determinística, já que $2^{\lceil \log_2 d \rceil} + d/2 < 3d - 1$ para qualquer $d \in \mathbb{N}$. Os cálculos a seguir mostram a comparação entre os tempos de execução dos dois algoritmos para a versão em que os agentes devem retornar ao ponto inicial, e como essa distância entre o ponto de encontro e os marcadores impactam em seus desempenhos.

Queremos, portanto, calcular para quais distâncias iniciais o tempo esperado de execução do algoritmo randomizado é menor que o do algoritmo determinístico. Ou seja,

$$\begin{aligned} E[t'_{rand}(d)] &< t'_{det}(d) \\ \frac{11}{4} \cdot 2^{\lceil \log_2 d \rceil} + d - 2 &< 7d - 3 \\ 2^{\lceil \log_2 d \rceil} &< \frac{24d - 4}{11} \end{aligned}$$

Para todo $d \geq 2$, é verdade que

$$2^{\lceil \log_2 d \rceil} < 2d \leq \frac{24d - 4}{11},$$

e para o único caso entre os naturais não coberto pelas expressões acima, $d = 1$, segue que $E[t'_{rand}(1)] = 5/2 < t'_{det}(1) = 4$.

Portanto, para a versão com retorno às posições iniciais, a estratégia randomizada possui um desempenho superior à determinística para *todas* as distâncias iniciais $d \in \mathbb{N}$.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, estudamos uma classe de problemas conhecida como Rendezvous Simétrico em Grafos. Mostramos suas principais variantes encontradas na literatura para diversas classes de grafos e suas respectivas soluções. Pudemos perceber que uma grande quantidade de parâmetros em cada modelo estudado tem impacto direto na viabilidade da solução e na eficiência das técnicas utilizadas. Vimos que características, tanto do grafo (assimetria da topologia, orientação, etc), quanto dos agentes (rótulos únicos, capacidade de fazer marcações, etc) podem ser ferramentas úteis na construção de soluções determinísticas, cujo principal desafio é a quebra da simetria dos movimentos dos agentes. Da mesma forma, outras características como pesos dos agentes e quantidade de memória podem significar limitações que demandam soluções mais elaboradas. Combinações destas características dão origem a modelos variados, cada um com suas particularidades e restrições, fazendo com que essa classe de problemas desperte interesse na exploração de novos algoritmos específicos para cada caso.

Para o ambiente unidimensional simétrico, contribuímos com uma estratégia randomizada e analisamos seu desempenho em função da distância inicial. Mostramos que, para uma distância inicial d escolhida aleatoriamente de um intervalo $[1, D]$, em média a estratégia randomizada leva menos tempo que a estratégia determinística mais rápida conhecida, para todo $D \in \mathbb{N}$. Mostramos também que, para todo $D \in \mathbb{N}$, é maior do que 50% a probabilidade de que o tempo esperado de execução do algoritmo randomizado para distância d uniformemente escolhida em $[1, D]$ seja menor que o tempo do algoritmo determinístico para a mesma entrada.

Uma vez que não apenas o tempo de execução, mas também a distância do

ponto de encontro aos pontos iniciais é reduzida pela estratégia proposta, consideramos a versão em que os agentes devem retornar aos pontos de origem após se encontrarem. Para esta versão, o algoritmo randomizado apresenta tempo esperado de execução menor que o determinístico para *toda* distância inicial $d \in \mathbb{N}$.

Com o objetivo de ganhar intuição para outras soluções, implementamos algumas variações da estratégia randomizada, por exemplo, a escolha aleatória da direção a cada passo do agente (conhecida como "andar do bêbado"), porém notamos que a distância entre os agentes poderia aumentar indefinidamente e eles nunca se encontrarem. A extensão da estratégia para outras classes de grafos (o ciclo, por exemplo) não se mostrou melhor do que as soluções já encontradas na literatura.

Como trabalhos futuros, sugerimos a modificação da razão da progressão geométrica usada no crescimento da amplitude das oscilações da estratégia randomizada. Possivelmente o valor ótimo para esta base pode ser alcançado através de técnicas de otimização.

REFERÊNCIAS

ALPERN, S. The rendezvous search problem. **SIAM Journal on Control and Optimization**, [S.l.], v.33, n.3, p.673–683, 1995.

ALPERN, S. Rendezvous search: a personal perspective. **Operations Research**, [S.l.], v.50, n.5, p.772–795, 2002.

ALPERN, S. Rendezvous search on labeled networks. **Naval Research Logistics (NRL)**, [S.l.], v.49, n.3, p.256–274, 2002.

ALPERN, S.; GAL, S. **The theory of search games and rendezvous**. [S.l.]: Springer Science & Business Media, 2006. v.55.

ANDERSON, E. J.; ESSEGAIER, S. Rendezvous search on the line with indistinguishable players. **SIAM Journal on Control and Optimization**, [S.l.], v.33, n.6, p.1637–1642, 1995.

ANDERSON, E. J.; FEKETE, S. P. Two dimensional rendezvous search. **Operations Research**, [S.l.], v.49, n.1, p.107–118, 2001.

ANDERSON, E. J.; WEBER, R. The rendezvous problem on discrete locations. **Journal of Applied Probability**, [S.l.], v.27, n.4, p.839–851, 1990.

BABA, D. et al. Space-Optimal Rendezvous of Mobile Agents in Asynchronous Trees. In: SIROCCO. **Anais...** [S.l.: s.n.], 2010. p.86–100.

BAEZA-YATES, R. A.; CULBERSON, J. C.; RAWLINS, G. J. E. Searching in the plane. **Information and Computation**, [S.l.], v.106, p.234–252, 1993.

BASTON, V.; GAL, S. Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. **SIAM Journal on Control and Optimization**, [S.l.], v.36, n.6, p.1880–1889, 1998.

BEVERIDGE, A.; OZSOYELLER, D.; ISLER, V. Symmetric rendezvous on the line with an unknown initial distance. , [S.l.], 2011.

CZYZOWICZ, J. et al. The power of tokens: rendezvous and symmetry detection for two mobile agents in a ring. **Lecture Notes in Computer Science**, [S.l.], v.4910, p.234–246, 2008.

CZYZOWICZ, J.; KOSOWSKI, A.; PELC, A. How to meet when you forget: log-space rendezvous in arbitrary graphs. **Distributed Computing**, [S.l.], v.25, n.2, p.165–178, 2012.

CZYZOWICZ, J.; KOSOWSKI, A.; PELC, A. Time versus space trade-offs for rendezvous in trees. **Distributed Computing**, [S.l.], v.27, n.2, p.95–109, 2014.

CZYZOWICZ, J.; PELC, A.; LABOUREL, A. How to meet asynchronously (almost) everywhere. **ACM Transactions on Algorithms (TALG)**, [S.l.], v.8, n.4, p.37, 2012.

DE MARCO, G. et al. Asynchronous deterministic rendezvous in graphs. **Theoretical Computer Science**, [S.l.], v.355, n.3, p.315–326, 2006.

DESSMARK, A. et al. Deterministic rendezvous in graphs. **Algorithmica**, [S.l.], v.46, n.1, p.69–96, 2006.

DIMAROGONAS, D. V.; KYRIAKOPOULOS, K. J. On the rendezvous problem for multiple nonholonomic agents. **IEEE Transactions on automatic control**, [S.l.], v.52, n.5, p.916–922, 2007.

FARRUGIA, A. et al. Deterministic rendezvous in restricted graphs. In: INTERNATIONAL CONFERENCE ON CURRENT TRENDS IN THEORY AND PRACTICE OF INFORMATICS. **Anais...** [S.l.: s.n.], 2015. p.189–200.

FLOCCHINI, P. et al. Multiple mobile agent rendezvous in a ring. In: LATIN. **Anais...** [S.l.: s.n.], 2004. v.4, p.599–608.

FRAIGNIAUD, P.; PELC, A. Delays induce an exponential memory gap for rendezvous in trees. **ACM Transactions on Algorithms (TALG)**, [S.l.], v.9, n.2, p.17, 2013.

GAL, S. Rendezvous search on the line. **Operations Research**, [S.l.], v.47, n.6, p.974–976, 1999.

GĄSIENIEC, L. et al. Optimal memory rendezvous of anonymous mobile agents in a unidirectional ring. **SOFSEM 2006: Theory and Practice of Computer Science**, [S.l.], p.282–292, 2006.

ISRAELI, A.; JALFON, M. Token management schemes and random walks yield self-stabilizing mutual exclusion. In: ACM SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING. **Proceedings...** [S.l.: s.n.], 1990. p.119–131.

KNUTH, D. E. **The Art of Computer Programming, Volume 1 (3rd Ed.)**: fundamental algorithms. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1997.

KOWALSKI, D. R.; MALINOWSKI, A. How to meet in anonymous network. In: SIROCCO. **Anais...** [S.l.: s.n.], 2006. p.44–58.

KRANAKIS, E. et al. Mobile agent rendezvous in a ring. In: DISTRIBUTED COMPUTING SYSTEMS, 2003. PROCEEDINGS. 23RD INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. p.592–599.

KRANAKIS, E.; KRIZANC, D.; MARKOU, E. Mobile agent rendezvous in a synchronous torus. In: LATIN. **Anais...** [S.l.: s.n.], 2006. p.653–664.

KRANAKIS, E.; KRIZANC, D.; MORIN, P. Randomized rendezvous with limited memory. **ACM Transactions on Algorithms (TALG)**, [S.l.], v.7, n.3, p.34, 2011.

KRANAKIS, E.; KRIZANC, D.; RAJSBAUM, S. Mobile agent rendezvous: a survey. In: INTERNATIONAL COLLOQUIUM ON STRUCTURAL INFORMATION AND COMMUNICATION COMPLEXITY. **Anais...** [S.l.: s.n.], 2006. p.1–9.

PARACHUTED Robots. Acessado: 2017-12-20, `https://en.wikibooks.org/wiki/Puzzles/Logic_puzzles/Parachuted_Robots`.

PELC, A. Deterministic rendezvous in networks: a comprehensive survey. **Networks**, [S.l.], v.59, n.3, p.331–347, 2012.

PETER, D. **Web Page**. Acessado: 2016-12-10, `http://david-peter.de/parachuting-robots/`.

SINHA, A.; GHOSE, D. Generalization of linear cyclic pursuit with application to rendezvous of multiple autonomous agents. **IEEE Transactions on Automatic Control**, [S.l.], v.51, n.11, p.1819–1824, 2006.

TA-SHMA, A.; ZWICK, U. Deterministic rendezvous, treasure hunts and strongly universal exploration sequences. In: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS. **Proceedings...** [S.l.: s.n.], 2007. p.599–608.

THOMAS, L. C.; HULME, P. Searching for targets who want to be found. **Journal of the Operational Research Society**, [S.l.], v.48, n.1, p.44–50, 1997.

YAMASHITA, M.; KAMEDA, T. Computing on anonymous networks. I. Characterizing the solvable cases. **IEEE Transactions on parallel and distributed systems**, [S.l.], v.7, n.1, p.69–89, 1996.