TÉCNICAS DE LEILÃO APLICADAS À COORDENAÇÃO DE MÚLTIPLOS ROBÔS EM MISSÕES DE EXPLORAÇÃO DE AMBIENTES

RODOLFO CARNEIRO CAVALCANTE

TÉCNICAS DE LEILÃO APLICADAS À COORDENAÇÃO DE MÚLTIPLOS ROBÔS EM MISSÕES DE EXPLORAÇÃO DE AMBIENTES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Luiz Chaimowicz
Co-orientador: Thiago F. Noronha

Belo Horizonte

Fevereiro de 2013

Carneiro Cavalcante, Rodolfo
C376t Técnicas de Leilão Aplicadas à Coordenação de Múltiplos Robôs em Missões de Exploração de Ambientes / Rodolfo Carneiro Cavalcante. — Belo Horizonte, 2013
xviii, 74 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Luiz Chaimowicz
Co-orientador: Thiago F. Noronha

1. cooperação entre múltiplos robôs. 2. exploração de ambientes. 3. leilões.

CDU 519.6*82.9 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Técnicas de leilão aplicadas à coordenação de múltiplos robôs em missões de exploração de ambientes

RODOLFO CARNEIRO CAVALCANTE

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. LUIZ CHAIMOWICZ - Orientador
Departamento de Ciência da Computação - UFMG

PROF. THIAGO FERREIRA DE NORONHA - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. MARIO FERNANDO MONTENEGRO CAMPOS
Departamento de Ciência da Computação - UFMG

PROF. SEBASTIAN ALBERTO URRUTIA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 27 de fevereiro de 2013.

A José Carneiro e Maria Elza.

Agradecimentos

A Deus, porque tenho a absoluta certeza que tudo devemos a Ele. Agradeço por Ele estar sempre presente na minha vida. Tenho certeza que a Sua presença contribui diretamente na minha caminhada.

Aos meus pais, José Carneiro e Maria Elza, pela educação que me deram e por sempre me apoiarem em todas as minhas batalhas. A meu irmão Ricardo, que me apoiou bastante durante meu mestrado, e a minha irmã Mariana por todos os momentos de alegria e descontração.

Aos professores Luiz Chaimowicz e Thiago Noronha, por terem contribuído diretamente com minhas pesquisas e, sobretudo, pelo que eu pude aprender com eles.

Ao professor Mário e todos os amigos do Verlab, que sempre estiveram à disposição para ajudar no que fosse necessário.

À banca pelas contribuições para este trabalho, ao PPGCC e à Capes pelo apoio financeiro.

Aos amigos da Los-Computeros Alan Castro, Clayson Celes, Felipe Ávila, Rickson Guidolini, Rogério Fonteles e Wallace Favoreto e aos amigos que fiz em Belo Horizonte, por todos os momentos de descontração. Com certeza foram de grande ajuda para o desenvolvimento deste trabalho.

Aos meus antigos professores da UFAL Arapiraca, em especial ao professor Elthon Oliveira, pelo incentivo e força que deram para que eu conseguisse chegar até aqui.

Ao meu tio Léo, que nos deixou no início do mestrado. Você sempre será lembrado como aquela pessoa honesta e alegre e nunca sairá de nossos corações.

A minha amada namorada Joyce, que está sempre ao meu lado e me dá forças para buscar o sucesso. Ela sempre está em meu pensamento.

Resumo

A exploração de ambientes utilizando múltiplos robôs autônomos é considerada um dos problemas fundamentais da robótica móvel. Um problema de exploração típico é aquele onde os robôs possuem um modelo do ambiente e precisam visitar um conjunto de pontos ou alvos virtuais neste ambiente de forma a executar alguma operação ou apenas coletar informações nestes pontos. A utilização de um time de robôs cooperativos para executar esta tarefa de exploração apresenta várias vantagens como rapidez na missão e robustez do sistema de exploração em comparação a sistemas com único robô. No entanto, atribuir robôs para a visita dos alvos de forma a minimizar os custos da missão é um problema NP-Difícil. Leilões de único item e leilões sequenciais de único item são mecanismos aproximados que têm sido muito utilizados para resolver o problema de coordenação em missões de exploração. No entanto, dadas as limitações destes leilões, estes métodos acabam produzindo alocações com baixa qualidade. Neste trabalho, foi investigado a aplicação dos leilões combinatórios ao problema de coordenação de múltiplos robôs na exploração de ambientes. Leilões combinatórios são mais complexos, porém são capazes de prover soluções de melhor qualidade. Foi proposto o uso de alguns algoritmos de forma a tornar o leilão combinatório um processo de alocação polinomial. Tais algoritmos foram implementados e diversos experimentos foram realizados de forma a comparar a qualidade das soluções providas pelos leilões combinatórios com aquelas providas pelos leilões de único item e leilões sequenciais. Os resultados mostraram que alguns destes algoritmos utilizados nos leilões combinatórios apresentaram resultados melhores que os leilões de único item e sequenciais.

Palavras-chave: cooperação entre múltiplos robôs, exploração de ambientes, leilões.

Abstract

Exploring an environment using multiple autonomous robots is one of the fundamental problems in mobile robotics. A typical exploration problem is that in which robots have a model of the environment and they need to visit a set of virtual targets in this environment in order to perform some operation or just collect informations in these points. Using a team of cooperative robots in exploration tasks presents several advantages such as more speed in the mission and more robustness when compared with single robot systems. However, assign robots to visit targets in order to minimize the mission costs is a NP-Hard problem. Single-item Auctions and Sequential Single-Item Auctions are approximated mechanisms widely used in solving the coordination problem in exploration missions. However, given the limitations of these kind of auctions, these mechanisms provide low quality solutions. In the present work, it was investigated the use of combinatorial auctions in exploration of environments. It was proposed the use of some algorithms in order to make the combinatorial auction a polynomial allocation process. These algorithms were implemented and several experiments were performed in order to compare the quality provided by combinatorial auction with those provided by single-item auctions and sequential auctions. The results showed that some algorithms used in combinatorial auctions achieved better results than single-item and sequential auctions.

Keywords: multi-robot cooperation, exploration of environments, auctions.

Lista de Figuras

Sumário

Capítulo 1

Introdução

Durante as últimas décadas, os robôs têm se tornado cada vez mais presentes na vida humana, sendo utilizados em uma vasta gama de aplicações. Atualmente, podemos ver robôs sendo utilizados desde em aplicações simples e repetitivas, como limpeza de pisos e transporte de objetos [Matarić et al., 1995], até em aplicações complexas e críticas, como manufatura e construção, assistência médica, vigilância e monitoramento e na exploração de ambientes [Dias et al., 2006].

Diversas pesquisas têm se dedicado a investigar mecanismos que possibilitem a um grupo de robôs autônomos trabalhar de forma cooperativa como um time. Existem diversas vantagens no uso de múltiplos robôs em detrimento a sistemas com único robô, tais como: (a) simplicidade no projeto dos robôs, pois um único robô não precisa possuir todas as especialidades necessárias para resolver um problema; (b) rapidez na resolução de múltiplas tarefas, já que os robôs podem executá-las em paralelo; (c) robustez e tolerância a falhas individuais, pois se alguns robôs falharem, os demais ainda podem concluir a missão; (d) o fato de que um sistema com múltiplos robôs é capaz de executar tarefas que não podem ser executadas por um único robô, tais como carregamento de objetos grandes ou tarefas geograficamente distribuídas e que precisam ser executadas em paralelo [Chaimowicz et al., 2001; Gerkey & Matarić, 2002].

No entanto, apesar de todas as vantagens advindas da cooperação, a multiplicidade de robôs introduz o problema de como coordenar os robôs de forma a evitar conflitos durante a missão e melhorar a utilização dos recursos do time. Diversas pesquisas têm investigado comportamentos cooperativos em várias áreas do conhecimento como biologia, ciências sociais, economia e teoria dos jogos, com o objetivo de desenvolver formas de aplicar esses comportamentos cooperativos em sociedades de agentes autônomos de forma a construir sistemas eficientes.

Dentre os vários problemas que a robótica cooperativa se dedica a resolver, o

problema de exploração tem recebido grande atenção da comunidade científica, sendo considerado um dos problemas fundamentais da robótica móvel [Fox et al., 2006]. É importante destacar que o problema de exploração de ambientes tem sido abordado de várias formas na literatura.

Um tipo de problema de exploração que tem sido muito investigada é conhecida como *mapeamento de ambientes desconhecidos*, onde um ou mais robôs precisam explorar um ambiente totalmente desconhecido e construir um mapa confiável desse ambiente [Burgard et al., 2000, 2005; Fox et al., 2006; Ko et al., 2003; Simmons et al., 2000; Yamauchi, 1998]. A Figura 1.1 ilustra exemplos da aplicação do problema de mapeamento. Em ambas as figuras, pode-se observar os resultado do mapeamento de dois ambientes, onde o mapa produzido diferencia os espaços livres, os obstáculos que formam corredores e passagens, e os espaços desconhecidos no ambiente mapeado.

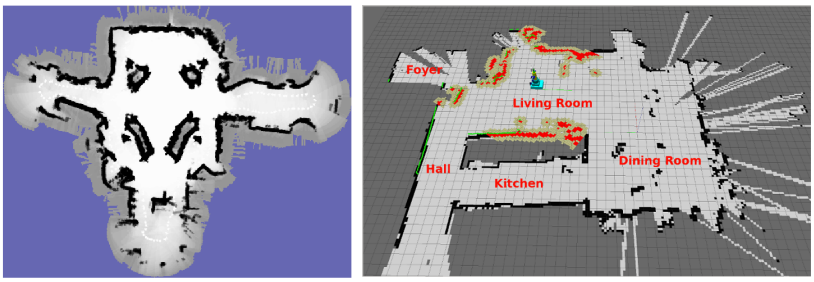

Figura 1.1. Exploração e mapeamento de ambientes desconhecidos [Goebel, 2006].

Um segundo tipo de problema de exploração, e que o presente trabalho se dedica a investigar, é conhecida como *observação de múltiplos alvos*, onde o ambiente é totalmente ou parcialmente conhecido, porém existem diversos pontos ou alvos virtuais que precisam ser visitados pelos robôs e alguma ação precisa ser executada nesses pontos [Berhault et al., 2003; Nanjanath & Gini, 2006; Zheng et al., 2006; Zlot & Stentz, 2005]. Existem várias aplicações onde encontramos esse tipo de problema de exploração, das quais podemos citar: sistemas vigilância e monitoramento, onde um ou mais robôs precisam visitar pontos de um ambiente conhecido de tempos em tempos de forma a identificar eventuais intrusos ou situações anormais; missões de busca e salvamento, onde os robôs conhecem a planta de um ambiente, como uma fábrica ou local onde ocorreu um desastre, e precisam procurar por vítimas em determinados pontos desse ambiente; coleta de dados em redes de sensores, onde os robôs precisam visitar regiões nas quais estão localizados os sensores e coletar os dados destes sensores [Soares et al., 2007]; entre outras aplicações. A Figura 1.2 mostra exemplos da aplicação deste tipo de problema. Na Figura 1.2(a) pode-se observar a superfície de um planeta já mapeada,

onde os robôs devem visitar um conjunto de alvos virtuais e realizar um sensoreamento ou coletar objetos nesses pontos. A Figura 1.2(b) ilustra a planta de um hospital, onde os robôs precisam visitar alguns pontos a procura de possíveis vítimas que precisam ser resgatadas.

Figura 1.2. Exploração de ambientes conhecidos [Dias et al., 2006].

Métodos de negociação baseado em leilões são mecanismos distribuídos capazes de prover soluções aproximadas para o problema de coordenação de múltiplos robôs, tendo sido muito utilizados na literatura para resolver problemas de exploração do tipo observação de múltiplos alvos [Dias et al., 2006; Zlot et al., 2002]. Leilões de Único Item são o tipo de leilão mais simples, onde um item é leiloado por vez sem considerar os itens que já foram leiloados anteriormente ou aqueles que ainda serão leiloados. Leilões Sequenciais de Único Item são uma extensão dos Leilões de único item onde os itens são leiloados segundo uma sequência e os robôs consideram os itens já leiloados anteriormente durante a computação do lance para um novo item. Leilões Combinatórios são um tipo mais complexo de leilão, que permite que os robôs ofereçam lances para combinações dos itens a serem leiloados, permitindo uma boa alocação de itens para os robôs, já que se consideram as relações de complementariedade ou sinergia entre os itens leiloados.

1.1 Problema e Objetivos

Sejam um conjunto de robôs, $R = \{r_1, r_2, ..., r_n\}$, um conjunto de alvos, $T = \{t_1, t_2, ..., t_m\}$, o problema de coordenação de múltiplos robôs em missões de observação de múltiplos alvos consiste em encontrar uma atribuição de robôs para a visita dos alvos que minimize a soma das distâncias viajadas pelos robôs durante a missão e consequentemente o custo desta missão. Cada alvo deve ser visitado uma única vez e os robôs precisam retornar as suas posições iniciais para que possam ser posteriormente

recolhidos por uma equipe humana. Encontrar a atribuição ótima é um problema NP-Difícil [Lagoudakis et al., 2004].

Leilões de Único Item e Leilões Sequenciais têm sido muito empregados na robótica para o problema de exploração de ambientes conhecidos. No entanto, dada a simplicidade destes leilões, estes métodos acabam produzindo alocações de baixa qualidade. Os leilões combinatórios, por outro lado, são capazes de prover soluções mais próximas da melhor alocação possível, no entanto, este tipo de leilão não têm sido muito empregado para resolver o problema de coordenação dos robôs durante a exploração de ambientes conhecidos devido a sua alta complexidade computacional.

Neste trabalho é investigado como resolver o problema de alocação de alvos para robôs durante uma missão de observação de múltiplos alvos de forma aproximada, através dos leilões. Foram implementados e avaliamos os leilões de único item e dois algoritmos utilizados para a geração de lances durante os leilões sequenciais. A maior contribuição deste trabalho encontra-se no estudo dos leilões combinatórios aplicados a este problema. Neste sentido, foram implementadas duas estratégias para tornar o leilão combinatório um processo polinomial, permitindo que este mecanismo possa ser aplicado ao problema de exploração. Alguns algoritmos foram implementados e avaliados em combinação com as estratégias de leilão combinatório. Diversos experimentos foram realizados de forma a comparar a qualidade das soluções providas pelos leilões combinatórios com os leilões de único item e sequenciais.

1.2 Organização do Trabalho

O restante desta dissertação está organizado da seguinte forma: o Capítulo 2 apresenta uma revisão da literatura relacionada ao tema de cooperação em sistemas com múltiplos robôs e a exploração de ambientes. No Capítulo 3, são discutidos os mecanismos de cooperação investigados neste trabalho, bem como os algoritmos utilizados. No Capítulo 4, são apresentados e discutidos os experimentos realizados e os resultados dos mecanismos de cooperação implementados. Por fim, o Capítulo 5 apresenta as conclusões deste trabalho e as possíveis extensões da presente pesquisa.

Capítulo 2

Revisão Bibliográfica

Neste capítulo descrevemos os principais mecanismos propostos para resolver o problema de exploração em suas diferentes facetas. Na Seção 2.1, é introduzida a cooperação entre múltiplos robôs em geral. A Seção 2.2 descreve o problema de exploração de ambientes. A Seção 2.3 apresenta uma taxonomia proposta neste trabalho com o objetivo de organizar a literatura. Por fim, na Seção 2.4 são discutidos os principais mecanismos de cooperação aplicados a exploração de ambientes propostos na literatura.

2.1 Cooperação em Sistemas com Múltiplos Robôs

A construção de sistemas com múltiplos robôs tem recebido uma crescente atenção por parte da comunidade de pesquisa em robótica. Durante os últimos anos, vários pesquisadores têm investigado como desenvolver mecanismos onde um grupo de robôs seja capaz de trabalhar de forma cooperativa como um time [Shiroma & Campos, 2009]. Existem várias vantagens de se utilizar sistemas com múltiplos robôs em detrimento daqueles com robô único. Um time de robôs é capaz de resolver problemas que envolvem tarefas complexas e que não podem ser resolvidas por um único robô seja devido a restrições físicas ou de espaço, seja devido a necessidade de diversas especialidades e habilidades que são caras ou difíceis de embutir em um único robô [Zlot et al., 2002; Kaleci et al., 2010].

Um sistema com múltiplos robôs pode ser mais fácil de projetar, mais barato e mais flexível que um sistema com único robô [Chaimowicz et al., 2001; Thomas & Williams, 2009]. Mais fácil de projetar porque um sistema com múltiplos robôs pode ser visto como uma sociedade virtual capaz de apresentar comportamentos como cooperação e colaboração, e assim os problemas podem ser resolvidos de forma análoga ao mundo real, por meio de divisão de trabalho e de capacidades entre os

agentes do time [Cavalcante et al., 2012a]. Pode ser mais barato construir um sistema composto por vários robôs simples e com diferentes habilidades do que construir um único robô complexo dotado de várias ferramentas e habilidades [Dudek et al., 2002]. Além disso, um sistema com único robô está limitado pelas restrições espaciais, não podendo realizar tarefas distribuídas geograficamente que precisam ser executadas em paralelo. Por fim, um sistema com um único robô apresenta um ponto central de falhas, que é o próprio robô, enquanto que um sistema com vários robôs é tolerante a falhas individuais, de forma que se alguns robôs falham, ainda há possibilidades de a tarefa ser completada.

Mecanismos de cooperação entre múltiplos robôs se dedicam a resolver o problema de como coordenar um conjunto de robôs que compartilham os mesmos objetivos de forma a executar um conjunto de tarefas de forma eficiente sem que hajam conflitos entre os robôs [Cao et al., 1997]. Na literatura, duas classes principais de comportamentos cooperativos podem ser encontradas: *comportamento emergente* e *comportamento intencional*.

Comportamento emergente consiste em um tipo de cooperação implícita entre os robôs. Neste tipo de cooperação, um robô não coordena suas ações com os outros através de comunicação direta e explícita, mas sim por meio da observação da presença, ausência, comportamentos e resultados das ações de outros robôs. Geralmente, esse tipo de cooperação é construído para resolver problemas de um domínio de aplicação específico, não podendo ser diretamente aplicado a outro problema que não aquele para o qual o sistema cooperativo foi projetado.

Comportamento intencional, por outro lado, é aquele onde os indivíduos do grupo se comunicam diretamente com seus companheiros. Mecanismos que se baseiam em cooperação intencional geralmente são capazes de prover comportamentos cooperativos mais adequados a resolução de problemas do mundo real, permitindo até que os robôs possam cooperar com times de seres humanos [Gerkey & Matarić, 2004]. No entanto, a cooperação explícita introduz alguma complexidade na construção dos sistemas, dado que os robôs precisam de hardware e software dedicado para comunicação, além da introdução de outros problemas relacionados com a comunicação, como confiabilidade e segurança [Farinelli et al., 2004].

Um importante desafio para a construção de sistemas com múltiplos robôs é como permitir que os robôs distribuam as tarefas e coordenem suas ações de forma a maximizar a utilidade do time e evitar conflitos. Este problema, definido como Problema de Alocação de Tarefas para Múltiplos Robôs (do inglês, *Multi-Robot Task Allocation Problem − MRTA*) [Gerkey & Matarić, 2003], é um problema de otimização que consiste em encontrar uma atribuição de tarefas a robôs dentro do espaço de possíveis

atribuições de forma a maximizar (ou minimizar) uma função objetivo. Diversos mecanismos têm sido propostos nos últimos anos para resolver este problema de alocação. Campbell & Wu [2011] recentemente fizeram um exame de literatura que descreve os principais mecanismos aplicados ao problema de alocação de tarefas para múltiplos agentes.

2.2 Exploração de Ambientes

No contexto da robótica cooperativa, durante as últimas décadas, diversos pesquisadores têm investigado como usar robôs móveis autônomos para explorar ambientes perigosos ou inacessíveis ao homem, como cavernas e minas, zonas de desastre nuclear, territórios hostis ou mesmo a superfície de outros planetas. Nestes tipos de ambientes hostis, o uso de robôs autônomos pode tanto viabilizar a exploração como evitar o risco da perda de vidas humanas.

O conceito de exploração de ambientes utilizando robôs móveis está associado a diferentes tipos de abordagens, sendo definido basicamente como o processo de descoberta de informações em um ambiente por meio de robôs móveis [Basilico & Amigoni, 2011]. Na literatura, podemos encontrar duas abordagens principais para a exploração de ambientes utilizando robôs móveis: *mapeamento de ambientes desconhecidos* e *observação de múltiplos alvos*.

No contexto da primeira classe de abordagens, Yamauchi [1997] define que "mapeamento de ambientes é o ato de se mover em um ambiente desconhecido à medida que se constrói um mapa que poderá ser utilizado para subsequente navegação neste ambiente". Estratégias de mapeamento de ambientes desconhecidos têm o objetivo de construir um mapa confiável do ambiente explorado, onde esse mapa contém a localização de obstáculos e espaços livres ou mesmo características distintas deste ambiente com o mínimo possível de incerteza.

A observação de múltiplos alvos, por outro lado, é a abordagem onde os robôs devem visitar diversos alvos virtuais no ambiente com o objetivo de executar alguma tarefa ou apenas coletar informações nesses pontos. Nesse tipo de abordagem, os alvos podem ser definidos por um usuário externo, como uma equipe de pesquisadores que tem interesse na exploração ou esses pontos podem ser definidos pelos próprios robôs durante a execução da missão de acordo com o objetivo da exploração. A observação de múltiplos alvos é muito utilizada em aplicações de limpeza de pisos, vigilância e monitoramento e em missões de busca e salvamento [Berhault et al., 2003].

Diversas abordagens para coordenação de grupos de robôs em tarefas de explo-

ração foram propostos nas últimas décadas. No entanto, esses trabalhos diferem em diversos aspectos, como o conhecimento que os robôs possuem do ambiente, a estratégia de geração dos alvos ou o mecanismo de coordenação propriamente dito. Na próxima seção são discutidos os principais trabalhos propostos para o problema de exploração.

2.3 Taxonomia

De forma a organizar a literatura, nós propomos uma taxonomia para classificar os trabalhos de cooperação entre múltiplos robôs em tarefas de exploração de ambientes. Esta taxonomia é ilustrada na Figura 2.1. Nas próximas subseções serão descritos os conceitos que compõem esta taxonomia.

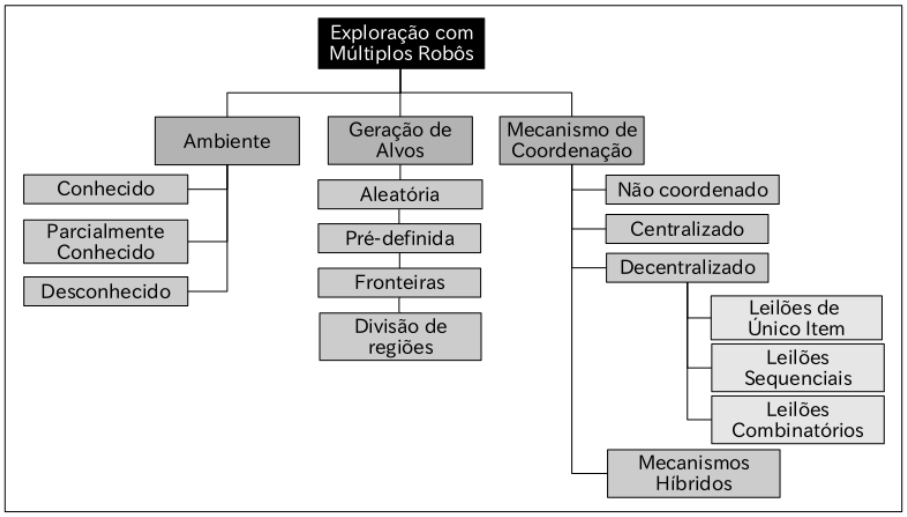

Figura 2.1. Taxonomia para classificar os trabalhos em exploração.

2.3.1 Conhecimento do Ambiente

O conhecimento do ambiente é essencial para o sucesso de uma missão de exploração utilizando múltiplos robôs. Na taxonomia proposta neste trabalho, pode-se ver que os ambientes para os quais a exploração é aplicada podem ser classificados como: (a) *ambientes conhecidos*, onde os robôs possuem um modelo do ambiente e precisam visitar algumas áreas de interesse nesses ambientes, (b) *ambientes parcialmente conhecidos*, onde os robôs possuem um modelo parcial do ambiente, porém este modelo pode não identificar a presença de obstáculos ou outros eventos capazes de impedir a livre circulação dos robôs no ambiente modelado, (c) *ambientes desconhecidos*, onde os robôs não têm nenhum conhecimento prévio do ambiente a ser explorado.

2.3.1.1 Ambientes Conhecidos

Missões de exploração em ambientes previamente conhecidos não precisam lidar com incertezas no ambiente, pois os robôs possuem um mapa ou outras informações que denotam obstáculos e espaços livres neste ambiente. Um exemplo de aplicação com essa configuração é um sistema de vigilância e monitoramento, onde os robôs têm que visitar pontos pré-definidos no ambiente de tempos em tempos. Em aplicações nestes tipos de ambientes, os robôs utilizam alguma forma de localização, como posição global, estimativa probabilística de posição, uso de *beacons*, uso de informações topológicas, entre outras. Geralmente, missões de exploração em ambientes conhecidos pertencem a classe das abordagens de observação de múltiplos alvos.

Alguns trabalhos na literatura definem a exploração de ambientes conhecidos como o problema de Roteamento de Múltiplos Robôs [Ekici et al., 2009; Zheng et al., 2006; Zheng & Koenig, 2009b]. Lagoudakis et al. [2004] demonstraram que este é um problema NP-Difícil. Sendo assim, não são conhecidos algoritmos capazes de encontrar a melhor solução para o problema em tempo polinomial. Para instâncias pequenas do problema, a solução ótima pode ser obtida através de Programação Linear Inteira [Ekici et al., 2009]. Diversas heurísticas e algoritmos aproximados têm sido propostos para resolver grandes instâncias do problema. O presente trabalho se dedica a investigação de heurísticas baseadas em técnicas de leilão para resolver esse problema de forma distribuída.

2.3.1.2 Ambientes Parcialmente Conhecidos

Ambientes parcialmente conhecidos são aqueles onde os robôs têm um modelo incompleto do ambiente e, durante a missão, podem encontrar obstáculos ou outros eventos não esperados que os impeçam de se deslocar livremente no ambiente. Podemos citar como exemplo de ambiente parcialmente conhecido uma fábrica que sofreu algum tipo de desastre. A planta dessa fábrica é um modelo parcial do ambiente, pois corredores, portas e espaços livres descritos no mapa da fábrica podem estar bloqueados com destroços e entulhos.

O problema de coordenação em ambientes parcialmente conhecidos pode ser resolvido com técnicas similares àquelas aplicadas a ambientes previamente conhecidos. Pontos de interesse são definidos por um operador externo e os robôs devem ser alocados à visita desses pontos de forma a maximizar (ou minimizar) alguma função de otimização. O problema de alocação nesses tipos de ambientes é mais complexo do que aquele em ambientes previamente conhecidos, pois os obstáculos não esperados podem impedir que os robôs completem a missão [Nanjanath & Gini, 2006]. Uma forma de

tratar esse problema é permitir a realocação de alvos sempre que um robô obtém novas informações do ambiente e percebe que não poderá visitar um determinado alvo. Alguns trabalhos na literatura propuseram o uso de leilões com realocação de alvos durante a missão para lidar com esse problema [Nanjanath & Gini, 2006; Koenig et al., 2006].

2.3.1.3 Ambientes Desconhecidos

Nesta última classe de ambientes, os robôs não possuem nenhuma informação *a priori* do ambiente. Um exemplo de aplicação com esse tipo de configuração é uma missão de exploração planetária. Basicamente, a função do time de robôs nesses ambientes é cobrir a maior região possível do ambiente e construir um mapa confiável desse ambiente. Geralmente, abordagens para resolver esse tipo de problema apresentam uma estratégia gulosa de exploração, onde os robôs tentam coletar o máximo de informação possível em menor tempo, compartilhando os mapas individuais com os outros robôs do time sempre que possível [Burgard et al., 2000, 2005; Fox et al., 2006; Ko et al., 2003; Marjovi et al., 2009; Simmons et al., 2000].

2.3.2 Estratégia de Geração de Alvos

Uma tarefa de exploração pode ser definida como a ação de se mover em um ambiente enquanto se coleta informações de interesse nesse ambiente. Segundo Yamauchi [1997], o maior desafio em uma tarefa de exploração é decidir para onde os robôs devem se mover de forma a ganhar o máximo de informação possível com menor tempo ou custo. Duas questões principais estão envolvidas nessa decisão: (a) como gerar alvos virtuais a seren visitado pelos robôs e (b) qual critério deve guiar os robôs na escolha dos alvos a serem visitados. De acordo com a taxonomia proposta neste trabalho, existem quatro estratégias principais de geração de alvos, que serão descritas a seguir.

Uma das primeiras estratégias de geração de alvos propostas na literatura é aquela baseada no conceito de células de fronteira [Berhault et al., 2003; Ekici et al., 2009; Koenig et al., 2006; Nanjanath & Gini, 2006; Sariel et al., 2009; Zhang et al., 2010; Zheng & Koenig, 2009a,b; Zheng et al., 2006]. Esta estratégia consiste em guiar os robôs para regiões que estão na fronteira entre os espaços já explorados e aqueles ainda não explorados no ambiente. Esta estratégia gulosa é muito utilizada na exploração de ambientes desconhecidos, onde os robôs precisam construir um mapa do ambiente. Geralmente, um mapa representado como uma grade de ocupação é compartilhado pelos robôs, e os robôs continuamente enviam atualizações em seus mapas locais para os outros, informando sobre espaços livres, ocupados ou desconhecidos no ambiente. Através

desse mapa compartilhado, robôs podem identificar regiões de fronteira e coordenar a missão. O critério usado para escolha dos alvos a serem visitados é a maximização do ganho de informação em menos tempo.

Uma estratégia simples de geração de alvos é a estratégia aleatória, onde os robôs elegem pontos aleatórios do ambiente para visitar, descartando aqueles que caem em regiões já visitadas [Zlot et al., 2002]. Wagner et al. [1998] demonstraram que, mesmo sendo um método simples, esta estratégia apresenta bons resultados em missões de exploração de ambientes desconhecidos.

Uma estratégia mais sofisticada para geração de alvos é aquela baseada na segmentação do ambiente. Haumann et al. [2010] propuseram uma abordagem que utiliza Voronoi para particionar o ambiente em várias células de Voronoi, alocando os robôs para a exploração dessas células. Em [Wurm et al., 2008], um grafo de Voronoi é construído sobre as partes já exploradas do ambiente, permitindo a identificação de nós do grafo que levam a corredores e outras passagens para locais ainda não explorados. Os robôs são então atribuídos à visita desses nós. Estratégias de segmentação do ambiente apresentam a vantagem de distribuir eficientemente os robôs pelo ambiente, evitando tanto interferências entre as ações dos robôs como exploração repetida, otimizando o uso de múltiplos robôs na exploração.

Uma outra forma de geração de alvos descrita na taxonomia proposta consiste em delegar a um usuário externo a definição de quais pontos do ambiente devem ser visitados pelos robôs [Berhault et al., 2003; Ekici et al., 2009; Koenig et al., 2006; Nanjanath & Gini, 2006; Sariel et al., 2009; Zhang et al., 2010; Zheng & Koenig, 2009a,b; Zheng et al., 2006]. Neste modelo, o ambiente é totalmente ou parcialmente conhecido e os alvos representam pontos de interesse no ambiente. Critérios como minimização da soma das distâncias viajadas ou tempo de missão são utilizados para guiar a exploração nessas abordagens.

2.4 Mecanismos de Cooperação e Exploração

As diversas abordagens de exploração de ambientes descritas anteriormente divergem em vários aspectos, tais como no conhecimento prévio que os robôs têm do ambiente, nas estratégias de geração de alvos a serem visitados e no critério utilizado para escolha dos alvos. Consequentemente, diversos mecanismos foram propostos na literatura, de forma a resolver as diversas modalidades de problemas de exploração, como pode ser percebido por meio da taxonomia proposta.

Os primeiros mecanismos propostos se concentraram em resolver o problema de

mapeamento de ambientes utilizando apenas um único robô. No entanto, percebeu-se que um time de robôs pode executar essas tarefas em paralelo, compondo um sistema de exploração mais rápido e mais robusto a falhas individuais. O maior desafio da utilização de múltiplos robôs reside em como coordenar esses robôs de forma a maximizar a performance do sistema e evitar conflitos na exploração.

A seguir, são discutidos os principais tipos de mecanismos de coordenação aplicados a exploração de ambientes, desde os sistemas de exploração com único robô até os sistemas que utilizam mecanismos complexos e que utilizam diversos critérios para guiar a missão. A discussão se inicia com os sistemas com único robô. Em seguida, são descritos os mecanismos de exploração com múltiplos robôs porém sem coordenação. A discussão segue abordando os mecanismos centralizados e por fim os mecanismos descentralizados.

2.4.1 Exploração com Único Robô

As primeiras abordagens de exploração de ambientes utilizando robôs móveis consistiam em um sistema com único robô e eram empregadas na construção de mapas de ambientes totalmente desconhecidos. O problema de localização e mapeamento simultâneos (*Simultaneous Localization and Mapping* – SLAM) [Dissanayake et al., 2001] é um dos métodos de mapeamento mais conhecidos na literatura. Uma abordagem típica de SLAM consiste em um único robô que navega pelo ambiente utilizando seus sensores para construir um modelo confiável desse ambiente ao passo que se localiza no mapa construído [Colares & Chaimowicz, 2012].

Uma das principais preocupações em um mecanismo de exploração que utiliza um único robô consiste na análise dos dados coletados pelos sensores de forma a construir um mapa confiável do ambiente [Kuipers & Byun, 1991]. Além disso, é importante definir como este mapa será representado. Na literatura existem dois paradigmas principais empregados para modelagem de ambientes: mapas métricos em grade e mapas topológicos [Thrun & Bücken, 1996]. Mapas em grade, também chamados de grades de ocupação são uma forma de representar o ambiente como uma matriz na qual as células estão associadas a uma probabilidade de ocupação, onde essa probabilidade indica se a célula está livre, ocupada ou se é desconhecida [Elfes, 1989]. Mapas topológicos, por outro lado, são mapas que representam o ambiente como um grafo de marcos interconectados. Um marco é um objeto ou local distinto que pode ser utilizado como um ponto de referência no ambiente, como um corredor ou uma porta, por exemplo.

2.4.2 Mecanismos Não Coordenados

Mecanismos com múltiplos robôs sem coordenação são uma extensão dos sistemas de exploração com único robô. Neste tipo de sistema de exploração, vários robôs são utilizados para explorar o ambiente, porém os robôs executam suas ações em paralelo e de forma independente dos demais. Yamauchi [1998] utilizou um time de robôs não coordenados para construir um mapa métrico de um ambiente desconhecido. Cada robô mantém um mapa de ocupação que compartilha com outros robôs sempre que possível. A exploração é baseada na estratégia de fronteiras e os robôs dão prioridade a visita das fronteiras mais próximas de suas posições.

A principal vantagem da utilização de mecanismos de exploração não coordenados é a capacidade de se construir mapas mais confiáveis. Dado que cada robô constrói seu mapa individualmente e que muitas áreas são exploradas várias vezes por diferentes robôs, os mapas individuais podem ser combinados eliminando-se erros de sensoriamento individuais, produzindo mapas com menor incerteza. No entanto, a exploração pode não ser eficiente, pois dado que os robôs navegam independentemente, várias áreas são exploradas de forma redundante por diferentes robôs, o que resulta em maior consumo de tempo e de energia durante a missão. A ausência de coordenação pode também causar conflitos durante a execução da missão.

2.4.3 Mecanismos Centralizados

Este grupo de mecanismos de coordenação consiste naqueles mecanismos baseados em uma entidade central, o planejador, responsável por decidir para onde os robôs devem se mover durante a missão. Mecanismos centralizados podem ser aplicados a qualquer tipo de ambiente e utilizar qualquer estratégia de exploração. Brumitt & Stentz [1996] utilizaram um mecanismo centralizado para coordenação de times de robôs para explorar ambientes parcialmente conhecidos. O planejador central era responsável por atribuir os robôs para a visita de um conjunto de alvos pré-definidos com base no critério de minimização de distâncias viajadas.

Alguns trabalhos empregaram mecanismos de coordenação centralizados para o problema de exploração onde o ambiente é totalmente desconhecido e a estratégia de exploração é baseada na visita de regiões de fronteira [Burgard et al., 2000, 2005; Fox et al., 2006; Ko et al., 2003; Marjovi et al., 2009]. Nestes trabalhos, um planejador central é responsável por receber atualizações dos mapas individuais de cada robô e combinar essas informações em um mapa global. Este mapa global contém informações sobre as coordenadas dos robôs e também sobre as células de fronteira a serem visitadas. O planejador central utiliza as informações do mapa global para atribuir os robôs a

visita dos alvos. Esta atribuição é baseada em uma função de utilidade que considera o custo para um robô visitar um alvo e a quantidade de informação que se espera que o robô adquira quando alcançar aquele alvo. Essa é uma função de utilidade gulosa que guia a missão com base no critério de ganhar o máximo de informação com o menor tempo possível.

Embora mecanismos centralizados tenham sido muito estudados na literatura, esta técnica de coordenação não é ideal quando há um grande número de robôs no sistema [Dias et al., 2006]. Encontrar a melhor atribuição de robôs para a visita de alvos é um problema combinatório NP-Difícil [Lagoudakis et al., 2004]. Além da complexidade computacional, outros problemas são característicos de abordagens de coordenação centralizadas. É preciso que todos os robôs sejam capazes de se comunicar com a entidade central para que a coordenação seja efetiva. Além disso, a entidade central se apresenta tanto como um gargalo como um ponto central de falhas para o sistema.

2.4.4 Mecanismos Decentralizados

A próxima classe de mecanismos de coordenação a ser discutida inclui os mecanismos decentralizados, aqueles que não apresentam um agente central responsável pelo planejamento da missão. Estes mecanismos não apresentam alguns dos problemas que existem nos mecanismos centralizados, como um ponto central de falhas. Além disso, são mais rápidos, mais flexíveis e mais escaláveis. No entanto, devido ao seu caráter distribuído, onde os robôs tomam decisões com base em conhecimento local, a atribuição final pode não ser a ótima.

Um importante mecanismo para coordenação decentralizada de robôs em tarefas de exploração é aquele baseada em técnicas de mercado, como os leilões. Devido a sua simplicidade e facilidade de implementação, os leilões têm sido bastante utilizados na literatura de robótica [Lagoudakis et al., 2005; Zlot et al., 2002]. Nestas abordagens, os robôs competem pela execução das tarefas, tentando maximizar seus resultados individuais.

Em um mecanismo baseado em leilão, os robôs agem como licitantes ou compradores e os alvos a serem visitados são os itens negociados no leilão. Cada alvo tem um custo associado à sua execução e um robô dá lance para um alvo como base na sua aptidão ou utilidade para a visita do alvo. O leilão precisa de uma entidade que atue como leiloeiro, que é a entidade responsável por receber os lances dos robôs e decidir quem são os vencedores dos itens. Assim, o sistema pode ter um robô dedicado para agir como leiloeiro, podendo ser um mesmo robô durante todo o leilão, ou ter um robô

diferente a cada fase do leilão. O leilão pode também ser completamente distribuído, onde todo robô age tanto como comprador como leiloeiro. Neste caso, cada robô envia seu lance para todos os outros e após receber os lances de todos os robôs, cada robô individual decide quem foi o vencedor do item oferecido [Lagoudakis et al., 2005].

Embora os robôs sejam egoístas, a maximização das recompensas individuais resulta na maximização da recompensa total do time. Leilões são eficientes tanto em comunicação, dado que as informações trocadas pelos robôs são lances numéricos, quanto em computação, dado que cada robô computa seu lance em paralelo [Gerkey & Matarić, 2002; Koenig et al., 2010]. Leilões são ainda tolerantes a falhas individuais, pois se um robô falha durante o leilão, ele é simplesmente ignorado. Nesta sessão, serão examinados os Leilões de Único Item, os Leilões Sequenciais de Único Item e os Leilões Combinatórios.

2.4.4.1 Leilões de Único Item

Neste tipo de leilão, um alvo é oferecido em cada etapa do leilão e os robôs computam lance para um item independentemente dos alvos que já foram leiloados em fases anteriores ou dos alvos que ainda irão ser oferecidos em fases subsequentes. Alguns trabalhos propostos na literatura utilizaram leilões de único item como mecanismo de coordenação dada a sua facilidade de implementação e baixo custo computacional requerido para computar lances.

Simmons et al. [2000] propuseram um dos primeiros trabalhos que empregam leilões de único item à tarefa de exploração. O objetivo da missão era construir um mapa de um ambiente desconhecido usando a estratégia de exploração baseada em fronteiras. Em tal abordagem, os robôs compartilham um mapa global que contém as coordenadas de todos os robôs e alvos de fronteira a serem visitados. Cada robô computa individualmente o custo para visitar os alvos com base neste mapa compartilhado e enviam seus lances para o leiloeiro.

Uma abordagem similar foi proposta por Zlot et al. [2002]. Um time de robôs deveria construir um mapa do ambiente minimizando o tempo da missão. Um robô gera um novo alvo no ambiente e tenta vender esse alvo para outros robôs que estejam no seu domínio de comunicação naquele momento. O robô anuncia o alvo e o preço de reserva do alvo gerado, ou seja, o preço mínimo de venda do alvo, que é o custo para que o próprio robô que anuncia o alvo visitá-lo. Os outros robôs dão lances com base em uma função de utilidade que considera o custo de visitar o alvo e a quantidade de informação que se espera obter naquele avo. Os autores avaliaram a qualidade do mecanismo utilizando três estratégias de geração de alvos: aleatória, baseada em fronteiras e por

divisão de espaço. Os resultados mostraram que as estratégias aleatórias e por divisão de espaço foram melhores em termos de custo e ganho de informação.

O trabalho proposto por Nanjanath & Gini [2006] investigou como utilizar leilões de único para o problema de exploração em ambientes parcialmente conhecidos e dinâmicos. Em sua abordagem, os robôs possuem um modelo do ambiente e os alvos a serem visitados são definidos por um operador externo. Em cada etapa do leilão, um alvo é oferecido e os robôs dão lances para os alvos com base no critério de minimização das distâncias viajadas. Quando, durante a execução da missão, um robô falha ou encontra um obstáculo não esperado que o impede de alcançar um alvo já alocado para ele, como uma porta fechada, por exemplo, o robô oferece o alvo para os outros robôs em um leilão. Esta abordagem é robusta a falhas individuais e é ideal para ambientes parcialmente conhecidos.

Os trabalhos propostos por Hanna [2005] e Spaan et al. [2010] se dedicaram a investigar o problema de alocação em missões de exploração onde há a presença de incertezas nos custos dos alvos. Em ambientes não conhecidos previamente, os robôs podem não saber o custo real relativo à visita de um alvo e por isso não são capazes de computar o lance para esse alvo durante o leilão, resultando em uma alocação com de alto custo. Em [Hanna, 2005], foi utilizado um Processo de Decisão de Markov (*Markov Decision Process* − MDP) para computar o custo esperado da execução dos alvos de forma probabilística. O custo esperado é utilizado para computar o lance para os alvos durante um leilão de único item. Em [Spaan et al., 2010], foi proposto o uso do Processo de Decisão de Markov Parcialmente Observável (*Partially Observable Markov Decision Process* − POMDP) para resolver o mesmo problema, porém para o caso onde os sensores dos robôs não são precisos e assim estes têm apenas uma estimativa de seu estado no ambiente.

2.4.4.2 Leilões Sequenciais de Único Item

Devido a sua simplicidade e baixa complexidade computacional, leilões de único item podem resultar em alocações sub-ótimas [Lagoudakis et al., 2004]. Em leilões onde os itens vendidos apresentam algum tipo de relação entre si, como no caso do leilão de alvos a serem visitados em uma missão de exploração, atribuir esses itens de forma independente pode não resultar em uma alocação de qualidade. Leilões sequenciais (*Sequential Single-Item Auctions* − SSI) são uma evolução dos leilões de único item que leva em consideração as *sinergias* entre os itens oferecidos no leilão. Neste trabalho o conceito de sinergia é definido como a propriedade que dois ou mais itens possuem quando apresentam maior benefício para o time quando alocados juntos para um mesmo

robô do que a soma de seus valores quando vendidos separadamente para diferentes robôs [Koenig et al., 2006].

Em um leilão do tipo SSI, um alvo é oferecido em cada fase do leilão, assim como no leilão de único item. No entanto, um robô considera o lance para o novo alvo oferecido com base nos alvos que ele já venceu em etapas anteriores do leilão, ao invés de considerar apenas o alvo oferecido. Esse método de leilão tende a alocar, para cada robô, apenas alvos que apresentam algum tipo de sinergia com os alvos já alocados anteriormente, melhorando a alocação final.

Lagoudakis et al. [2004] propuseram o uso de um mecanismo de coordenação baseado em um leilão SSI em missões de exploração onde o ambiente é conhecido e os alvos a serem visitados são pré-definidos. O problema de exploração é modelado como um grafo ponderado, onde os vértices correspondem à posição inicial dos robôs e as posições dos alvos. Os pesos das arestas correspondem ao custo de o robô navegar de um alvo a outro. Os robôs computam lances para os alvos oferecidos com base em um algoritmo para computar a Árvore Geradora Mínima (*Minimum Spanning Tree* − MST). Ao longo do leilão, cada robô tenta construir uma MST com os alvos oferecidos, tendo como raiz da árvore a sua posição inicial. Em cada etapa do leilão, um alvo é oferecido e cada robô oferece um lance para esse alvo igual ao custo de se adicionar esse novo alvo na sua MST. O vencedor do alvo leiloado é o robô que oferece o menor lance, ou seja, o robô que menos incrementa o custo de sua MST com a adição do novo alvo. Segundo os autores, este algoritmo para computação de lances oferece a garantia de que o custo da solução é no máximo duas vezes o custo da solução ótima.

Koenig et al. [2006] propuseram um outro mecanismo de coordenação baseado em leilões SSI para resolver o mesmo problema de observação de múltiplos alvos. Da mesma forma, o problema é modelado como um grafo ponderado tendo como vértices as posições iniciais dos robôs e as posições dos alvos. Neste caso, os robôs computam lances para os alvos oferecidos com base em uma heurística polinomial para resolver o Problema do Caixeiro Viajante (*Traveling Salesman Problem* − *TSP*). Ao longo do leilão, cada robô tenta construir um circuito de menor custo com os alvos oferecidos, tendo como ponto de partida a sua posição inicial. Em cada etapa do leilão, um alvo é oferecido e cada robô oferece um lance para esse alvo igual ao custo de se adicionar o alvo em seu circuito. O vencedor do alvo leiloado é o robô que oferece o menor lance, ou seja, o robô que menos incrementa o custo de seu circuito com a inserção do novo alvo. Os autores provaram que esse mecanismo provê uma solução que é 1.5 vezes a solução ótima e 2 vezes o custo da solução ótima no pior caso.

Em um trabalho preliminar relacionado a esta dissertação, foi proposto um mecanismo com o objetivo de minimizar o custo de uma missão exploração de múltiplos alvos

utilizando leilões SSI em ambientes conhecidos [Cavalcante et al., 2012b]. Nesse trabalho, investigou-se como um algoritmo de busca local poderia otimizar o roteamento dos robôs para a visita dos alvos alocados em um mecanismo de leilão sequencial. Após o leilão, cada robô possui um ciclo contendo os alvos alocados durante o leilão. Cada robô utiliza o algoritmo de busca local de forma a encontrar um ciclo de menor custo para visitar seus alvos. Os experimentos mostraram que a busca local após o leilão foi capaz de reduzir o custo da rota inicial provida pelo leilão SSI.

2.4.4.3 Leilões Combinatórios

Um tipo de leilão mais complexo, porém bastante usado como mecanismo de alocação de recursos, são os leilões combinatórios. Leilões combinatórios são aqueles onde os robôs podem dar lances para conjuntos ou pacotes de itens a serem leiloados, sendo um mecanismo de interesse em várias áreas do conhecimento, como economia, pesquisa operacional e ciência da computação [Cramton et al., 2005]. Leilões combinatórios podem resultar em alocações de maior qualidade que os mecanismos de leilão de único item e sequenciais, principalmente em situações onde os itens apresentam alta sinergia, ou seja, onde o valor de dois itens para um comprador é maior quando vendidos juntos do que a soma de seus valores quando estes itens são vendidos separadamente. Leilões combinatório são ideais em aplicações logísticas, alocação de rotas de ônibus e alocação de espectros de rádio, por exemplo [Sandholm, 2002].

No entanto, apesar das vantagens providas pelos mecanismos de alocação baseados em leilões combinatórios, existem dois problemas que podem inviabilizar o uso desse mecanismo em algumas aplicações. O primeiro problema é decidir quais itens um comprador irá combinar para oferecer um lance. Dado n itens a serem leiloados, existem $2^n - 1$ combinações possíveis para as quais os compradores podem formular lances, o que torna o leilão impraticável quando n é grande [Rothkopf et al., 1998]. O segundo problema é conhecido como *problema de determinação do vencedor*, que consiste em determinar quais são os lances vencedores dentre os $2^n - 1$ lances dados pelos robôs. Segundo Sandholm [2002], esse é um problema NP-Difícil .

Rothkopf et al. [1998] define que uma forma de resolver estes dois problemas é limitar a forma como os robôs escolhem os pacotes para dar lances. Existem várias estratégias para limitar os pacotes leiloados. Uma estratégia simples é limitar o número de itens permitidos em cada pacote. No entanto, se o número de itens for baixo, as sinergias entre os itens podem ser perdidas. Se esse número for alto, a determinação do vencedor pode se tornar intratável. Uma outra estratégia para o problema é organizar as possíveis combinações ou pacotes de itens segundo uma hierarquia de árvore, e

permitir que os robôs deem lances apenas em combinações representadas pelos nós dessa árvore. Esta solução, que será discutida em mais detalhes no Capítulo 3, limita o número de combinações de alvos que os robôs precisam computar lances, que se torna linear no número de nós da árvore. Se a aplicação permitir que os itens possam ser organizados em uma ordem sequencial, os agentes compradores podem usar uma outra estratégia, que consiste em dar lances para apenas em combinações de itens que obedecem a essa sequência. Com essa terceira estratégia, a determinação do vencedor pode ser computada em tempo polinomial [Rothkopf et al., 1998].

Na literatura específica de robótica, poucos trabalhos utilizaram o uso de leilões combinatórios para resolver o problema de coordenação de múltiplos robôs na exploração de ambientes.

Berhault et al. [2003] propuseram um trabalho que utiliza um leilão combinatório como mecanismo de coordenação em tarefas de exploração de ambientes parcialmente conhecidos onde os robôs devem visitar alvos pré-definidos. Em seu trabalho, algumas estratégias de escolha de pacotes são propostas. A primeira delas é baseada no número de alvos possíveis em cada pacote, onde os robôs dão lances para todas as combinações que contém um limitado número de alvos. Uma outra estratégia é baseada no algoritmo de corte máximo (*MaxCut Algorithm*), uma heurística que considera a posição dos alvos como nós de um grafo e recursivamente divide esse grafo em subgrafos menores. Os robôs consideram estes subgrafos como pacotes e oferecem lances para estes pacotes durante o leilão.

Zlot & Stentz [2005] propuseram o uso de um tipo de leilão combinatório aplicado ao problema de reconhecimento de áreas. Um grupo de robôs precisa executar um conjunto de tarefas denominadas complexas, que consistem em áreas de interesse no ambiente que precisam ser observadas a partir de alguns pontos ou alvos ao redor destas áreas, coletando informações em tais pontos. O mecanismo proposto utiliza uma estratégia de árvore de tarefas onde os níveis mais altos da árvore constituem descrições abstratas das áreas a serem exploradas, e os níveis mais baixos contêm os alvos individuais que devem ser visitados pelos robôs. A árvore é criada decompondo-se a área completa a ser explorada em um conjunto de áreas de interesse e estas áreas são novamente subdivididas em pontos de observação, que são os pontos a serem visitados pelos robôs. A árvore é oferecida no leilão e os robôs podem dar lances em um ou mais nós dessa árvore. Diferentes robôs podem vencer diferentes nós da árvore no leilão. Os resultados desse trabalho mostraram que essa estratégia de leilão apresenta soluções de alta qualidade para o problema estudado quando comparado com leilões que negociam somente pontos de observação ou somente as áreas de interesse.

2.4.5 Abordagens Híbridas

Além dos mecanismos de coordenação até aqui descritos, alguns outros trabalhos se dedicaram a construção de sistemas com múltiplos robôs aplicados a tarefas de exploração que combinam mais de um mecanismo de coordenação. Tais mecanismos são descritos a seguir.

Zheng & Koenig [2009a] desenvolveram um mecanismo híbrido de coordenação com o objetivo de melhorar a qualidade da solução provida por um mecanismo de alocação simples, sem aumentar demasiadamente a complexidade computacional do mecanismo. O mecanismo desenvolvido combina um leilão de único item ou um mecanismo de alocação aleatória com um protocolo de realocação de alvos através de contratos de troca. Uma primeira alocação de alvos para os robôs é realizada utilizando-se o leilão de único item ou a alocação aleatória e, em seguida, cada robô pode criar contratos de troca de alvos com outros robôs. Esses contratos proveem uma linguagem comum que permite aos robôs expressar seu interesse em negociar alvos com outros robôs de forma a melhorar a alocação inicial. Quando um robô recebe um conjunto de contratos de outros robôs, ele decide quais deles são vantajosos e aceita a troca. Os resultados empíricos da abordagem de contratos de troca mostraram que ele é capaz de melhorar a qualidade da alocação provida pelo mecanismo de alocação inicial.

Zhang et al. [2010] propuseram um mecanismo híbrido para melhorar a qualidade da alocação de um método simples que segue a mesma ideia de troca de itens após a alocação inicial. No entanto, o rearranjo de alvos acontece de forma estocástica, por meio de um algoritmo de *Simmulated Annealing*, que procura por uma alocação ótima global no espaço de possíveis soluções de alocação. Um grafo representa a posição dos robôs e dos alvos e a alocação inicial cria as arestas que ligam os robôs aos alvos alocados para os mesmos. Os experimentos realizados neste trabalho mostraram que este mecanismo foi capaz de melhorar o custo da alocação inicial.

No trabalho proposto por de Hoog et al. [2009] foi desenvolvido um mecanismo de exploração híbrido que combina um mecanismo de coordenação baseado em papéis com a estratégia de exploração baseada em fronteiras para explorar ambientes totalmente desconhecidos. Neste trabalho, os robôs precisam entregar as informações coletadas ainda durante missão a uma central de comando. No mecanismo de coordenação desenvolvido, alguns robôs recebem o papel de *exploradores*, enquanto outros recebem o papel de *entregadores*. Os robôs exploradores são responsáveis por explorar o ambiente utilizando a estratégia de exploração baseada em fronteiras. Para escolher qual fronteira o explorador irá visitar, este robô computa o custo de todos os pares explorador-fronteira que ele conhece, e assim saberá qual fronteira é a melhor para

visitar. Os robôs entregadores são responsáveis por seguir os robôs exploradores e, de tempos em tempos, precisam voltar a central de comando para entregar as informações colhidas pelos exploradores. Quando um entregador precisa voltar a central, ele combina com seu robô explorador associado um ponto de encontro. Os experimentos deste trabalho mostraram que este mecanismo não é tão eficiente em termos dos custos da missão devido aos custos da volta dos robôs entregadores e as viagens aos pontos de reencontro. No entanto o principal objetivo do trabalho é entregar o maior número de informações possível a central de comando durante a missão.

Capítulo 3

Metodologia

Como foi discutido no Capítulo 2, mecanismos de coordenação baseados em técnicas de leilão são uma das abordagens mais utilizadas na coordenação de múltiplos robôs [Dias et al., 2006; Viguria & Howard, 2010]. Leilões são fáceis de projetar e de implementar e apresentam baixa complexidade computacional quando comparados com soluções centralizadas. Além disso, leilões são mecanismos tolerantes a falhas individuais e apresentam baixa necessidade de comunicação [Gerkey & Matarić, 2002]. Neste trabalho, estudamos os principais tipos de leilão propostos na literatura, bem como alguns algoritmos utilizados para computação de lances, aplicados à observação de múltiplos alvos em ambientes conhecidos. A Figura 3.1 ilustra a classificação do presente trabalho de acordo com a taxonomia proposta no Capítulo 2.

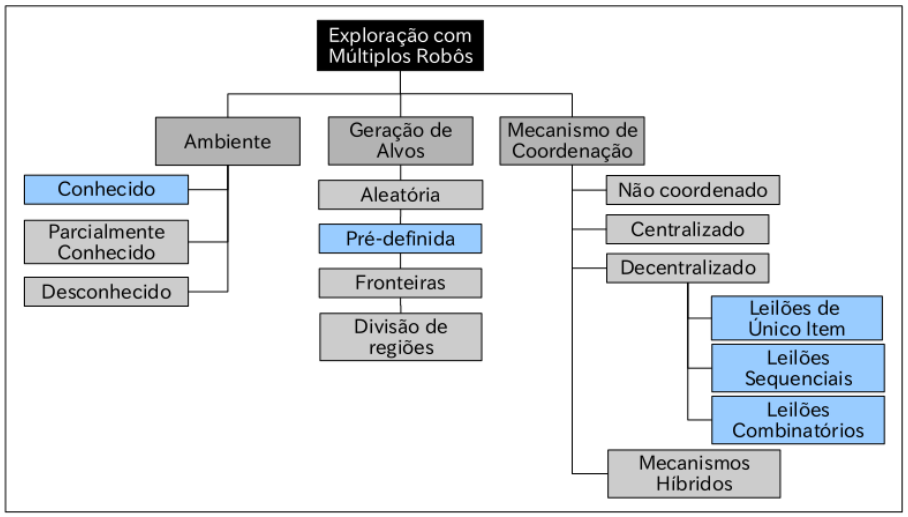

Figura 3.1. Classificação do presente trabalho.

O arcabouço de leilão implementado neste trabalho é completamente decentra-

lizado, no sentido que não existe uma entidade central, o leiloeiro, responsável por receber os lances de todos os robôs e resolver o problema de determinação do vencedor. Neste tipo de leilão, todos os robôs agem como leiloeiros. Após computar o lance para um determinado item, os robôs enviam seus lances para todos os outros robôs. Assume-se que a comunicação é perfeita. Ao receber os lances de seus companheiros, ou após um determinado tempo, cada robô utiliza um mesmo algoritmo para determinar o vencedor. Um leilão totalmente descentralizado apresenta algumas vantagens e desvantagens em relação a um protocolo de leilão centralizado. A ausência de uma entidade que age como leiloeiro evita o problema de haver um ponto central de falhas no sistema, pois se o leiloeiro falhar, todo o sistema falha. O leiloeiro central também funciona como um gargalo para o sistema, pois ele precisa escutar as mensagens de todos os robôs e em seguida precisa notificar a todos sobre o vencedor cada etapa do leilão. No entanto, o leilão distribuído exige o tratamento de problemas de sincronia entre os robôs, além de que o processo de determinação do vencedor é redundante, pois é executado por cada robô.

Um outro aspecto dos mecanismos de leilão implementados neste trabalho é que estes são leilões reversos [Jap, 2002]. No leilão reverso, os robôs competem pela visita dos alvos, estando mais apto aquele robô que apresenta o menor custo para visitar o alvo leiloado. Assim, os robôs dão lances com base nos custos de execução, vencendo aquele que apresenta o menor custo. Esse tipo de leilão é muito utilizado em licitações governamentais, onde empresas competem pela execução de um projeto e vence aquela capaz de executar o projeto especificado com menor custo e maior qualidade.

3.1 Leilão de Único Item

Neste trabalho implementamos um leilão de único item como proposto na literatura [Dias et al., 2006]. Os passos deste protocolo de leilão podem ser vistos no Algoritmo 1. Cada robô executa o mesmo algoritmo de leilão em paralelo, comunicando-se com os demais robôs através de troca de mensagens. Todos os robôs conhecem a lista de alvos a ser leiloada e a respectiva posição dos alvos no ambiente. Esta lista de alvos é passada como parâmetro para o leilão. Em cada iteração do laço nas linhas 1–10, que consiste em uma etapa do leilão, um alvo da lista é analisado e o lance para aquele alvo é computado e enviado aos demais robôs. O próximo alvo da lista a ser leiloado é recuperado na linha 2 e o valor do lance para este alvo é gerado na linhas 3. No leilão de único item, o lance de um robô para um alvo consiste no custo para o robô se mover da sua posição inicial para a posição do alvo, e é calculado em $O(1)$. Na linha

4, o robô adiciona o seu lance à sua lista local de lances e em seguida envia este lance para todos os outros robôs (linha 5). Em seguida, o robô aguarda um tempo para receber os lances dos demais (linha 6). Após receber os lances de todos os outros, o robô computa o lance vencedor (linha 7). A computação do vencedor consiste apenas em procurar o menor lance na lista de lances e tem complexidade $O(r)$, onde r é o número de robôs. Caso o próprio robô seja o vencedor do alvo, este robô adiciona o alvo a sua lista de alvos visitar (linha 9). Em caso de empates, o robô que possui o menor índice é o vencedor. Na linha 10, o robô remove o alvo analisado da lista de alvos a serem leiloados e o processo se repete até a lista estar vazia. Ao final do leilão cada robô possui uma lista contendo os alvos vencidos durante o leilão e precisa visitar estes alvos. O laço nas linhas 1–10 é executado n vezes, onde n é o número de alvos a serem leiloados, e a operação de maior custo neste laço é a computação do vencedor, cuja complexidade é $O(r)$. Sendo assim, a complexidade total do leilão de único item implementado é da ordem $O(n \cdot r)$.

Algoritmo 1: Leilão de Único Item.

```
   Entrada: listaAlvosAmbiente
 1 enquanto leilao não acabou faça
 2     alvo ← proximo alvo a ser leiloado
 3     valor_lance ← distancia(posicao_robo, posicao_alvo)
 4     adicionarLance(lance)
 5     broadcastMensagem(lance)
 6     aguardarTempo(Δ)
 7     lanceVencedor ← computarVencedor(listaDeLancesRecebidos)
 8     se lanceVencedor.robo=proprio robo então
 9         adcionar(listaAlvosRobo, alvo)
10     remover(listaAlvosAmbiente, alvo)
```

A Figura 3.2 ilustra como ocorre o processo de leilão de único item. Inicialmente, todos os robôs conhecem a posição dos alvos a serem leiloados. A cada etapa, um alvo da sequência é atribuído ao robô que apresenta menor distância entre sua posição inicial e o alvo oferecido. Na Figura 3.2(a), os robôs oferecem como lance a distância entre si e o alvo t_1, vencendo o robô R_1. Na Figura 3.2(b), o robô R_1 é novamente o vencedor, pois sua distância para o alvo t_2 é a menor. O processo se repete até que todos os alvos tenham sido leiloados, como pode ser visto na Figura 3.2(j).

3.2 Leilões Sequenciais de Único Item

Leilões Sequenciais de Único Item (do inglês, *Sequential Single-Item* – SSI) são uma extensão dos leilões de único item capazes de prover soluções melhores que os

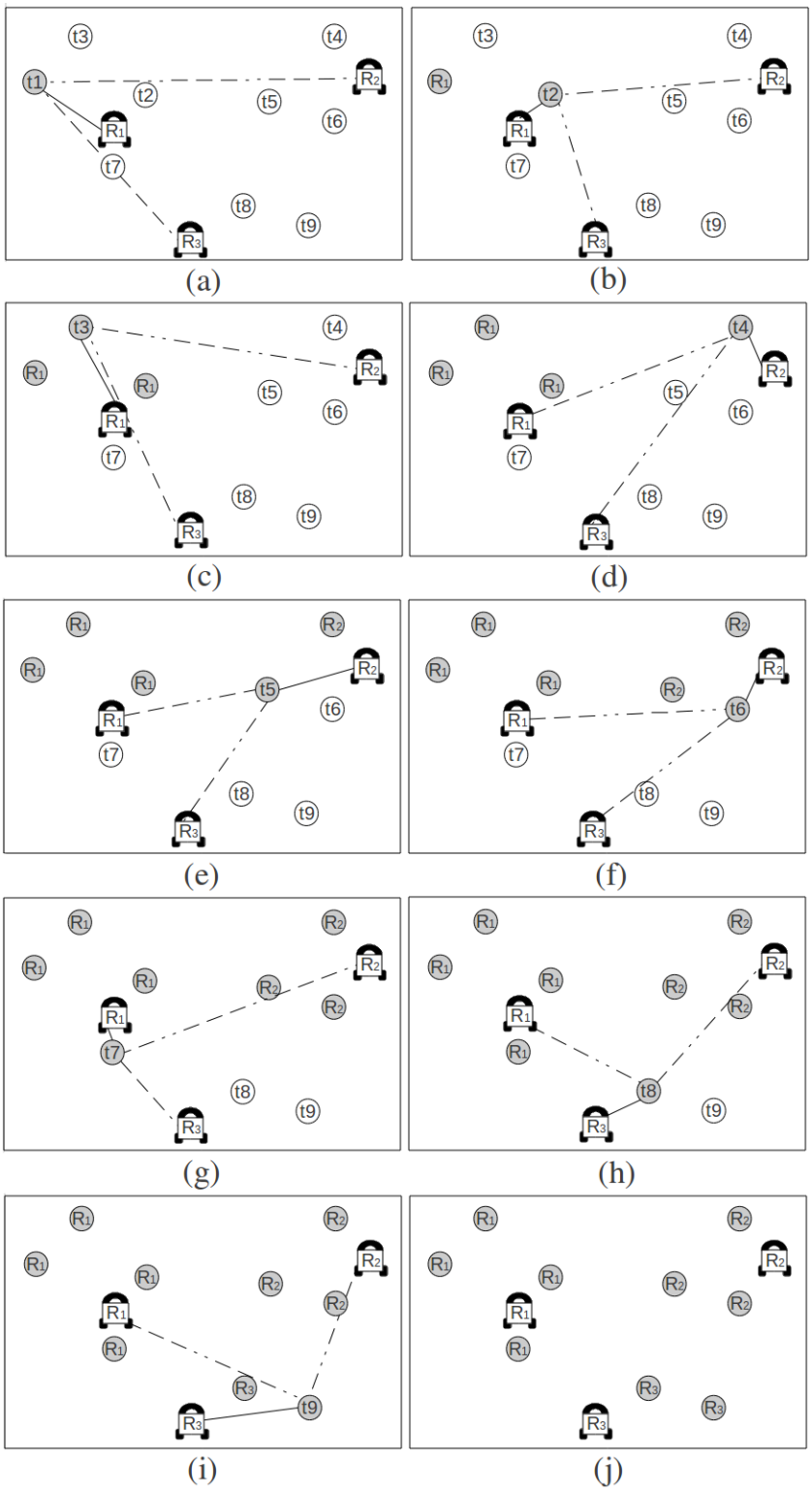

Figura 3.2. Leilão de único item.

leilões de único item por captar algumas sinergias entre os itens durante o leilão [Lagoudakis et al., 2004]. Em um cenário onde a função objetivo consiste em minimizar as distâncias viajadas pelos robôs durante a missão, quanto menor a distância entre dois alvos, maior a sinergia entre eles. Em leilões de único item, um robô computa o lance para um alvo oferecido de forma independente dos alvos já oferecidos em etapas anteriores do leilão. Em leilões SSI, por outro lado, os robôs utilizam algum algoritmo para computar lances para um novo alvo com base nos alvos que ele venceu em etapas anteriores do leilão, tentando adquirir alvos que apresentam alguma sinergia com os alvos já vencidos.

O Algoritmo 2 descreve o protocolo de leilão SSI, que é similar ao protocolo de leilão de único item (Algoritmo 1), com a diferença na função que computa o lance para o alvo na linha 3. A computação do lance no leilão SSI é realizada por um algoritmo capaz de analisar os alvos que o robô já venceu anteriormente. Os algoritmos utilizados neste trabalho para computação de lances durante o leilão sequencial serão descritos adiante. Ao final do leilão cada robô possui uma lista contendo os alvos vencidos durante o leilão e precisa visitar esses alvos. O custo computacional do leilão SSI depende do algoritmo utilizado para computar os lances.

Algoritmo 2: Leilão Sequencial − SSI.

```
   Entrada: listaAlvosAmbiente
 1 enquanto leilao não acabou faça
 2     alvo ← proximo alvo a ser leiloado
 3     valor_lance ← computarLance(algoritmo, alvo, lista de alvos já alocados para o robô)
 4     adicionarLance(lance)
 5     broadcastMensagem(lance)
 6     aguardarTempo(Δ)
 7     lanceVencedor ← computarVencedor(listaDeLancesRecebidos)
 8     se lanceVencedor.robo=proprio robo então
 9         adcionar(listaAlvosRobo, alvo)
10     remover(listaAlvosAmbiente, alvo)
```

O algoritmo utilizado para computar os lances durante o leilão SSI influencia diretamente na qualidade da solução final provida pelo leilão. Neste trabalho, dois algoritmos para computar lances em um leilão SSI foram implementados e serão discutidos a seguir.

O primeiro algoritmo implementado foi proposto em [Lagoudakis et al., 2004] e será referido no presente trabalho como PRIM *Allocation*, assim como no trabalho original. As posições dos robôs e dos alvos são modeladas como um grafo $G = (V, E)$ e, durante o leilão, cada robô tenta construir uma Árvore Geradora Mínima (do inglês *Minimum Spanning Tree* − MST) com os alvos leiloados. Ao se iniciar o leilão, cada

robô possui uma MST contendo somente o vértice que representa sua posição como raiz dessa MST. O lance de um robô para um novo alvo oferecido no leilão é o custo de uma nova MST com a adição do vértice que representa esse novo alvo menos o custo da MST antiga. O robô que menos incrementa o custo de sua MST com a adição do novo vértice será o vencedor do alvo, atualizando a sua MST, que será mantida durante todo o leilão. A computação de lances utilizando o algoritmo PRIM implementado neste trabalho tem complexidade computacional $O(n^2)$.

A Figura 3.3 ilustra o processo de leilão SSI utilizando o algoritmo PRIM Allocation. Cada robô inicia com uma MST contendo apenas sua posição inicial como raiz da árvore. Na Figura 3.3(a), cada robô dá lance para o alvo t_1 como sendo o custo para se inserir este alvo em sua MST, que é o custo da aresta entre a raiz da árvore e o novo alvo. O robô R_1 ganha o alvo pois apresenta o menor custo. Na Figura 3.3(b), os robôs novamente computam a diferença de custo com a inserção do alvo oferecido em suas MSTs. O robô R_1 vence novamente, ligando o alvo t_2 a raiz de sua MST. Na Figura 3.3(c), o robô R_1 dá lance para o alvo t_3 como sendo o custo da aresta (t_1,t_3), pois o custo dessa aresta representa a diferença entre a nova MST com a inserção do novo alvo e a MST antiga. O processo se repete até que todos os alvos estejam na MST de algum robô.

Uma segunda forma computar lances em leilões SSI implementada neste trabalho é baseada na ideia proposta por Koenig et al. [2006], no qual os autores sugerem o uso de heurísticas polinomiais capazes de resolver o problema do Caixeiro Viajante (TSP) para computar lances para os alvos durante o leilão SSI. Neste trabalho foi implementada a heurística de Inserção do Mais Distante (*Farthest Insertion Heuristic* – FI), uma heurística construtiva capaz de prover uma solução candidata de baixo custo para o TSP em tempo polinomial.

A heurística FI modela as posições do robô e dos vértices como um grafo $G = (V, E)$ completo. FI inicia com um circuito vazio e gradualmente constrói um circuito de menor custo com os alvos, sempre inserindo no circuito o alvo que está mais longe do circuito. A lógica por trás desta heurística é que alguns alvos serão mais caros para inserir no circuito. Logo tenta-se inserir primeiro os alvos mais caros formando um esboço inicial do que será o circuito final, e em seguida insere-se os alvos restantes nas posições mais adequadas desse circuito inicial. Dessa forma, tenta-se evitar a criação de arestas cruzadas, que incrementam o custo do circuito final.

O Algoritmo 3 descreve os passos da heurística FI para computação de um circuito de menor custo a partir de uma lista de alvos e a posição inicial do robô. Inicialmente, cria-se uma aresta entre a posição inicial do robô e o alvo mais distante dessa posição, (linhas 3 e 4). Procurar pelo alvo mais distante é uma operação de complexidade $O(n)$,

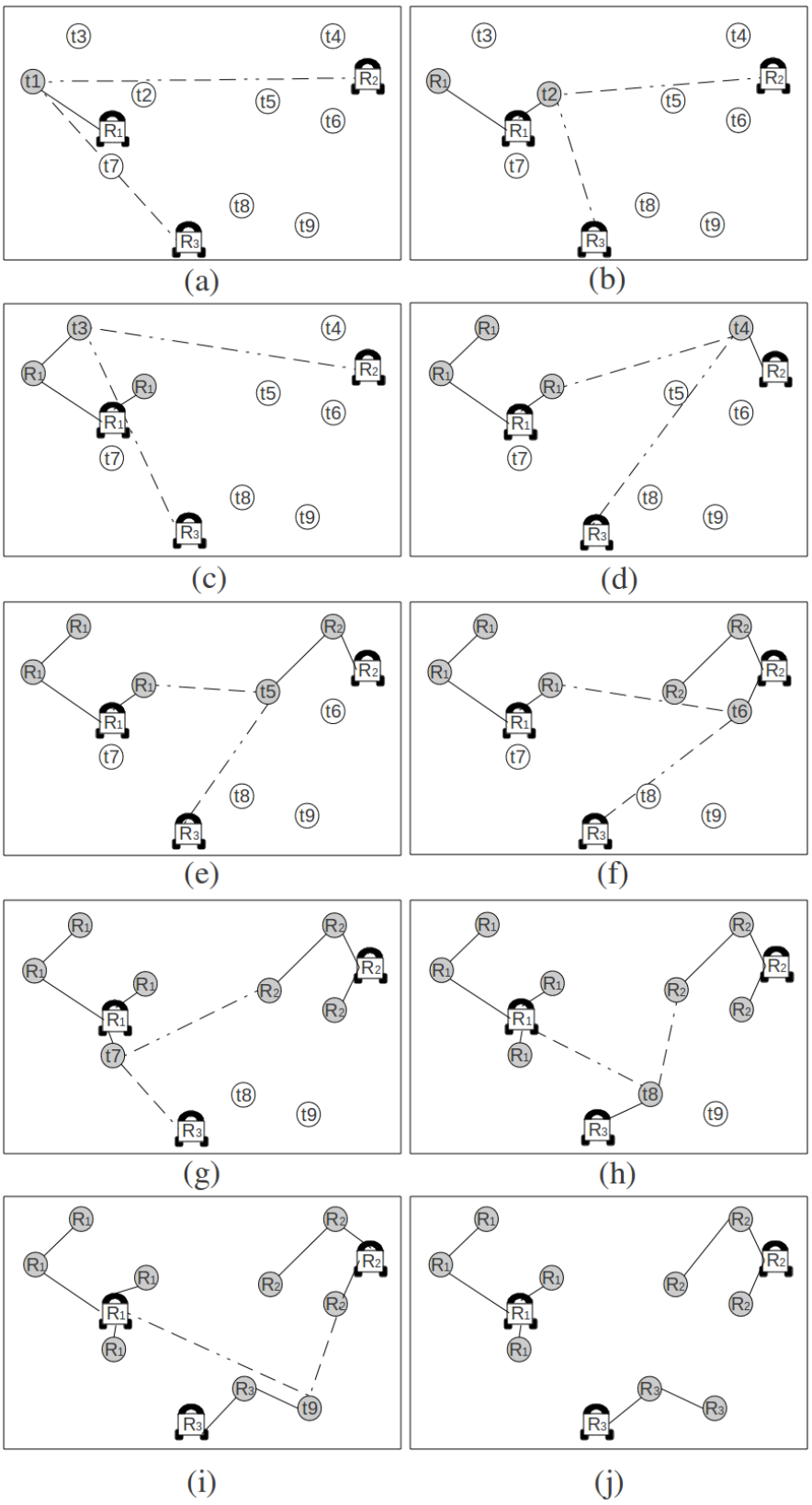

Figura 3.3. Leilão SSI utilizando a heurística PRIM Allocation.

para n alvos na lista. Em seguida, o algoritmo procura pelo alvo que se encontra mais distante da aresta criada, na linha 5, e com esse novo alvo, fecha-se um circuito inicial. Para todo os outros $n-2$ alvos, no laço das linhas $7-11$, o algoritmo procura o alvo mais distante do circuito em $O(n)$, na linha 8, e adiciona esse novo alvo na posição que menos incrementa o custo do circuito (linhas $8-11$). A operação de procurar pela posição mais adequada tem complexidade $O(n)$ também. Dessa forma, o custo total do algoritmo é $O(n^2)$.

Algoritmo 3: Heurística Inserção do Vizinho Mais Distante (FI).

```
1  computarCicloFI(posição do robô, lista de alvos)
2  início
3      encontre o alvo v_1 mais distante do vértice do robô v_r na lista de alvos
4      crie uma aresta (v_r,v_1)
5      encontre o alvo v_2 mais distante da aresta (v_r,v_1)
6      crie uma aresta (v_2,v_r) fechando um circuito
7      para todos os outros alvos v_i na lista de alvos faça
8          encontre o alvo v_x mais distante do circuito
9          encontre a aresta (v_a,v_b) do circuito da qual v_x está mais próximo
10         remova a aresta (v_a,v_b) do circuito
11         crie duas novas arestas (v_a,v_x) e (v_x,v_b)
12     retorne circuito
13 fim
```

A Figura 3.4 ilustra a criação de um circuito de menor custo com a utilização da heurística FI para um conjunto de 9 alvos. Inicialmente, encontra-se o alvo mais distante do robô, que é o alvo t_4 (Figura 3.4(b)), criando-se uma aresta entre os robô e este alvo. Em seguida, encontra-se o vértice mais longe da aresta criada, que é o alvo t_9, fechando-se um circuito, como pode ser visto na Figura 3.4(c). Nos passos seguintes, o algoritmo encontra o alvo mais distante do circuito formado, sempre inserindo esse alvo na posição do circuito que menos incrementa o custo desse circuito. O algoritmo termina quando todos os alvos da lista se encontram no circuito, como pode ser visto na Figura 3.4(j).

Durante o leilão, o lance de um robô para um novo alvo oferecido é a diferença no custo do novo circuito contendo os alvos já alocados para este robô e o novo alvo menos o custo do circuito anterior. A Figura 3.5 ilustra o processo de leilão SSI que utiliza a heurística FI para computação de lances. Na Figura 3.5(a), cada robô oferece como lance para t_1 o custo do circuito contendo apenas sua posição inicial e o alvo t_1, que corresponde ao custo de o robô ir até t_1 e retornar a sua posição inicial. O robô R_1 apresenta o menor custo e ganha o alvo. Na 3.5(b), os robôs dão lance para t_2 e, como robô R_1 é aquele que menos incrementa o custo do circuito com a inserção do alvo t_2, também vence esse alvo. O processo continua, sempre vencendo o robô que

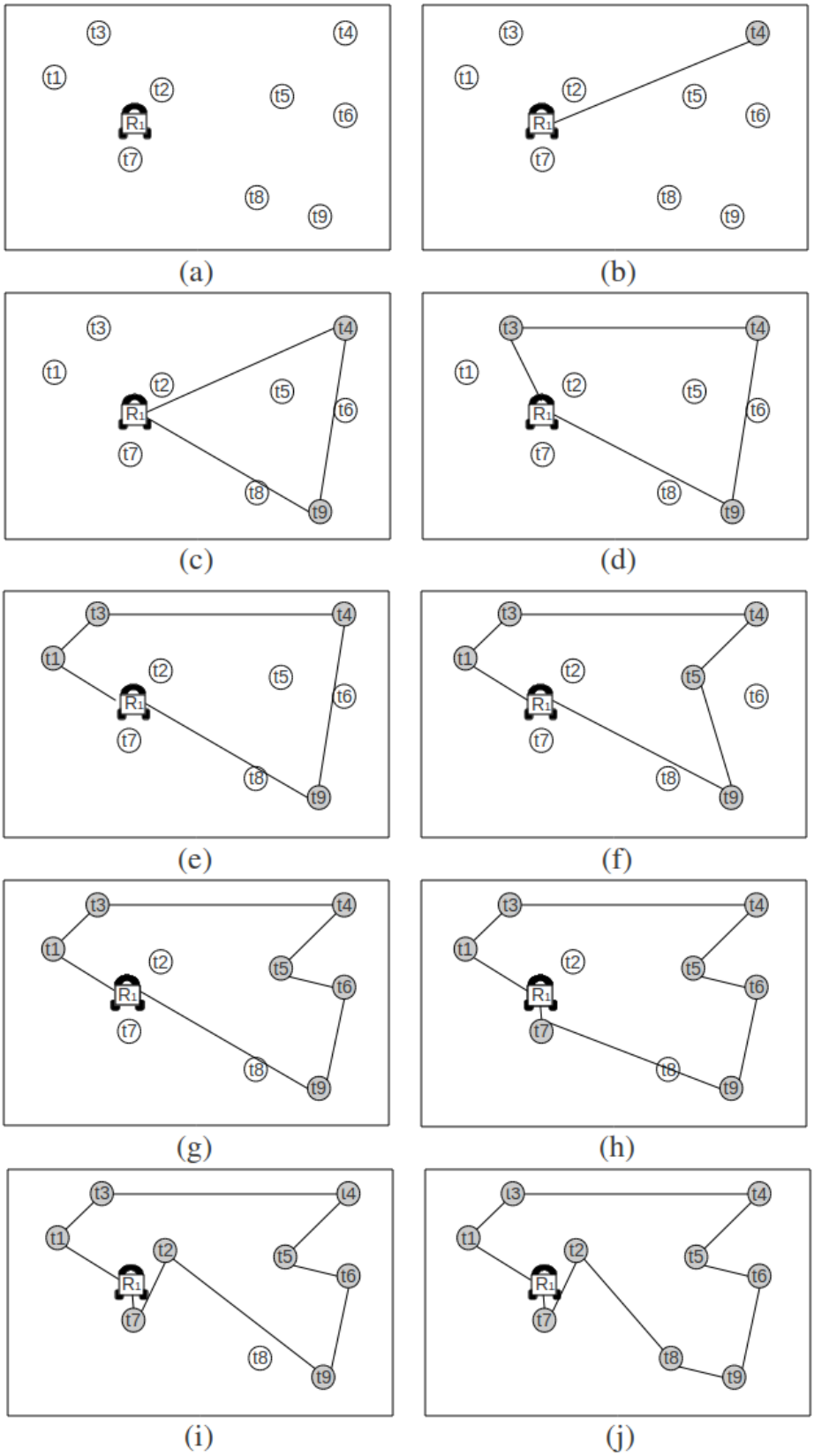

Figura 3.4. Heurística de Inserção do Vizinho Mais Distante.

menos incrementa o custo de seu circuito com a inserção do novo alvo. Ao final do leilão, cada robô possui um circuito de menor custo, como pode ser visto na 3.5(j), sendo necessário que cada robô realize esse percurso para visitar os alvos.

3.3 Leilões Combinatórios

Como discutido anteriormente, um leilão combinatório é aquele onde os compradores podem oferecer lances para mais de um item ao mesmo tempo. Leilões combinatórios se mostram mecanismos de alocação mais eficazes para o problema de observação de múltiplos alvos, pois permitem que os robôs ofereçam lances para combinações de alvos com alta sinergia, minimizando os custos da alocação final.

O protocolo de leilão combinatório se divide em duas etapas. Na primeira etapa, referida aqui como *Formulação de Lances*, cada comprador precisa escolher em quais combinações de itens ele dará lance. Avaliar todas as combinações possíveis é um problema de complexidade exponencial, dado o número de itens a ser avaliado. A segunda etapa do leilão consiste no problema de *Determinação dos Vencedores* do leilão, que consiste em determinar, dentre os lances oferecidos, quais são mais lucrativos. Este problema também é exponencial quando se permite que os compradores ofereçam lances para qualquer combinação possível de itens.

De forma a tornar os leilões combinatórios aplicáveis à alocação de alvos para os robôs em missões de exploração, é preciso investigar estratégias para eliminar a intratabilidade tanto na fase de formulação de lances, como na fase de determinação dos vencedores. Existem algumas estratégias de limitação dos lances que podem ser formulados pelos compradores durante o leilão combinatório capazes de tornar ambas as fases do leilão operações que podem ser resolvidas em tempo polinomial. Neste trabalho propomos a utilização de duas estratégias. A primeira, denominada *Árvore de Pacotes* é descrita na Seção 3.3.1. Já a segunda, denominada *Ordenação de Itens*, será descrita na Seção 3.3.2.

3.3.1 Árvore de Pacotes

Em um leilão combinatório baseado em árvore de pacotes, as etapas de Formulação dos Lances e de Determinação dos Vencedores são realizadas em quatro fases, como pode ser visto na Figura 3.6. Estas fases são descritas a seguir.

A primeira fase do leilão consiste em criar uma árvore contendo possíveis combinações dos itens para os quais os robôs podem dar lances durante o leilão combinatório. A raiz dessa árvore é um pacote contendo todos os itens a serem leiloados. Este pacote

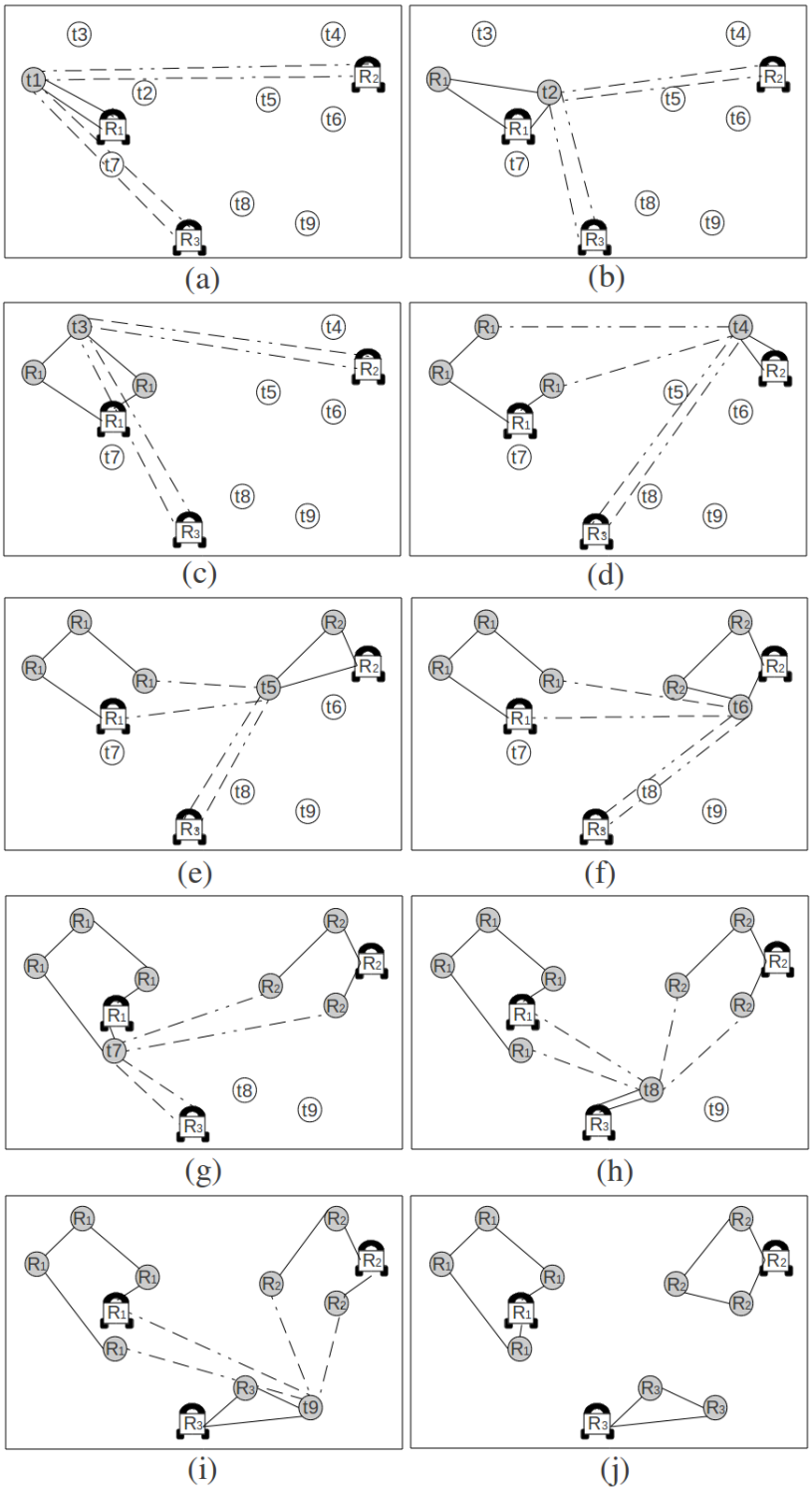

Figura 3.5. Leilão SSI utilizando a heurística FI.

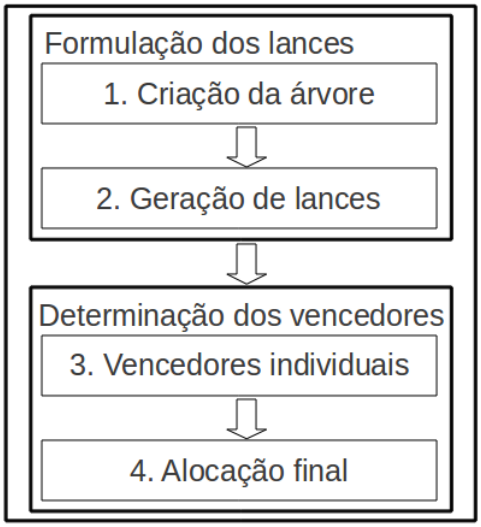

Figura 3.6. Fases do leilão combinatório baseado em árvore de pacotes.

maior é subdividido gerando um novo nível na árvore composto por pacotes menores. Os pacotes menores são novamente subdivididos gerando mais um nível, e o processo de subdivisão continua até que os nós-folha da árvore sejam constituídos apenas por pacotes indivisíveis, possuindo apenas um item individual. Existe uma restrição para a construção da árvore de pacotes que deve ser obedecida de forma a permitir que a computação dos vencedores possa ser feita em tempo polinomial: os nós ou pacotes que formam a árvore devem ser totalmente disjuntos ou um estar contido no outro, ou seja, dois nós não podem possuir um mesmo item a não ser que um nó represente um subconjunto dos itens contidos no outro nó, que está em níveis acima na árvore. Como cada robô utiliza o mesmo algoritmo para geração da árvore, ao final da primeira fase, todos os robôs têm como resultado uma mesma árvore. A Figura 3.7(b) ilustra uma possível árvore de pacotes contendo 5 alvos do ambiente ilustrado na Figura 3.7(a).

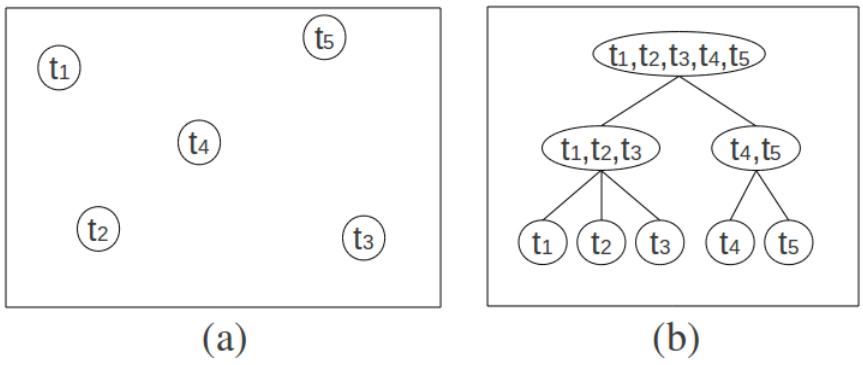

Figura 3.7. (a) Alvos a serem explorados em um ambiente. (b) Possível árvore de pacotes para os alvos.

A segunda fase do leilão consiste na geração de lances para os nós da árvore de pacotes. Nesta fase, os robôs podem oferecer lances para quaisquer pacotes de alvos da árvore. Dado que a árvore criada possui um número polinomial de pacotes, a estrutura de árvore torna a formulação de lances um problema polinomial em função desse número de pacotes. No exemplo ilustrado na Figura 3.7 os robôs que participam do leilão para comprar os alvos oferecidos precisam computar lances apenas para os 8 pacotes que constituem a árvore ao invés de 31 pacotes, caso fossem permitidas todas as combinações de alvos. O lance de um robô para um pacote da árvore é o custo para o robô visitar todos os alvos pertencentes àquele pacote, e pode ser calculado computando-se uma solução candidata para circuito TSP contendo estes alvos através de uma heurística polinomial. Após computar o lance para cada pacote da árvore, os robôs enviam suas árvores de lances para todos os outros robôs.

A terceira fase do leilão combinatório consiste em determinar os lances vencedores de cada pacote individual da árvore. Para cada pacote na árvore, o leilão compara os lances enviados por cada robô participante do leilão, gerando uma nova árvore contendo as informações dos lances vencedores, como o valor do lance e o robô que o gerou. Em caso de empates, o robô com menor índice é o vencedor. A Figura 3.8 ilustra um exemplo da terceira fase do leilão de 5 alvos para três robôs participantes. Pode-se ver, por exemplo, que para o nó-raiz, o robô 2, dentre os três, foi o que deu o menor lance. Já para o pacote $\{t_1,t_2,t_3\}$, o robô R_1, ofereceu menor lance.

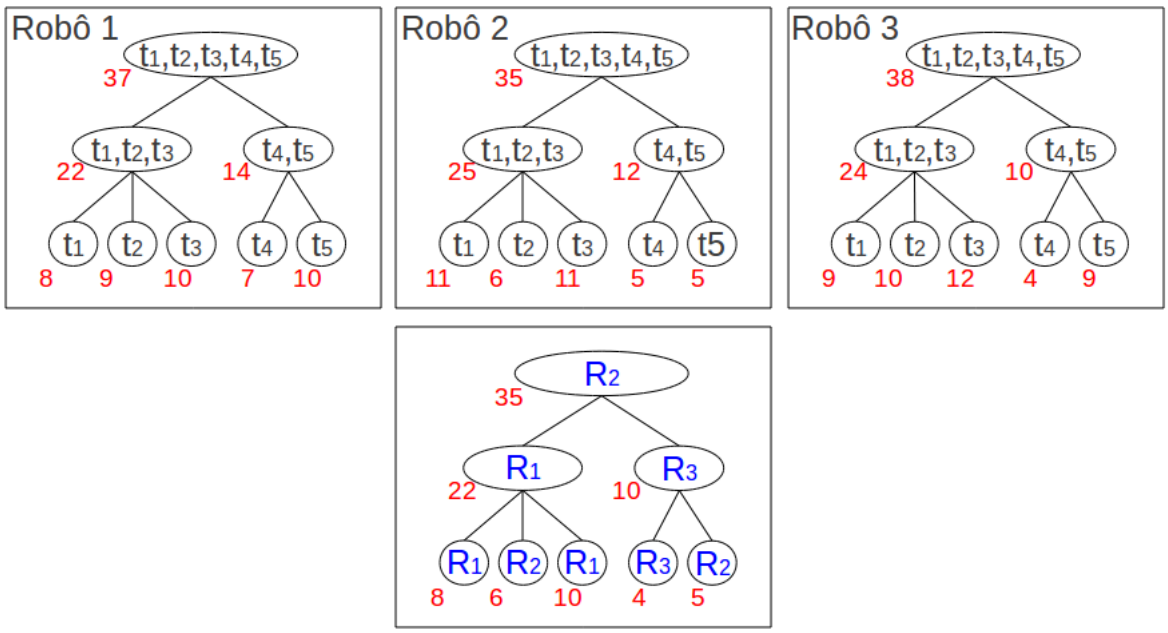

Figura 3.8. Terceira fase do leilão combinatório.

A quarta fase do leilão consiste em determinar a alocação final dos pacotes para os robôs. O Algoritmo 4 percorre recursivamente a árvore de pacotes gerada na terceira fase do leilão marcando quais são os pacotes ou nós desta árvore mais lucrativos para serem alocados aos robôs, determinando assim a alocação final. Este algoritmo recebe como entrada o nó P_i que é a raiz da árvore. O caso base deste algoritmo recursivo consiste em um nó que não possui filhos, então ele próprio é marcado parcialmente como vencedor (linhas 3 e 4). Se o nó P_i passado como parâmetro não é um nó folha, então o algoritmo invoca a si mesmo para cada filho de P_i passado como parâmetro (linhas 6 e 7). Ao determinar os vencedores para cada filho, o algoritmo decide se o lance recebido por P_i é menor ou igual a soma dos valores dos lances dados para os filhos de P_i. Se essa condição for verdadeira, então, na linha 10, P_i é marcado como vencedor e seus filhos são marcados como não alocados, na linha 11. Senão, na linha 13, P_i é marcado como não alocado, permanecendo a alocação anterior determinada para os filhos do nó e na linha 14, o valor de P_i é atualizado com o valor dos filhos. Esta atualização é feita para que o processo recursivo considere o valor P_i como sendo o valor dos filhos, e assim, as decisões sobre a determinação do vencedor nos níveis acima da árvore considerem o valor dos filhos e não de P_i. Ao final do leilão, os nós da árvore marcados como vencedores são alocados para os robôs que geraram os lances para os pacotes representados por estes nós.

Algoritmo 4: Determinação do vencedor em uma árvore de lances disjuntos.

```
 1 determinarVencedor(P_i)
 2 início
 3     se P_i não possui filhos então
 4         marque P_i como vencedor
 5     senão
 6         para cada filho F_i de P_i faça
 7             determinarVencedor(F_i)
 8         compute a soma dos valores dos filhos de P_i
 9         se valor de P_i <= soma dos valores dos filhos então
10             marque P_i como vencedor
11             marque os filhos de P_i como não alocados
12         senão
13             marque P_i como não alocado
14             atualize o valor de P_i com o valor dos filhos
15 fim
```

A Figura 3.9 ilustra um exemplo da alocação final dada a árvore gerada na Figura 3.8. O algoritmo recebe como parâmetro o nó-raiz da árvore, que representa o pacote com todos os alvos do ambiente. Como este nó não é folha, então o algoritmo é invocado recursivamente até chegar aos nós-folha, que são marcados parcialmente como

vencedores. Em seguida, o algoritmo verifica que o valor do nó $\{t_1,t_2,t_3\}$, que é 22, é menor que 8+6+10=24, que é a soma dos valores dos filhos. Logo, $\{t_1,t_2,t_3\}$ é marcado como vencedor e os nós $\{t_1\}$,$\{t_2\}$,$\{t_3\}$ são marcados como não alocados. Para o nó $\{t_4,t_5\}$, acontece diferente, pois a soma do valor de seus filhos é menor que seu próprio valor. Logo o nó $\{t_4,t_5\}$ é marcado como não alocado, permanecendo a marcação dos vencedores realizada anteriormente para os níveis abaixo deste nó na árvore. O valor de $\{t_4,t_5\}$ é atualizado como o valor de seus filhos. Ao final o algoritmo precisa decidir se o valor do nó-raiz da árvore, 35, é menor que a soma dos valores dos filhos. Como 35 é maior que 22+9=31, que é o lance dado por $\{t_1,t_2,t_3\}$, mais o valor atualizado do nó $\{t_4,t_5\}$, então o nó-raiz é marcado como não alocado, permanecendo os filhos como vencedores. O valor da raiz é atualizado para 31, que corresponde ao custo total da alocação.

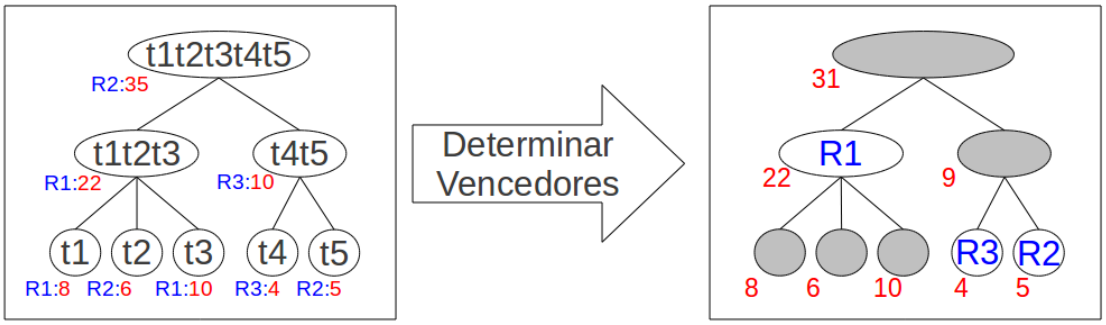

Figura 3.9. Determinação do vencedores em um leilão combinatório baseado em árvore de pacotes.

O algoritmo utilizado para para agrupar os alvos e construir a árvore de pacotes está diretamente relacionado com a qualidade da solução provida pelo leilão combinatório baseado em árvore de pacotes, pois é a árvore que limita as possíveis combinações de lances que os robôs podem oferecer durante o leilão. Neste trabalho, propomos a utilização de três algoritmos para criar a árvore de pacotes na primeira fase do leilão combinatório. A seguir iremos discutir estes três algoritmos.

3.3.1.1 Algoritmo de Corte Máximo

O primeiro método para criação da árvore implementado neste trabalho foi baseado na ideia proposta por [Berhault et al., 2003] de limitar as possíveis combinações de alvos a serem leiloadas com base no problema de corte máximo em um grafo. Durante a primeira fase do leilão, os alvos a serem leiloados são representados como um grafo ponderado completo $G = (V, E)$, onde os vértices V são as posições dos alvos no ambiente, as arestas E são ligações entre os vértices e os pesos destas estas arestas são

as distâncias entre os vértices no ambiente. O nó-raiz da árvore é formado por todos os vértices V. Esse nó-raiz precisa ser recursivamente subdividido, dando origem a nós-filho na árvore. Estes novos nós gerados são subdivididos dando origem a um novo nível na árvore, até que se chegue aos nós-folha, que são aqueles que possuem apenas um vértice.

O problema de subdivisão de um conjunto de vértices V de um nó da árvore foi modelado como o Problema do Corte Máximo (*MaxCut Problem*) em um grafo com pesos [Garey & Johnson, 1979]. Este problema consiste em encontrar uma partição de V produzindo dois subconjuntos de vértices, V' e seu complemento $V'' = V - V'$, de forma que a soma dos pesos das arestas que ligam os vértices em V' e V'' é o maior possível. Por meio do particionamento de um conjunto de vértices como um corte máximo, garante-se que a árvore gerada obedece a restrição de disjunção descrita anteriormente, pois um vértice nunca pertence a duas partições diferentes.

O problema de encontrar o corte máximo em um grafo é um problema NP-Difícil [Garey & Johnson, 1979], mas existem várias heurísticas polinomiais para resolver esse problema de forma aproximada. Neste trabalho, utilizamos uma heurística gulosa para criar a partição de um conjunto de vértices. Em seguida, um algoritmo de busca local é utilizado para refinar a partição fornecida pelo algoritmo guloso.

O procedimento guloso para computação do corte de um pacote pode ser visto no Algoritmo 5. Este algoritmo recebe como entrada um conjunto de vértices V a ser particionado, na linha 1. Se V possui apenas um vértice, então V não pode ser particionado e o algoritmo retorna V (linha 4). Senão, na linha 6 o algoritmo insere o primeiro vértice do conjunto V no subconjunto V'. Para todos os outros vértices em V, o laço nas linhas 7 a 11 decide se insere o vértice no subconjunto V' ou em V''. Se o valor do corte existente entre o vértice v_i e o conjunto V' é maior que o valor de corte entre o vértice v_i e o conjunto V'', então o algoritmo inserve v_i na partição V'', pois aumentará o valor do corte entre V' e V'' (linha 9). Caso contrário, v_i será inserido em V' (linha 10). Ao final todos os vértices em V estarão em um dos subconjuntos, que são retornados pelo algoritmo na linha 12. Este algoritmo tem complexidade $O(n^2)$, onde n é o número de vértices.

A Figura 3.10 ilustra o processo de criação da partição de um grafo utilizando o algoritmo guloso implementado neste trabalho. A Figura 3.10(a) mostra a disposição dos vértices $V = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ a ser particionado. O primeiro vértice, t_1, é inserido no conjunto V'. O algoritmo insere o vértice t_2 em V'', pois aumenta valor do corte entre V' e V'' (Figura 3.10(b)). Na Figura 3.10(c), o algoritmo verifica que o corte entre t_3 e $V''=\{t_2\}$ é maior que o corte entre t_3 e $V'=\{t_1\}$, logo insere t_3 em V'. Na Figura 3.10(d), o algoritmo identifica que o valor do corte entre t_4 e $V'=\{t_1 t_3\}$

Algoritmo 5: Algoritmo guloso de Corte Máximo.

```
1  corteMaximo(V)
2  início
3      se |V| = 1 então
4          retorne V
5      senão
6          V' ← v_1
7          para todos os outros vértices v_i ∈ V faça
8              se valor do corte (v_i, V')>=valor do corte (v_i, V'') então
9                  insira v_i em V''
10             senão
11                 insira v_i em V'
12     retorne V' e V''
13 fim
```

é maior que o corte entre t_4 e $V''=\{t_2\}$, pois a soma das arestas $E(t_4,t_1)$ + $E(t_4,t_3)$ é maior que o valor da aresta $E(t_4,t_2)$, logo t_4 é inserido em V'', mantendo o corte entre V e V'' o maior possível. O procedimento se repete até que todos os vértices V do grafo estejam em um dos dois subconjuntos.

Como esta heurística gulosa nem sempre encontra o corte ótimo, ao final de sua execução é executado um algoritmo de busca local para refinar a partição de vértices inicial. O espaço de busca é composto por todas as partições possíveis do conjunto V. A função de vizinhança utilizada define que a vizinhança de uma solução s é um conjunto de soluções $N(s)$ que podem ser geradas a partir da troca de um único vértice do conjunto V' para V'' ou de V'' para V'. Assim, uma solução possui $O(n)$ vizinhos, dado n vértices. Os passos da busca local podem ser vistos no Algoritmo 6. O algoritmo recebe como parâmetro a solução provida pelo algoritmo guloso (linha 1). O valor da solução auxiliar é inicializado com essa partição inicial passada como parâmetro (linha 3) e a solução corrente é inicializada com vazio ($\varnothing$) na linha 4. No laço das linhas 5–9, a busca local procede da seguinte forma: na linha 6 a solução corrente é atualizada com a solução auxiliar, permanecendo as duas soluções iguais. No laço das linhas 7–9, o algoritmo percorre todos os vizinhos da solução corrente, sempre guardando na solução auxiliar o vizinho z_i com maior custo encontrado, o que configura um vizinho aprimorante. Verificar o custo de um novo vizinho tem custo $O(n)$, que consiste em verificar a soma das novas arestas que surgem no corte com a troca de um vértice de uma partição para a outra. Após verificar todos os vizinhos da solução corrente, se foi encontrada uma solução auxiliar melhor, então o laço das linhas 5–9 se repete, atualizando a solução corrente com a solução auxiliar encontrada na iteração anterior (linha 6). Quando o algoritmo encontra uma solução que não possui

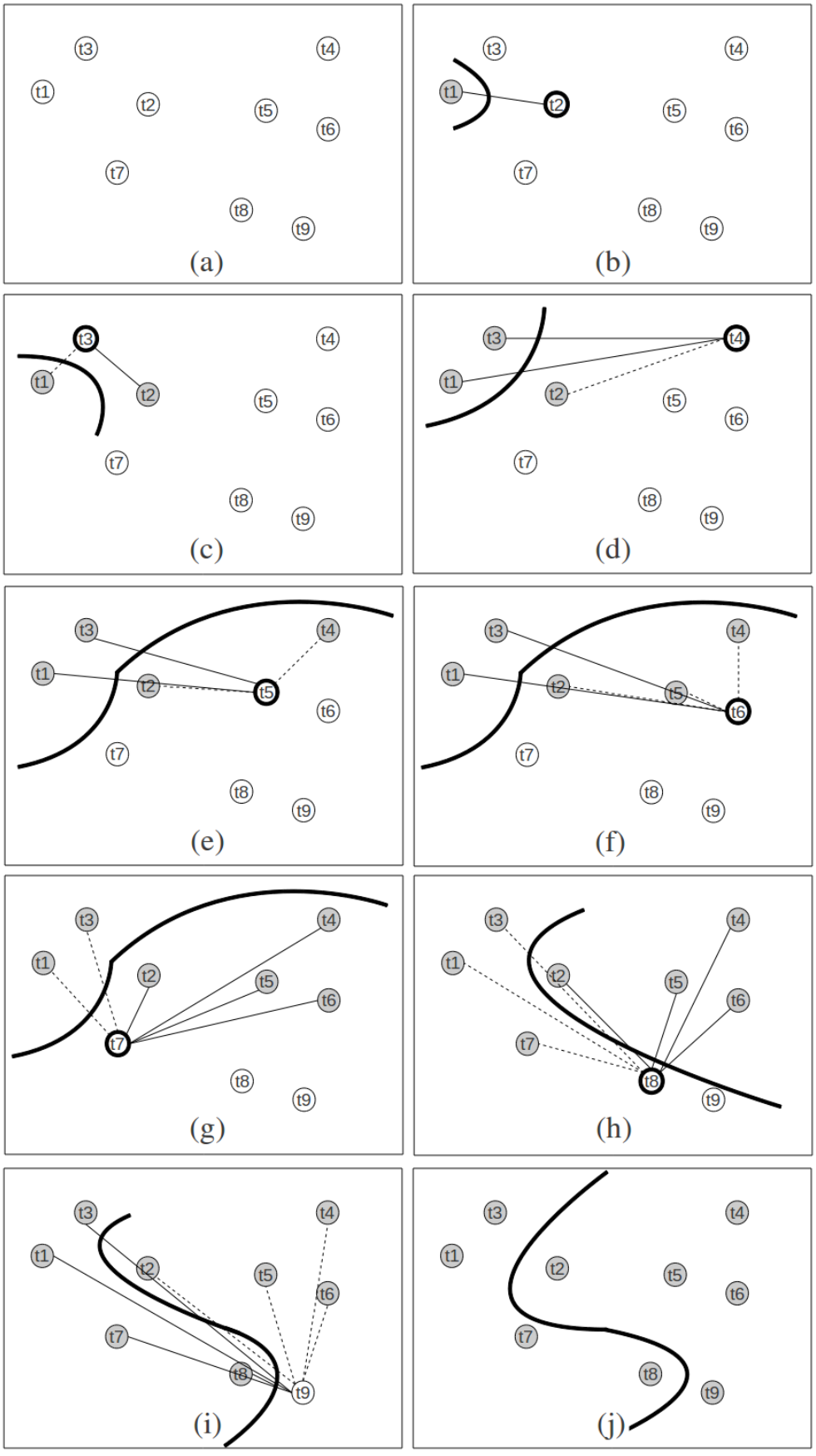

Figura 3.10. Partição de um grafo com o algoritmo guloso de corte máximo.

nenhum vizinho aprimorante, e por conseguinte a solução auxiliar não foi atualizada com nenhum vizinho da solução corrente, permanecendo as duas soluções iguais, o algoritmo finaliza sua execução, pois chegou em um ótimo local. Na linha 10, a solução corrente é retornada. Como cada solução possui $O(n)$ vizinhos e cada vizinho é avaliado em $O(n)$, a complexidade de cada iteração do algoritmo de busca local é $O(n^2)$.

Algoritmo 6: Algoritmo de busca local para o problema do corte máximo.

```
1  buscaLocalCorteMaximo(V',V'')
2  início
3      solução auxiliar ← (V',V'')
4      solução corrente ← ∅
5      enquanto solução auxiliar ≠ solução corrente faça
6          solução corrente ← solução auxiliar
7          para cada vizinho z_i da solução corrente faça
8              se custo de z_i > custo da solução auxiliar então
9                  solução auxiliar ← z_i
10     retorne solução corrente
11 fim
```

Este método de particionamento que combina o algoritmo guloso para o corte máximo junto com busca local é utilizado para construir a árvore de pacotes utilizada durante o leilão combinatório. A árvore gerada utilizando-se esta estratégia será uma árvore binária, dado que cada conjunto é particionado gerando dois novos nós. O Algoritmo 7 mostra o procedimento recursivo que constrói a árvore de pacotes que utiliza esse particionamento. O algoritmo recebe como entrada um nó P_i que contém todos os alvos do ambiente na linha 1. Este nó P_i será a raiz da árvore. A partir daí o algoritmo irá recursivamente particionar os nós da árvore até se deparar com o caso base da recursão, que é quando um nó gerado é um nó-folha (linha 4). Na linha 6, o algoritmo particiona P_i utilizando o algoritmo guloso para o corte máximo, produzindo uma partição inicial de P_i em V' e V''. Em seguida, o algoritmo refina o particionamento provido pelo algoritmo guloso com a busca local (linha 7). Após gerar dois novos nós na hierarquia da árvore, o algoritmo é invocado recursivamente para continuar a construção da árvore a partir desses novos nós (linhas 8 e 9).

3.3.1.2 Partição de Regiões

O segundo método para criação da árvore de pacotes implementado no presente trabalho foi baseado na ideia de partição de regiões, inicialmente proposta por Karp [1977]. Dado um conjunto de pontos em um plano, o método de partição de regiões encontra as dimensões da região que compreende todos os pontos e recursivamente subdivide

Algoritmo 7: Criação da árvore de pacotes utilizando o particionamento baseado no corte máximo.

```
1  arvoreCorteMaximo(P_i)
2  início
3      se P_i é nó-folha então
4          retorne P_i
5      senão
6          V' e V'' ← corteMaximo(P_i)
7          buscaLocalCorteMaximo(V',V'')
8          arvoreCorteMaximo(V')
9          arvoreCorteMaximo(V'')
10 fim
```

esta região em regiões menores, até que todas as regiões formadas contenham apenas um ponto cada.

Esse esquema de particionamento foi utilizado anteriormente para resolver o problema do Caixeiro Viajante (TSP) utilizando-se múltiplos processadores executando em paralelo [Johnson & McGeoch, 1997]. Dado um conjunto de cidades para o qual se deseja encontrar um circuito de menor custo contendo todas estas cidades, era preciso dividir esse conjunto de cidades em conjuntos menores e alocar os conjuntos menores para que cada processador compute uma parte do circuito TSP. Ao final da computação, cada processador fornece um circuito menor contendo o subconjunto de cidades para ele alocado, e estes circuitos menores são unidos, formando um circuito único que é a solução do TSP para o conjunto inicial de cidades.

A presente pesquisa propõe a utilização do método de partição de regiões para criar a árvore de pacotes contendo os alvos a serem leiloados no leilão combinatório. Inicialmente, encontra-se a região que compreende todos os alvos no ambiente e cria-se um pacote contendo todos estes alvos. Este pacote se torna o nó-raiz da árvore de pacotes. Ao passo que essa região é subdividida recursivamente, a árvore de pacotes vai sendo construída, onde uma região maior é um nó-pai e as sub-regiões que surgem pela divisão da região maior se tornam nós-filhos da região maior na árvore de pacotes. Novamente, como cada região é particionada gerando duas novas regiões, cada nó da árvore gera dois novos nós-filho, resultando em uma árvore binária. Espera-se que esta seja uma boa estratégia de construção da árvore, pois os pacotes formados são compostos por alvos que se encontram próximos no ambiente, e assim apresentam uma maior sinergia.

O Algoritmo 8 mostra os passos da computação recursiva para partição de regiões e criação da árvore de pacotes. O algoritmo recebe como entrada um nó P_i que contém todos os alvos do ambiente (linha 1). Este nó P_i será a raiz da árvore. A partir daí

o algoritmo irá recursivamente particionar os nós da árvore até se deparar com o caso base da recursão, que é quando um nó gerado é uma folha (linha 4). Se P_i não é folha, então o algoritmo encontra as dimensões l_1 e l_2 que definem dos lados do retângulo que compreende todos os alvos em P_i (linha 6). Esta operação tem complexidade $O(n)$, pois precisa verificar as coordenadas de todos os n alvos contidos em P_i. Se a dimensão l_x do retângulo for maior que a dimensão l_y então o retângulo será dividido ao meio por uma reta vertical (linha 8). Senão, o retângulo será divido ao meio por uma reta horizontal (linha 10). Na linha 11, o algoritmo cria os dois novos conjuntos P_i' e P_i'' que irão conter os alvos inicialmente em P_i. Em seguida, nas linhas 12 e 13 o algoritmo percorre o conjunto de alvos pertencente a P_i e, verificando das coordenadas do alvo, o algoritmo decide em qual das duas partições do retângulo o alvo pertence de acordo com a reta que divide o retângulo maior. Em seguida, o algoritmo invoca a si mesmo para particionar P_i' e P_i'', nas linhas 14 e 15. Ao final, na linha 16, o algoritmo torna P_i' e P_i'' nós-filho de P_i.

Algoritmo 8: Criação da árvore de pacotes através da partição de regiões.

```
1  particaoRegioes(P_i)
2  início
3      se P_i é nó-folha então
4          retorne P_i
5      senão
6          encontrar as dimensões l_x e l_x do retângulo
7          se (l_x >= l_x) então
8              dividir retângulo ao meio por uma reta vertical
9          senão
10             dividir retângulo ao meio por uma reta horizontal
11         crie as partições P_i' e P_i''
12         para cada alvo a_i ∈ P_i faça
13             insira a_i em P_i' ou P_i'1 de acordo com as coordenadas (x,y) de a_i em relação a
               reta traçada
14         particaoRegioes(P_i')
15         particaoRegioes(P_i'')
16         torne P_i' e P_i'' filhos de P_i
17 fim
```

A Figura 3.11 ilustra o processo de partição de regiões e criação de uma árvore de pacotes para um conjunto de 9 alvos em uma ambiente. O algoritmo encontra as dimensões do retângulo que compreende os alvos e cria o nó-raiz da árvore contendo todos estes alvos, como pode ser visto na Figura 3.11(a). Na Figura 3.11(b), o retângulo inicial é dividido por uma reta vertical, criando duas partições e consequentemente dois nós-filho na árvore. Na Figura 3.11(c), as duas partições criadas anteriormente dão origem a mais duas partições, criando mais um nível na árvore de pacotes. O processo

se repete até que cada região possua apenas um alvo, que são os nós-folha na árvore de pacotes, como pode ser visto na Figura 3.11(e).

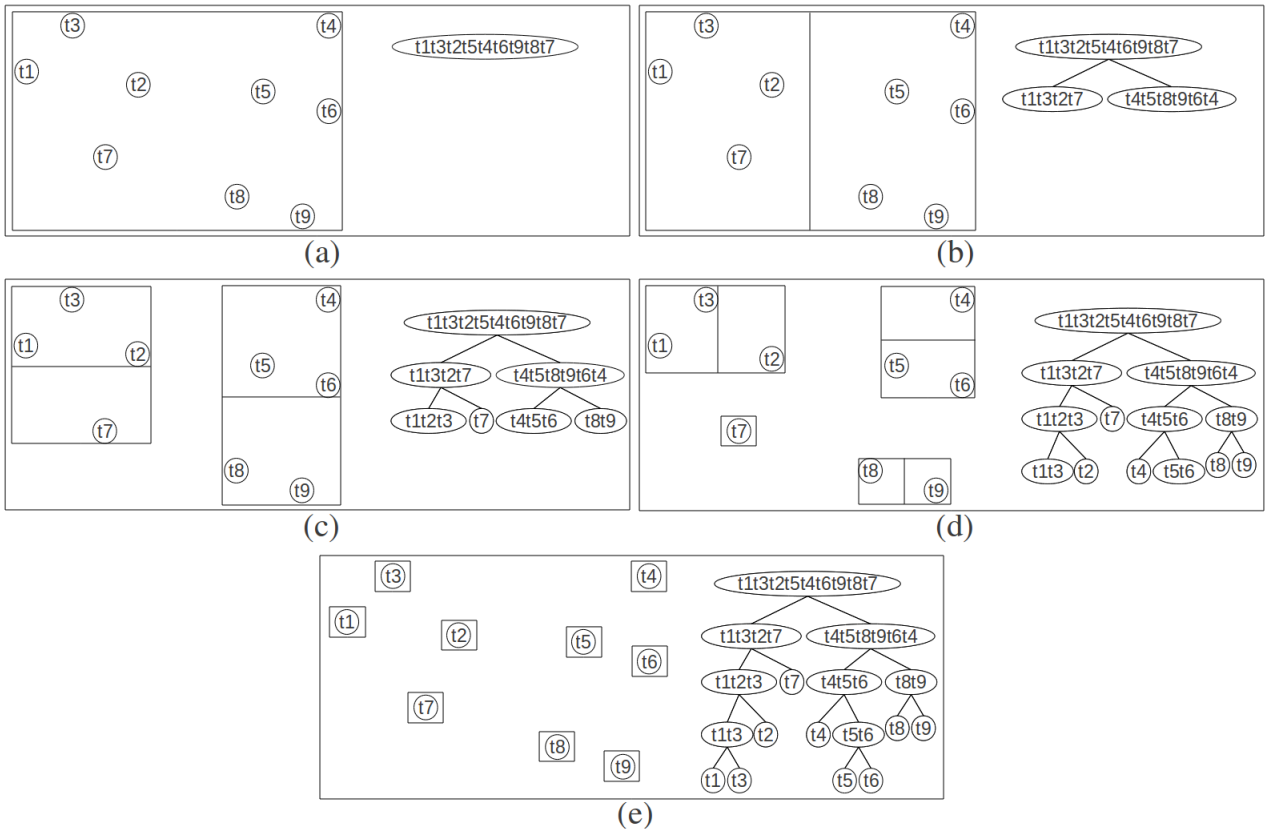

Figura 3.11. Construção da árvore com o algoritmo de partição de regiões.

3.3.1.3 Agrupamento Baseado em TSP

Diferentemente das duas estratégias descritas anteriormente, que seguem uma abordagem *top-down* de construção da árvore, estratégia de agrupamento baseada em TSP segue uma abordagem *bottom-up*, onde a árvore de pacotes vai sendo construída dos nós-folha para a raiz. Os alvos a serem visitados no ambiente são modelados como um grafo completo $G = (V, E)$, onde as posições dos alvos são os vértices V e os pesos das arestas são as distâncias entre os alvos. O algoritmo de agrupamento baseado em TSP consiste em incrementalmente agrupar os vértices do grafo de acordo com o custo do circuito TSP formado com estes vértices. Inicialmente, cada vértice individual compõe um nó-folha da árvore de pacotes. O algoritmo tenta agrupar estes vértices caso o circuito TSP formado por estes vértices tenha um custo inferior a um dado limite de custo. Se alguns alvos puderem ser agrupados, então criam-se novos nós na árvore de pacotes contendo estes vértices agrupados, e estes novos nós se tornam nós-pai dos nós

agrupados. O limite de custo permitido é incrementado, permitindo o agrupamento de circuitos de maior custo, criando mais um nível acima na árvore. O limite de custo permitido é novamente incrementado e o processo se repete até que o limite permita a criação de um circuito TSP que contenha todos os alvos do ambiente, criando o nó-raiz da árvore de pacotes. Como dois ou mais nós podem ser agrupados gerando um nó maior na hierarquia, a árvore gerada não necessariamente será uma árvore binária neste caso, diferentemente das estratégias anteriores.

O Algoritmo 9 descreve os passos da estratégia de agrupamento baseada em TSP. O algoritmo recebe como parâmetro o conjunto de vértices que representa os alvos existentes no ambiente e uma variável Δ que representa o incremento no limite de custo TSP permitido para o agrupamento de vértices (linha 1). Na linha 3, a variável que limita o custo de TSP permitido é atualizada com o valor de Δ. Nas linhas 4 e 5, cada vértice em V se torna um nó-folha da árvore de pacotes. O laço nas linhas 6 a 8 realiza o processo de agrupamento dos nós até chegar a um nó-raiz da árvore, aquele que contenha todos os vértices em V. O método *agrupar*(), na linha 7, recebe como parâmetros a lista de nós já agrupados e a variável $limiteTSP$ que diz o custo do circuito TSP máximo permitido para que dois ou mais nós sejam agrupados. Esse método testa os possíveis agrupamentos entre os nós já gerados na árvore procurando por agrupamentos onde o circuito TSP que contém os vértices desses nós tenham custo menor que o limite permitido. Quando dois ou mais nós podem ser agrupados, o algoritmo cria um nó maior que se torna pai dos nós que foram agrupados na árvore. Em seguida, o limite de TSP é incrementado com o valor Δ (linha 8), para que nós maiores possam ser formados até que se chegue ao nó raiz da árvore, ou seja, o pacote que contém todos os vértices do grafo. Quando este nó for gerado, significa que a árvore está completa e a execução do algoritmo é finalizada.

Algoritmo 9: Algoritmo Agrupamento Baseado em TSP.

```
1 agrupamentoTSP(V, Δ)
2 início
3     limiteTSP ← Δ
4     para cada vértice v_i ∈ V faça
5         criar nó na árvore contendo v_i
6     enquanto não gerou o nó-raiz da árvore faça
7         agrupar(nós existentes, limiteTSP)
8         limiteTSP ← limiteTSP + Δ
9 fim
```

É importante, nesta estratégia, encontrar valor de Δ que esteja relacionado com o número de alvos e com a disposição destes alvos no ambiente. Se o valor de Δ for muito

alto, nas primeiras iterações já se criam agrupamentos com muitos alvos fazendo com que a árvore gerada tenha poucos níveis, e assim, perde-se a flexibilidade do leilão, pois os robôs terão poucas opções de pacotes para dar lances. Caso o valor de Δ seja muito baixo, serão necessárias várias iterações para que o algoritmo consiga construir toda a árvore. Esta árvore resultante possuirá muitos pacotes de alvos e vários níveis. Isto é um problema porque os robôs precisarão computar lances para muitos nós tornando a determinação do vencedor um processo mais caro computacionalmente.

A Figura 3.12 ilustra o funcionamento do algoritmo para a criação da árvore de pacotes utilizando-se o agrupamento baseado em TSP. Na Figura 3.12(a), todos os alvos dão origem aos nós-folha da árvore. O limite de custo de TSP permitido é incrementado e na segunda iteração do algoritmo (Figura 3.12(b)) alguns nós já podem ser agrupados, pois o circuito TSP contendo estes nós é menor que o limite de TSP permitido. Este limite é mais uma vez incrementado e mais nós podem ser agrupados, formando mais um nível na árvore, como pode ser visto na Figura 3.12(c). O processo continua até que seja gerado o nó-raiz da árvore, que contém todos os vértices, como na Figura 3.12(e).

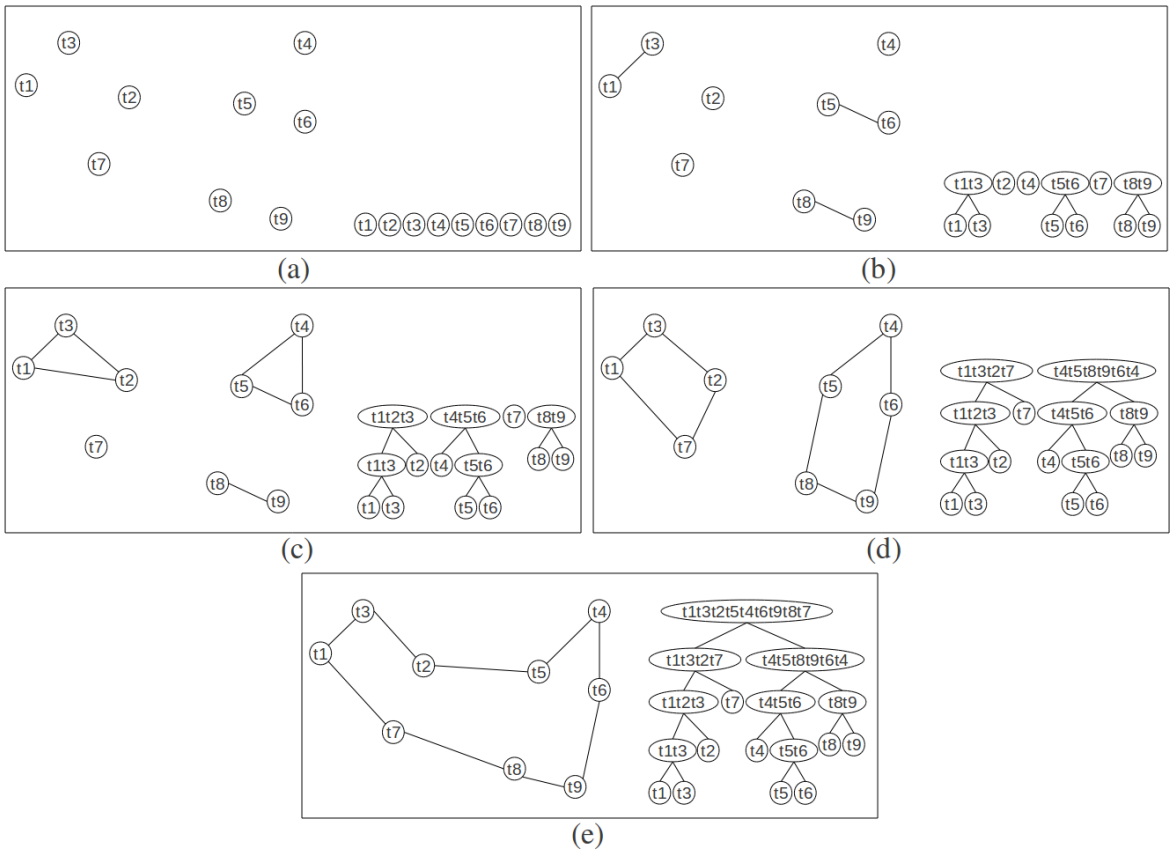

Figura 3.12. Construção da árvore com o algoritmo de agrupamento baseado em TSP.

3.3.2 Ordenação de Itens

Na seção anterior foi discutido como limitar as combinações de alvos permitidas em um leilão combinatório através da criação de árvores de pacotes. Três algoritmos utilizados para agrupar os alvos e criar a árvore foram discutidos. Nesta seção é discutida uma outra estratégia de limitação das possíveis combinações de alvos para as quais os robôs podem dar lances durante o leilão combinatório que não é baseada na criação de árvores de pacotes.

Rothkopf et al. [1998] mostrou que se existe uma boa estratégia para ordenar os itens a serem leiloados, então uma estratégia de limitação das combinações dos itens durante o leilão combinatório é permitir que os compradores ofereçam lances apenas para conjuntos de itens consecutivos de acordo com a sequência. Por meio desta estratégia de leilão combinatório, tanto a computação de lances quanto a determinação do vencedor podem ser realizadas em tempo polinomial.

No presente trabalho, propomos a utilização da estratégia proposta por Rothkopf et al. [1998] para a ordenação dos alvos a serem explorados no ambiente e assim leiloar estes alvos durante um leilão combinatório. O leilão que utiliza esta estratégia possui quatro fases como descritas a seguir.

Na primeira fase do leilão, é preciso que os robôs criem uma sequência para os alvos a serem leiloados. A ordenação dos alvos adotada neste trabalho foi realizada computando-se uma solução candidata de baixo custo para o circuito TSP contendo todos os alvos do ambiente. A ordem de visita dos alvos neste circuito é a ordenação utilizada pela estratégia de leilão combinatório. Para computar o circuito TSP utilizamos a heurística de Inserção do Vizinho mais Distante discutida anteriormente (Algoritmo 3). A Figura 3.13 ilustra o exemplo da criação de uma sequência de alvos para um conjunto de 4 alvos no ambiente. À esquerda temos os alvos a serem explorados no ambiente. Neste exemplo, a sequência $\{t_1,t_2,t_3,t_4,\}$ foi provida pela heurística e será utilizada pelo leilão combinatório.

A sequência provida na primeira fase do leilão será utilizada para limitar as possíveis combinações para as quais os robôs podem dar lances durante o leilão. Na segunda fase do leilão, cada robô precisa computar lance para todas as $\frac{n^2}{2}$ possíveis combinações de segmentos dentro da sequência definida, produzindo uma matriz de lances como esta ilustrada na Figura 3.14. Na primeira linha da matriz, estão os lances para intervalos da sequência que possuem um único alvo, como $[t_1,t_1] = \{t_1\}$, $[t_2,t_2] = \{t_2\}$ e assim por diante. Na segunda linha da matriz, estão os lances para intervalos de dois alvos da sequência, como $[t_1,t_2] = \{t_1,t_2\}$ e $[t_2,t_3] = \{t_2,t_3\}$, por exemplo. Na terceira linha, estão lances para intervalos de três alvos da sequência, como $[t_1,t_3] =$

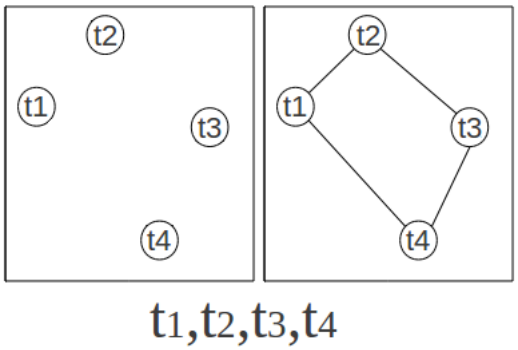

Figura 3.13. Criação da sequência de alvos.

$\{t_1,t_2,t_3\}$ e $[t_2,t_4] = \{t_2,t_3,t_4\}$. O lance para um segmento da matriz é o custo do circuito necessário para o robô visitar todos os alvos pertencentes àquele segmento, e é computado através da heurística de Inserção do Vizinho Mais Distante (Algoritmo 3). Após computar lances para todos estes segmentos, cada robô envia sua matriz de lances para todos os outros robôs.

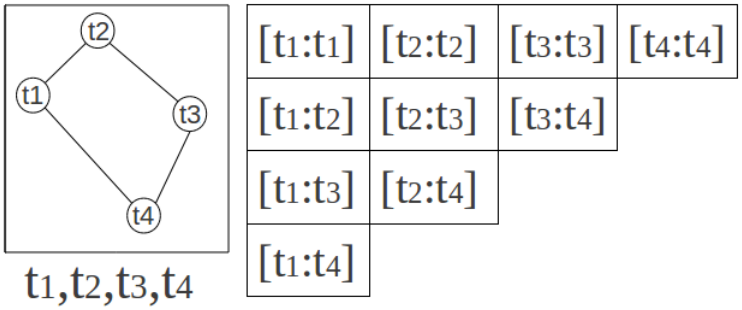

Figura 3.14. Matriz de lances.

A terceira fase do leilão consiste em receber as matrizes de lances de cada robô e criar uma matriz única contendo apenas os lances vencedores para cada segmento da sequência, representados pelas células da matriz. Assim, para cada célula da matriz, o leilão compara os lances enviados por cada robô participante do leilão, gerando uma nova matriz contendo as informações dos lances vencedores, como o valor do lance e o robô que o gerou. Em caso de empates, o robô com menor índice é o vencedor.

A quarta fase do leilão consiste em determinar a alocação final dos segmentos para os robôs. Esse problema é resolvido através de um algoritmo baseado programação dinâmica (Algoritmo 10) proposto por Rothkopf et al. [1998] de forma exata em tempo polinomial. O algoritmo recebe como parâmetro a matriz que contém os lances vencedores em cada célula, gerada na terceira fase do leilão, e o parâmetro n que representa a quantidade de alvos a ser leiloada (linha 1). O vetor *celulas_vencedoras*

é utilizado para guardar a solução parcial da determinação dos vencedores e é iniciada com a célula da matriz de vencedores que representa o intervalo da sequência que começa e termina no primeiro item (linha 3). Para cada iteração do laço das linhas 5−11, o algoritmo opera da seguinte forma: para cada iteração, o algoritmo decide se é mais lucrativo o valor do lance dado para a sequência de itens que começa no primeiro item e tem tamanho r ou os valores dados para as partições dessa sequência de r itens analisados nas iterações anteriores do algoritmo. Assim, na linha 6 o vetor de células vencedoras recebe como solução a célula que representa o intervalo da sequência que começa no primeiro item e termina no item r, que representa o número da iteração corrente do laço. Em seguida o algoritmo verifica se a soma dos lances dados para os fragmentos do intervalo da sequência que começa no item 0 e termina em r é menor que o lance dado para todo o intervalo de 0 a r (linha 8). Se essa condição for verdadeira, o valor de $celulas\_vencedoras(r)$ que contém o intervalo de 0 a r é substituído pelos fragmentos do intervalo, que possuem valor menor (linha 9). Na linha 10, o valor de r é incrementado para que o algoritmo possa analisar um intervalo da sequência maior. O laço se repete até que seja analisado o intervalo que contém todos os n itens da sequência. A solução do algoritmo será composta pelos intervalos da sequência contidos em $celulas\_vencedoras(r)$. Em cada iteração são realizadas $r-1$ comparações. Como o laço se repete até que r seja igual a n, então são realizadas n iterações, esse algoritmo tem complexidade $O(n^2)$.

Algoritmo 10: Algoritmo Programação Dinâmica para Determinação do Vencedor.

```
1  determinarVencedor(matriz, n)
2  início
3      celulas_vencedoras(0) ← celula(intervalo_seq[0:0])
4      r ← 1
5      enquanto (r<n) faça
6          celulas_vencedoras(r) ← celula(intervalo_seq[0:r])
7          para (l ← 1 até r) faça
8              se (valor(celulas_vencedoras(l − 1))+
               valor(intervalo_seq[l:r])<valor(celulas_vencedoras(r))) então
9                  celulas_vencedoras(r) ← celulas_vencedoras(l − 1) ∪
                   celula(intervalo_seq[l:r])
10         r ← r + 1
11 fim
```

A Figura 3.15 mostra um exemplo dos primeiros passos da execução do Algoritmo 10. A Figura 3.15 (a) ilustra a matriz de lances vencedores gerada na terceira etapa do leilão. Na Figura 3.15 (b) o algoritmo toma como intervalo vencedor aquele que começa e termina no primeiro item. Como esse intervalo não pode ser fragmentado

ele permanece como a solução atual. Na Figura 3.15 (c), a variável r é incrementada e agora será analisado o intervalo que começa no primeiro item e termina no segundo, sendo esta solução armazenada nas células vencedoras. Figura 3.15 (d), o algoritmo compara se o valor dado para o intervalo $[t_1{:}t_2] = \{t_1,t_2\}$ é menor que os fragmentos desse intervalo. Como a soma dos valores dados para os fragmentos do intervalo, na Figura 3.15 (e) a solução corrente é atualizada em células vencedoras. Como não há mais fragmentos a serem analisados, há uma nova iteração do algoritmo, onde o valor de r é incrementado para analisar o intervalo $[t_1{:}t_3] = \{t_1,t_2,t_3\}$, como pode ser visto na Figura 3.15 (f). O processo segue sempre a cada iteração comparando o valor dado para uma sequência maior com as combinações dos fragmentos daquela sequência. Ao final, a última posição do vetor *celulas_vencedoras* é quem contém a alocação final, que no exemplo consiste em atribuir o intervalo $[t_1{:}t_3] = \{t_1,t_2,t_3\}$ para o robô r_1 e o intervalo $[t_4{:}t_4] = \{t_4\}$ para r_2, como pode ser visto na Figura Figura 3.15 (j).

No próximo capítulo serão discutidos os experimentos realizados com a aplicação das estratégias descritas neste capítulo e os principais resultados destes experimentos.

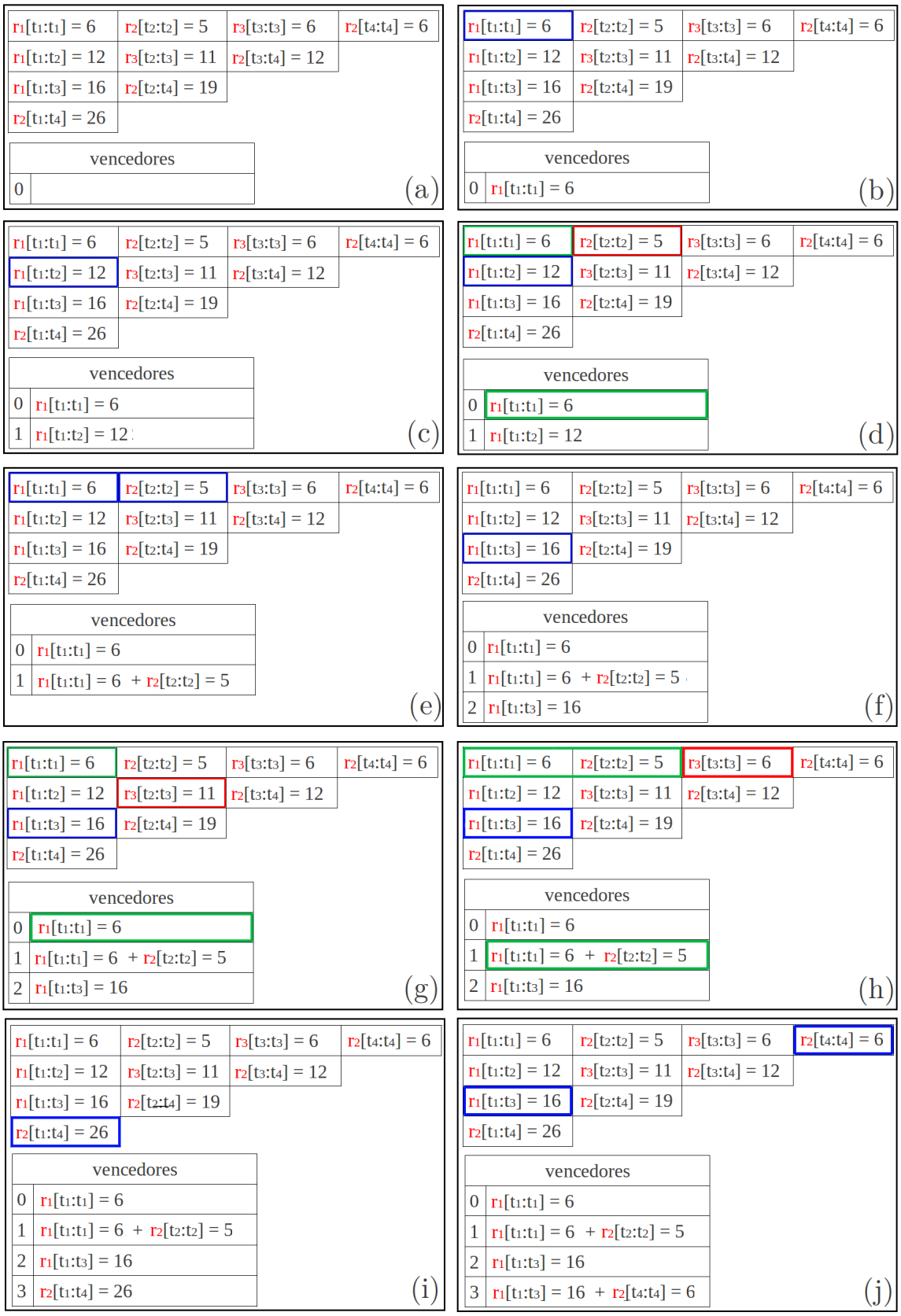

Figura 3.15. Quarta etapa do leilão baseado na ordenação de itens.

Capítulo 4

Experimentos

Este capítulo apresenta os resultados dos experimentos computacionais que foram executados em simulação neste trabalho. Por meio destes experimentos, foi realizado um estudo comparativo dos mecanismos de leilão descritos no capítulo anterior. Os experimentos foram executados utilizando-se a plataforma Player/Stage [Gerkey et al., 2003], em um computador com processador Intel Core i5 e memória RAM de 4 GB DDR3. Os algoritmos foram implementados na linguagem C++ e compilados com o compilador g++ versão 4.4.5.

Os experimentos foram realizados em dois tipos de cenários de forma a comparar a qualidade das soluções providas pelos algoritmos implementados. No primeiro cenário, os alvos a serem explorados foram distribuídos no ambiente segundo uma distribuição aleatória não uniforme. O objetivo dos experimentos neste primeiro cenário foi de avaliar o comportamento dos algoritmos quando os alvos se encontram agrupados em determinadas regiões do ambiente. No segundo cenário investigado, os alvos foram distribuídos uniformemente pelo ambiente. O objetivo deste segundo cenário foi de avaliar a aplicação dos mecanismos em um cenário onde os alvos estão aleatoriamente distribuídos, não possuindo nenhuma relação direta entre suas localizações.

Para cada cenário, foram feitos testes variando-se tanto o número de robôs, como o número de alvos no ambiente. De forma a avaliar o impacto causado pelo aumento do número de robôs no sistema, realizaram-se experimentos com 3, 5 e 9 robôs uniformemente distribuídos pelo ambiente. Com relação ao número de alvos, realizaram-se experimentos com 20 e 40 alvos no ambiente. O ambiente com 20 alvos possui área de 200x200m e o ambiente com 40 alvos possui área de 282x282m. Para cada instância, foram executados 30 simulações variando-se as posições dos alvos, e computada a média e a variância dos resultados obtidos.

Duas métricas foram utilizadas para avaliar os algoritmos implementados. A

primeira delas foi a soma das distâncias percorridas pelos robôs para visitar todos os alvos durante a missão, já que o objetivo dos mecanismos é minimizar essa métrica. A segunda métrica avaliada foi o tempo que os mecanismos consumiram parar fornecer a alocação dos alvos para os robôs.

Nas próximas seções, serão discutidos os resultados dos mecanismos para os dois cenários avaliados. Nos resultados apresentados, denota-se por *SI* a implementação do Leilão de Único Item (Seção 3.1). Denotam-se por *SSI-PRIM* e *SSI-FI* a implementação dos Leilões Sequenciais que utilizam o algoritmo PRIM e a heurística de Inserção do Vizinho Mais Distante, respectivamente (Seção 3.2). Denotam-se por *C-MAX*, *C-REG* e *C-TSP* a implementação dos Leilões Combinatórios baseados em árvores de pacote que utilizam o algoritmo de Corte Máximo, o algoritmo de partição por regiões e o algoritmo de partição por TSP, respectivamente (Seção 3.3.1). Por fim, denota-se por *C-SORT* a implementação do leilão combinatório baseado na ordenação de itens (Seção 3.3.2).

4.1 Distribuição de Alvos Não Uniforme

Nesta seção são apresentados e discutidos os resultados dos mecanismos de alocação quando aplicados ao cenário onde os alvos se encontram não uniformemente distribuídos no ambiente. As instâncias utilizadas nos experimentos neste tipo de cenário foram criadas da seguinte forma: (i) o ambiente foi dividido em 16 regiões de mesmas dimensões; (ii) sorteou-se aleatoriamente 5 das 16 regiões; (iii) os alvos foram distribuídos aleatoriamente apenas nestas 5 regiões sorteadas. A Figura 4.1 ilustra o resultado do processo de criação de uma instância com alvos distribuídos de maneira não uniforme. Neste exemplo, foram sorteadas as regiões 2, 4, 10, 12 e 13 (Figura 4.1(a)). Em seguida, os alvos foram distribuídos escolhendo-se aleatoriamente pontos que estivessem dentro destas regiões (Figura 4.1(b)).

4.1.1 Comparação da Qualidade das Soluções Obtidas

A Figura 4.2 mostra os resultados dos mecanismos de alocação com relação a soma das distâncias percorridas pelos robôs durante a exploração de um ambiente com área de 200x200m e 20 alvos. Os gráficos mostram a média e a variância dos resultados obtidos em 30 execuções utilizando sistemas de exploração com 3, 5 e 9 robôs.

Por meio destes gráficos, pode-se observar que o leilão SI apresenta os piores resultados. Isso se deve ao fato de que, dado que neste mecanismo os robôs computam os lances para um alvo oferecido no leilão independentemente dos outros alvos do

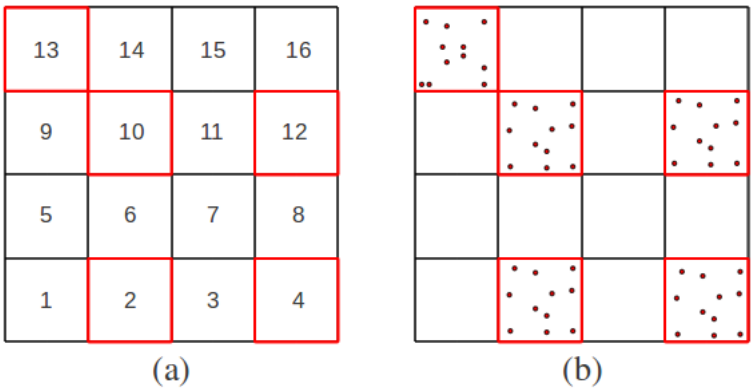

Figura 4.1. Instância com alvos agrupados.

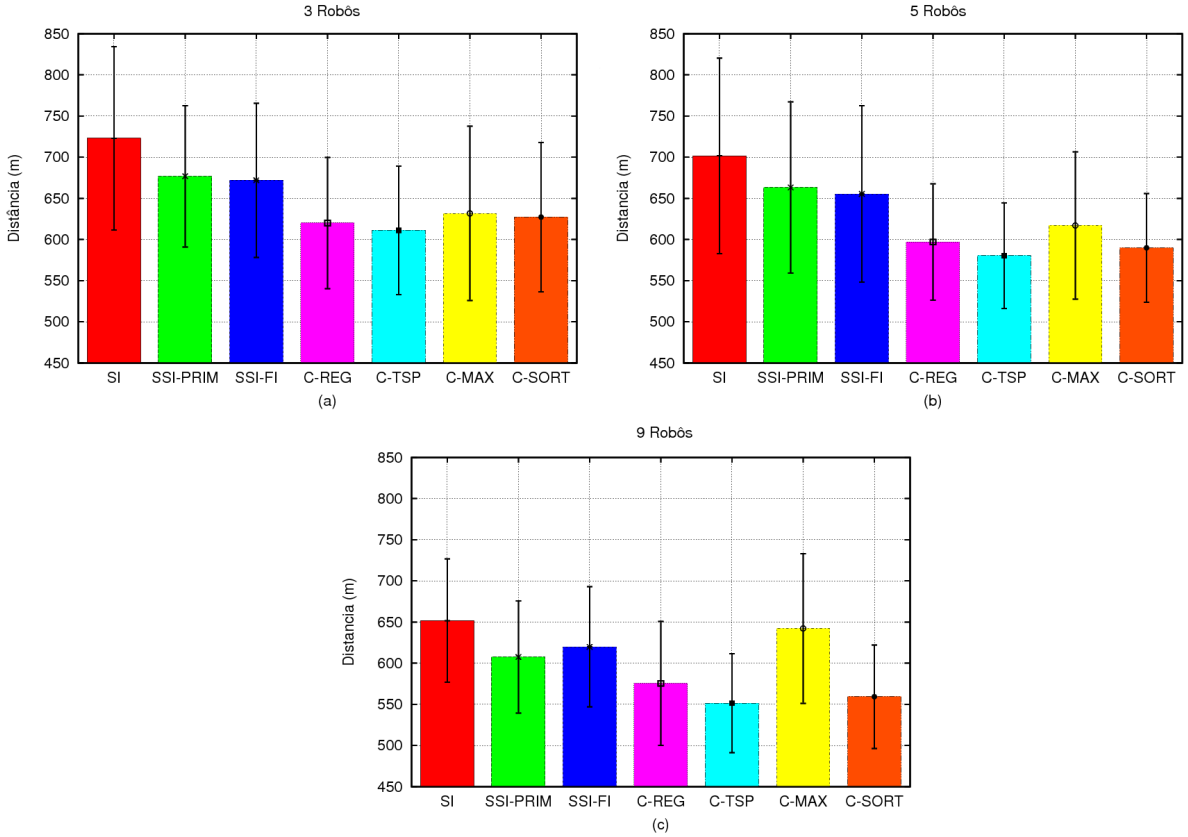

Figura 4.2. Média da soma das distâncias percorridas em ambientes de 200x200m e 20 alvos. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs

ambiente, as relações de proximidade entre os alvos não são consideradas, resultando em alocações de baixa qualidade. O leilão SI também apresenta alta variância. Isso se deve ao fato de que, como o leilão SI é guloso, ele é fortemente dependente da disposição dos alvos no ambiente e da ordem com que estes alvos são leiloados.

Os dois mecanismos de leilão sequencial, SSI-PRIM e SSI-FI, apresentaram resultados equivalentes entre si quando aplicados a este cenário, sendo por sua vez melhores que os resultados providos pelos leilões SI. Os algoritmos SSI-PRIM e SSI-FI tendem a apresentar resultados semelhantes porque, como os alvos se encontram agrupados em determinadas regiões do ambiente, tanto a MST construída por cada robô durante o leilão SSI-PRIM quanto o circuito de menor custo construído durante o leilão SSI-FI, tendem a conter o mesmo conjunto de alvos, que são aqueles que estão agrupados em determinadas regiões do ambiente. Assim, quando os robôs transformam suas MSTs em circuitos para realizar a visita dos alvos, eles geralmente obtêm o mesmo circuito obtido durante o leilão SSI-FI.

Nos gráficos da Figura 4.2 observa-se que os leilões combinatórios, com exceção do leilão C-MAX, obtiveram resultados superiores em termos de minimização das distâncias percorridas, quando comparados aos demais tipos de leilão. Durante o leilão C-MAX, a árvore é construída subdividindo os pacotes em duas partições de forma que a soma dos custos das arestas que ligam os vértices em partições diferentes seja alto, enquanto a soma dos custos das arestas que ligam os vértices dentro de uma mesma partição sejam baixas. No entanto, este fato não implica que os vértices dentro de uma partição formem um circuito de baixo custo durante a exploração. Assim, a árvore de pacotes criada durante o leilão C-MAX nem sempre é constituída por pacotes de alvos que apresentam alta sinergia entre si, o que faz com que C-MAX obtenha soluções de baixa qualidade.

Pode-se ainda observar por meio dos gráficos que, com o aumento do número de robôs no sistema, diminui-se a soma das distâncias percorridas nas soluções providas por todos os mecanismos de alocação. Isso se deve ao fato de que, como mais robôs estão distribuídos no ambiente, alguns deles podem estar mais perto das regiões que contêm os alvos agrupados, precisando viajar menos para alcançá-los.

A Figura 4.3 mostra a comparação dos resultados dos algoritmos quando aplicados a um ambiente com 282x282m e 40 alvos a serem explorados. O leilão SI continuou obtendo resultados piores em comparação com os demais leilões. Os leilões SSI-PRIM e SSI-FI novamente apresentaram resultados equivalentes. O leilão combinatório C-MAX não apresentou bons resultados em relação aos demais mecanismos. O leilão combinatório C-TSP foi aquele que apresentou melhores resultados dentre todos os mecanismos.

Por meio dos gráficos da Figura 4.3 também pode-se observar que, com o aumento no número de robôs no sistema, diminui-se a variância nos resultados. Isso se deve ao fato de que quando há poucos robôs no ambiente, dependendo das posições dos alvos no ambiente, os robôs terão de viajar mais para visitar todos os alvos, enquanto em outros casos, os robôs já se encontram próximos das regiões onde os alvos estão agrupados, causando a alta variância. Quando existem muitos robôs espalhados pelo ambiente, a forma com que os alvos foram distribuídos não influencia de forma significativa no resultado final, já que há uma maior probabilidade de algum robô estar localizado próximo a uma região onde os alvos estão situados.

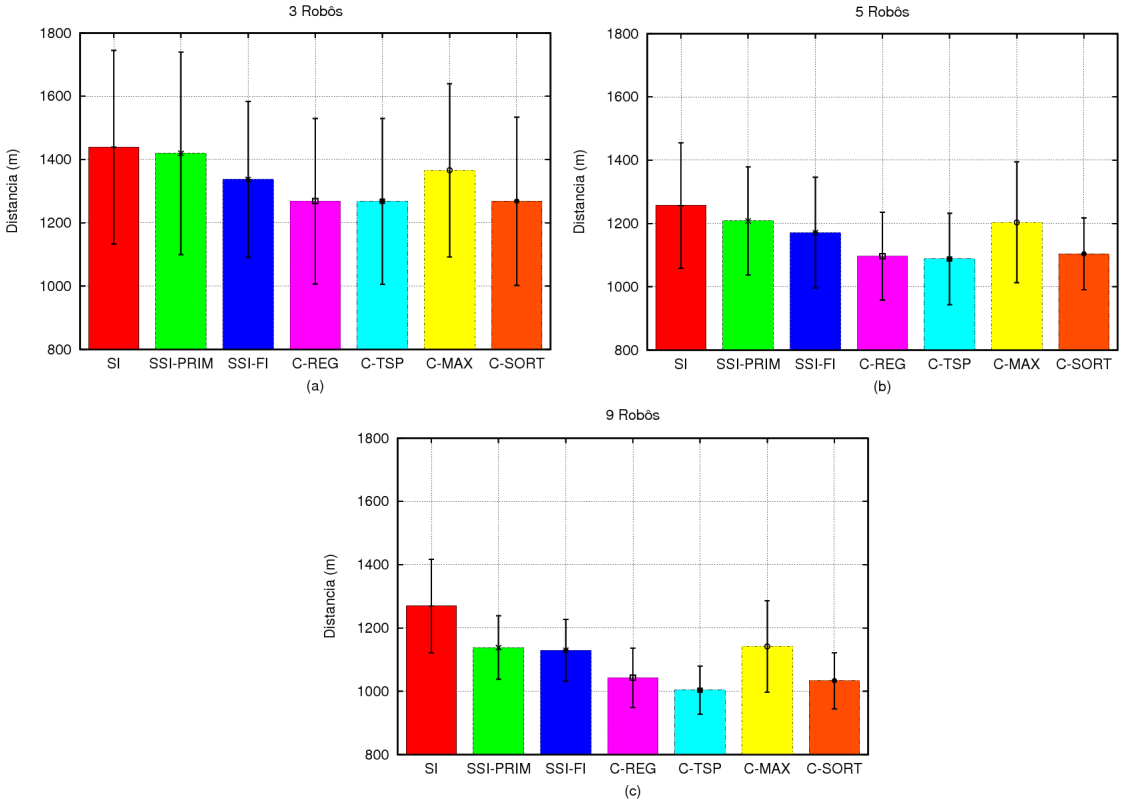

Figura 4.3. Média da soma das distâncias percorridas em ambientes de 282x282m e 40 alvos. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs

A Figura 4.4 mostra a alocação provida pelos algoritmos em um cenário com 5 robôs, 282x282m de área e 40 alvos distribuídos de maneira não uniforme no ambiente. O leilão SI (Figura 4.4(b)) obteve a alocação de maior custo, utilizando todos os robôs na missão. O leilão SSI-PRIM (Figura 4.4(c)) alocou os alvos apenas para 4 robôs e foi um pouco melhor que o leilão SSI-FI (Figura 4.4(d)), que utilizou todos os robôs. Os leilões combinatórios C-REG (Figura 4.4(e)), C-TSP (Figura 4.4(f)) e C-SORT (Figura 4.4(h)) foram capazes de prover a alocação com menor custo que os demais, utilizando

apenas 3 dos 5 robôs disponíveis. C-MAX (Figura 4.4(g)) não foi capaz de encontrar uma solução de boa qualidade em relação aos demais leilões.

4.1.2 Comparação do Tempo de Execução

A Figura 4.5 mostra o tempo (em escala logarítmica) consumido pelos mecanismos para computar a alocação de 20 e 40 alvos para 3, 5 e 9 robôs, respectivamente. Os leilões SI e SSI-PRIM obtiveram os menores tempos de computação, seguidos pelo leilão SSI-FI. Dentre os leilões combinatórios, C-MAX e C-REG apresentaram baixo consumo de tempo, que foi de menos de 1 segundo para computar a alocação de 40 alvos para os robôs.

Os leilões combinatórios C-TSP e C-SORT foram aqueles que consumiram um tempo maior para computar a alocação. No leilão C-TSP, a montagem da árvore tem custo computacional alto, pois utiliza a heurística polinomial de Inserção do Vizinho mais Distante para tentar agrupar os alvos em pacotes maiores. Como visto no capítulo 3, essa heurística tem custo quadrático. Durante o processo de montagem da árvore, é preciso executar essa heurística várias vezes, testando-se o limite de TSP permitido para o agrupamento até que se consiga gerar a raiz da árvore.

O leilão C-SORT foi aquele que apresentou um maior consumo de tempo para computação da alocação. Isto se deve ao alto custo computacional para um robô computar a matriz de lances. Como cada robô precisa computar lance para $O(n^2)$ células dessa matriz, e para computar cada lance, é utilizada a heurística de Inserção do Vizinho Mais Próximo, que tem custo $O(n^2)$, então o custo total para computar lances para todas as células da matriz é da ordem $O(n^4)$.

4.2 Distribuição de Alvos Uniforme

Nesta seção, são discutidos os experimentos realizados aplicando-se os mesmos mecanismos de leilão ao cenário onde a distribuição dos alvos foi uniforme. O objetivo deste conjunto de experimentos foi avaliar a qualidade da solução provida pelos mecanismos em um cenário onde os alvos não possuem nenhuma relação de proximidade pré-estabelecida, como nos experimentos anteriores, onde os alvos estavam agrupados em determinadas regiões do ambiente. Aplicações reais como vigilância e monitoramento ou missões de busca e salvamento, geralmente seguem este modelo, pois os robôs precisam vistar pontos em diferentes locais do ambiente sem nenhuma ordem de posicionamento definida.

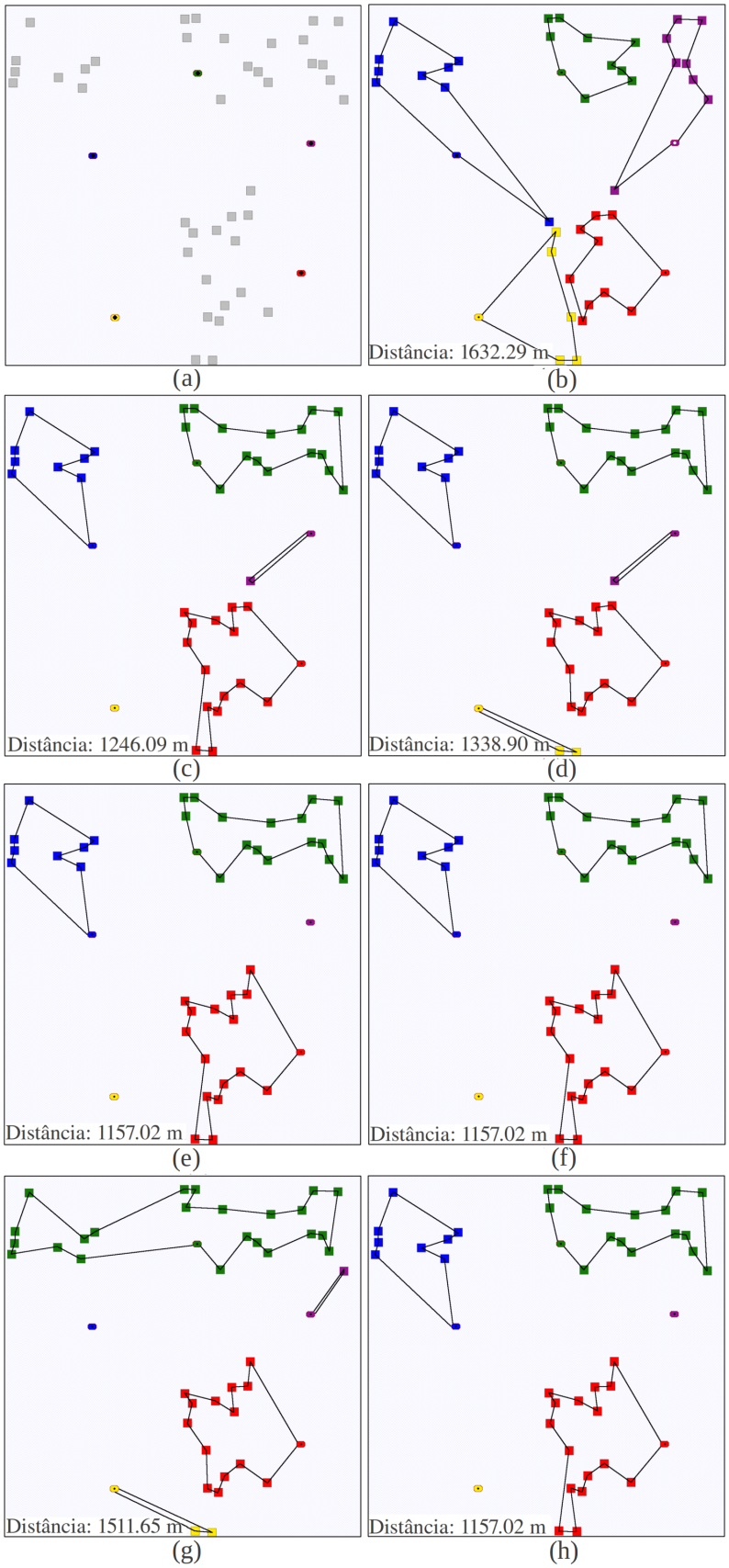

Figura 4.4. Alocações providas pelos leilões em um cenário com 5 robôs, 282x282m de área e 40 alvos. (a) Configuração inicial. (b) SI. (c) SSI-PRIM. (d) SSI-FI. (e) C-REG. (f) C-TSP. (g) C-MAX. (h) C-SORT

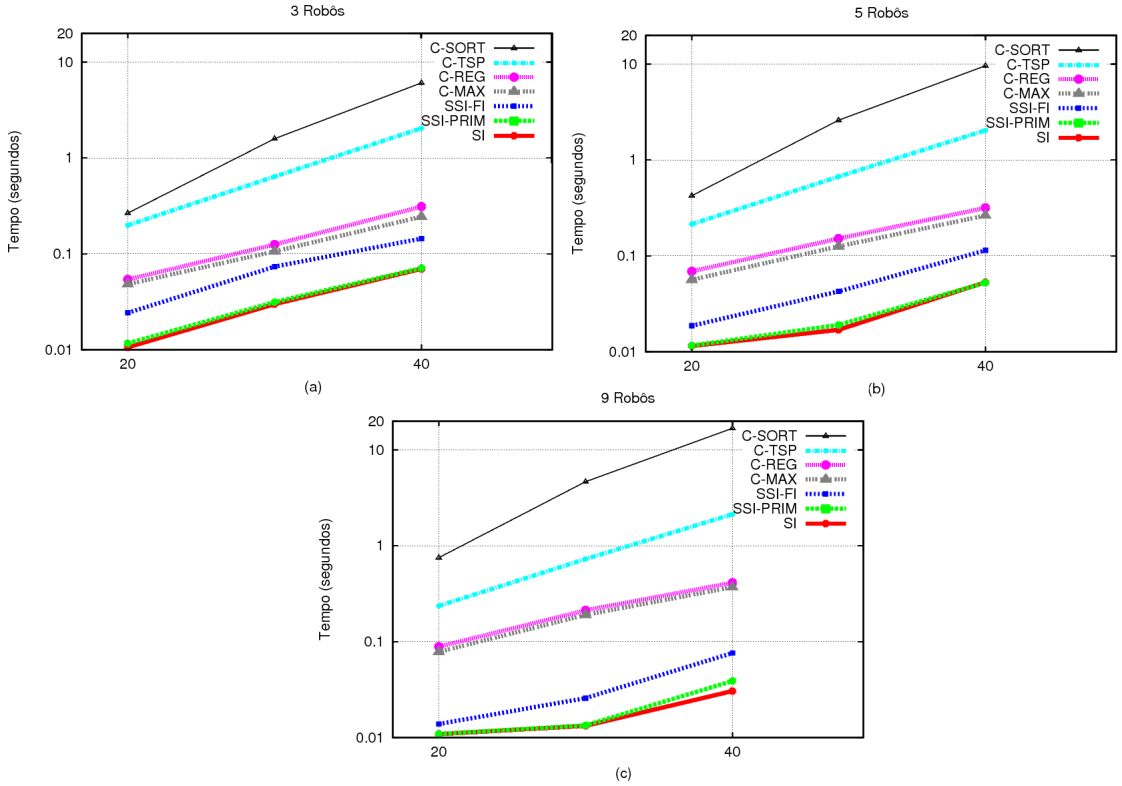

Figura 4.5. Tempo consumido pelos leilões. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs.

4.2.1 Comparação da Qualidade das Soluções Obtidas

A Figura 4.6 mostra o resultado dos mecanismos de alocação com relação a soma das distâncias percorridas em um ambiente de dimensões 200x200m com 20 alvos. Pode-se observar que nos casos onde os alvos estão distribuídos aleatoriamente, os leilões combinatórios também são superiores. O leilão combinatório C-MAX mais uma vez não apresentou bons resultados, além de ter alta variância.

Por meio dos gráficos pode-se observar que neste cenário, diferentemente do cenário descrito na seção anterior, o leilão SSI-FI apresentou resultados melhores do que o leilão SSI-PRIM. Isso se deve ao fato de que como neste cenário os alvos não estão agrupados, a MST criada pelos robôs durante o leilão SSI-PRIM geralmente contém alvos que estão muito distantes da posição inicial do robôs, já que os alvos se encontram bem distribuídos no ambiente. Assim os robôs precisam realizar um trajeto de alto custo para visitar todos os alvos contidos em suas MSTs e ainda retornar as suas posições iniciais. Como no leilão SSI-FI os robôs vão construindo circuitos de menor custo com os alvos oferecidos no leilão já considerando o custo da volta para sua posição inicial, o custo da alocação final tende a ser menor.

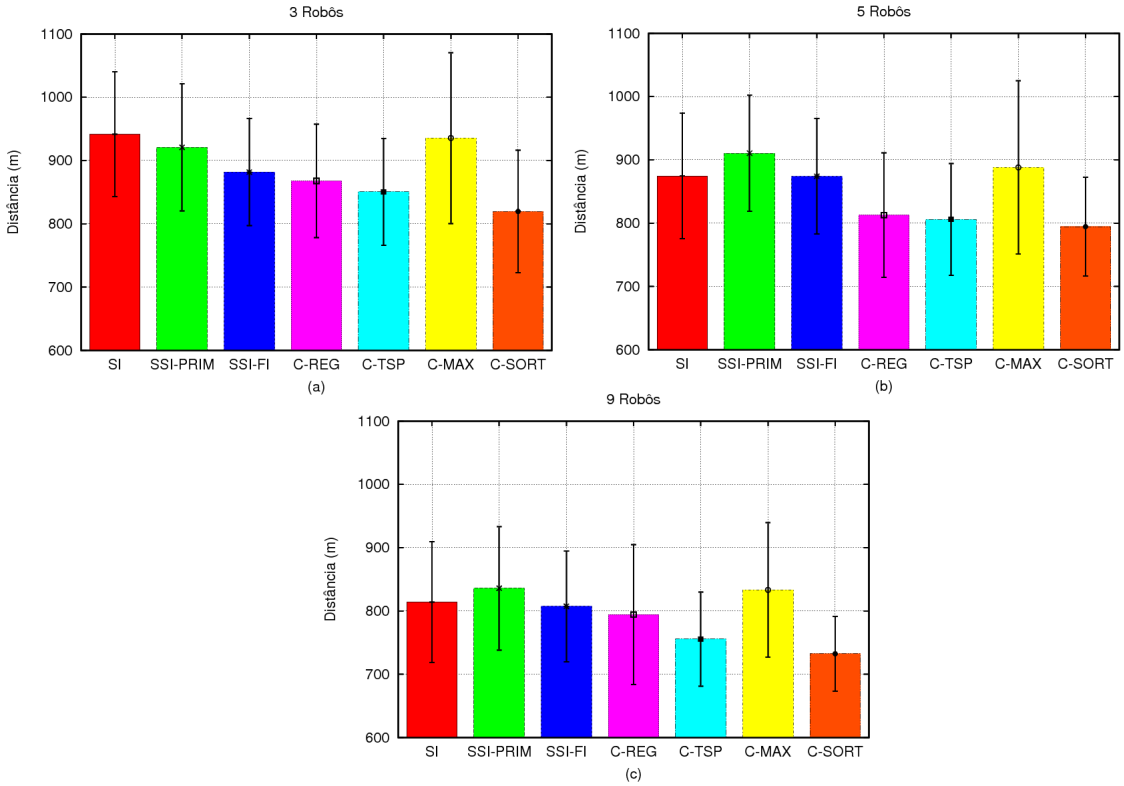

Figura 4.6. Média da soma das distâncias percorridas em ambientes de 200x200m e 20 alvos. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs

A Figura 4.7 mostra os resultados obtidos pelos algoritmos para o caso onde o ambiente possui 282x282m e 40 alvos. Neste gráficos, mais uma vez pode ser observado que os leilões combinatórios C-TSP e C-SORT apresentam soluções com menor custo. Isso se deve ao fato de que nestes dois tipos de leilão, as possíveis combinações de alvos para as quais os robôs oferecem lances são determinadas pelo custo do circuito TSP formado pelos alvos em cada combinação. Assim, os robôs já consideram o circuito final que eles terão de percorrer caso vença os pacotes leiloado. Novamente, o leilão combinatório C-MAX obteve os piores resultados.

Vale a pena ressaltar também o fato de o leilão SSI-FI apresentar bons resultados em relação os demais, chegando próximo aos resultados fornecidos pelo leilão combinatório C-REG. Este fato fica evidente principalmente nos casos onde há 9 robôs no time (Figura 4.7(c)). Isso acontece porque no leilão SSI-FI os robôs vão construindo, ao longo do leilão, circuitos TSP de baixo custo com os alvos oferecidos. Quando há mais robôs distribuídos no ambiente, estes criam, durante o leilão, ciclos de baixo custo locais, melhorando a qualidade dos resultados.

O leilão combinatório C-REG obteve resultados piores do que nos experimentos

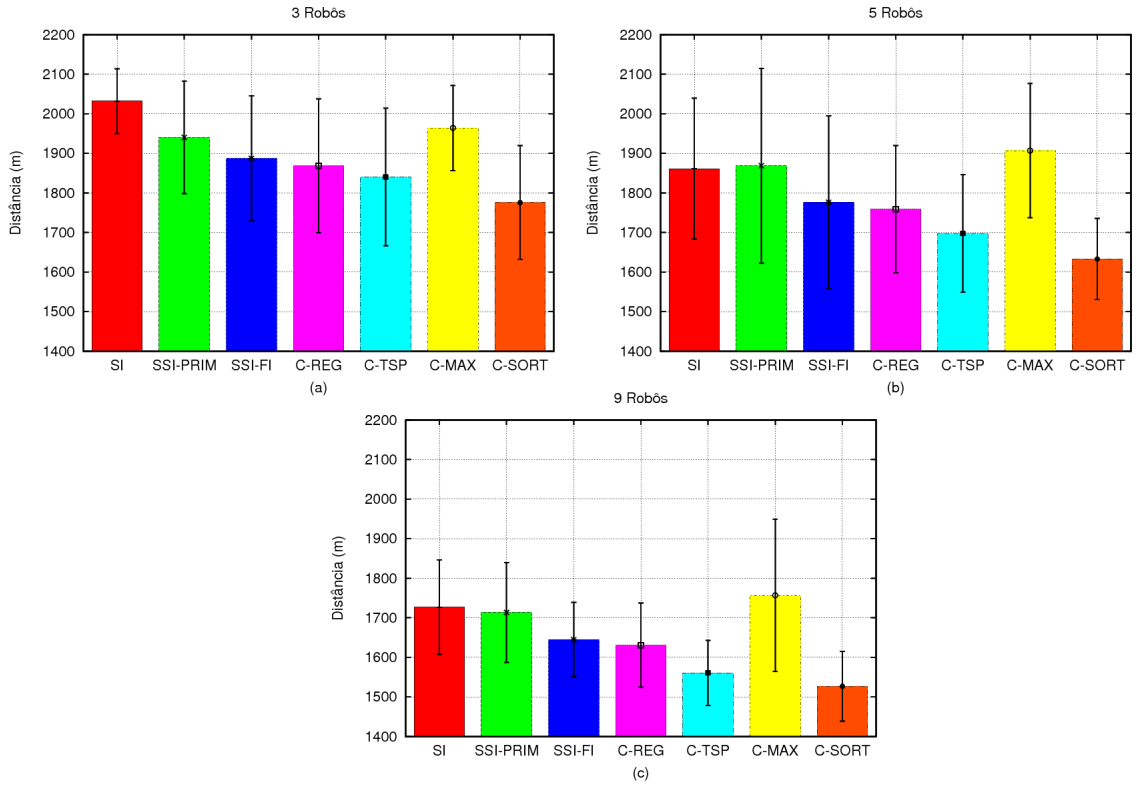

Figura 4.7. Média da soma das distâncias percorridas em ambientes de 282x282m e 40 alvos. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs.

descritos na seção anterior, onde os alvos estavam agrupados. Isso se deve ao fato de que agora os alvos estão dispersos por todo o ambiente e não mais agrupados em regiões menores. Dessa forma a árvore de pacotes criada durante o leilão C-REG não consegue agrupar alvos com alta sinergia como no cenário anterior.

A Figura 4.8 mostra a alocação provida pelos algoritmos em um cenário com 5 robôs, 282x282m de área e 40 alvos uniformemente distribuídos no ambiente. Novamente o leilão SI (Figura 4.8(b)) forneceu a solução com pior custo dentre os leilões. Diferentemente do cenário onde os alvos estavam agrupados (Figura 4.4), agora o leilão SSI-FI (Figura 4.8(d)) obteve uma solução melhor que o leilão SSI-PRIM ((Figura 4.8(c))). Os leilões combinatórios forneceram as melhores soluções, com exceção do C-MAX ((Figura 4.8(g)). O leilão combinatório C-TSP (Figura 4.8(f)) foi aquele que encontrou a melhor alocação dentre todos os leilões.

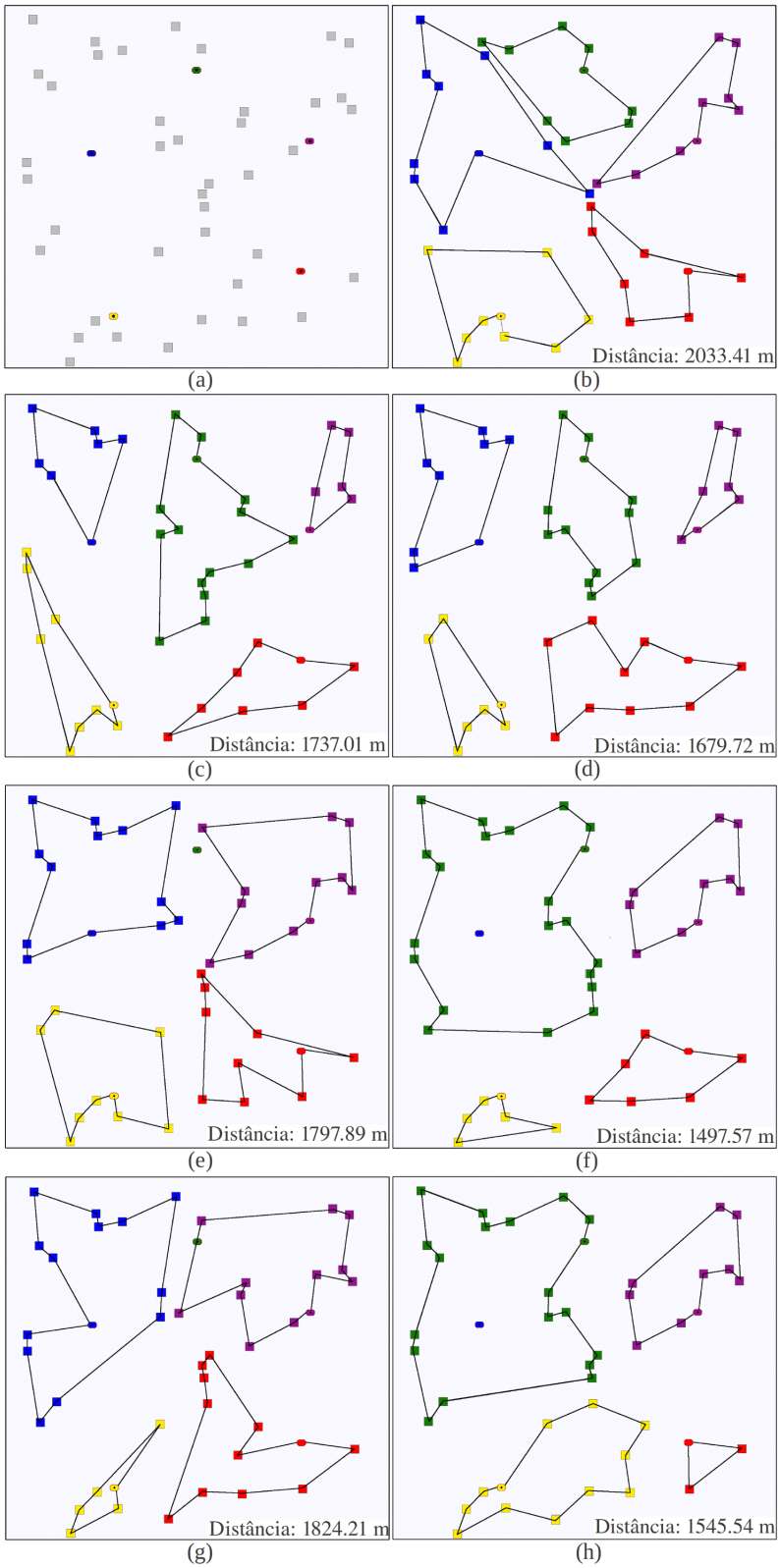

Figura 4.8. Alocações providas pelos leilões em um cenário com 5 robôs, 282x282m de área e 40 alvos. (a) Configuração inicial. (b) SI. (c) SSI-PRIM. (d) SSI-FI. (e) C-REG. (f) C-TSP. (g) C-MAX. (h) C-SORT

4.2.2 Comparação do Tempo de Execução

A Figura 4.9 mostra o tempo (em escala logarítmica) consumido pelos mecanismos para computar a alocação de 20 e 40 alvos para 3, 5 e 9 robôs, respectivamente. Pode-se observar que o tempo de execução dos leilões aplicados ao cenário com distribuição uniforme dos alvos foi semelhante ao resultado dos experimentos aplicados ao cenário onde os alvos estavam agrupados, descritos na seção anterior. Isso acontece porque o tempo de computação da alocação não depende da distribuição dos alvos ou das dimensões do ambiente, estando relacionado apenas ao número de alvos a serem alocados para os robôs e à quantidade de robôs no sistema.

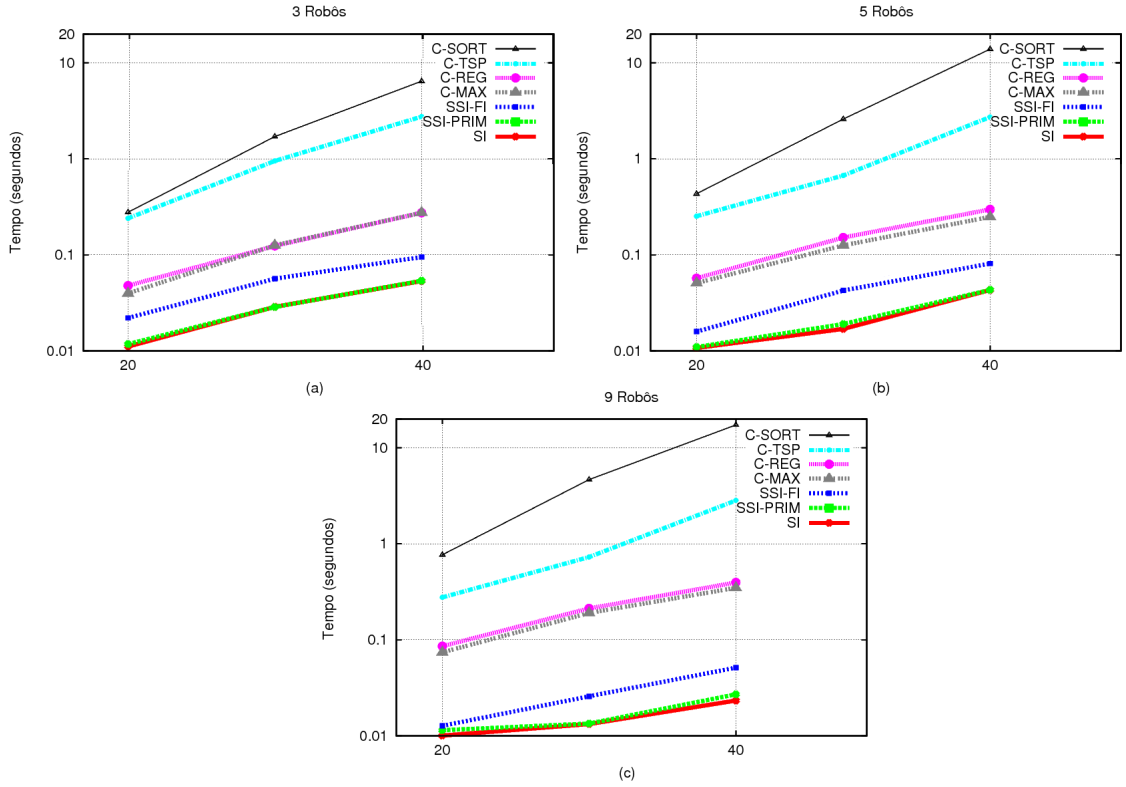

Figura 4.9. Tempo consumido pelos leilões. (a) Time com 3 robôs. (b) Time com 5 robôs. (c) Time com 9 robôs.

Capítulo 5

Conclusões e Trabalhos Futuros

O problema de exploração de ambientes utilizando múltiplos robôs autônomos é considerado um dos problemas fundamentais da robótica móvel. Diversas tipos de problema de exploração utilizando robôs móveis têm sido investigadas na literatura. No presente trabalho foi investigado o problema de exploração do tipo observação de múltiplos alvos, onde diversos pontos ou alvos virtuais precisam ser visitados pelos robôs de forma a minimizar a soma das distâncias viajadas e consequentemente os minimizar os custos da missão.

A utilização de vários robôs em uma missão de exploração apresenta várias vantagens em comparação com os sistemas de único robô, como maior rapidez na exploração, maior robustez e maior confiabilidade. No entanto, a utilização de múltiplos robôs no sistema introduz o problema de coordenação destes robôs de forma a maximizar a utilização dos recursos e minimizar conflitos durante a exploração.

No presente trabalho, investigamos a utilização de diversos mecanismos de leilão para a coordenação dos robôs durante a exploração de ambientes previamente conhecidos. Estudamos também a aplicação de diversas heurísticas para a criação de lances durante os leilões de forma a investigar a qualidade das soluções fornecidas por estes mecanismos de coordenação.

A principal contribuição deste trabalho é a investigação de duas estratégias de leilões combinatórios para a limitar as possíveis combinações de pacotes que os robôs poderiam dar lances de forma a tornar tratáveis tanto o problema de computação de lances quanto o problema de determinação dos vencedores. Com relação a primeira estratégia, que é baseada na construção de árvores de pacotes, foram propostos dois algoritmos para criação das árvores, C-REG, C-TSP, que foram comparadas com o algoritmo C-MAX. Por meio dos experimentos foi possível observar que C-MAX não obteve bons resultados em nenhum dos cenários em que foram realizados os testes.

C-REG e C-TSP apresentaram resultados equivalentes entre si nos cenários onde os alvos estão agrupados no ambiente, enquanto que nos cenários onde os alvos foram uniformemente distribuídos, C-TSP obteve resultados superiores. No entanto, C-TSP apresenta um custo computacional maior que C-REG.

A segunda estratégia foi o leilão combinatório C-SORT, que obteve melhores resultados dentre todos os mecanismos investigados. A diferença de qualidade entre este mecanismo e os demais foi ainda maior nos cenários onde os alvos forma uniformemente distribuídos. No entanto, este mecanismo apresentou um alto consumo de tempo para computar a alocação.

Em relação ao custo-benefício da qualidade da solução e do custo computacional, os leilões combinatórios C-REG e C-TSP se mostraram melhores que os demais mecanismos, pois apresentam soluções de alta qualidade e baixo custo computacional. No entanto, para missões de exploração em ambientes de grandes dimensões, onde a missão irá demorar um tempo considerável para que os robôs visitem todos os alvos do ambiente, pode ser mais interessante utilizar um mecanismo de alocação que dure um tempo maior, porém seja capaz de fornecer uma solução capaz de economizar algumas horas de exploração propriamente dita, como é o caso do leilão C-SORT.

Apesar das conclusões obtidas com a implementação dos leilões aqui investigados, vale ressaltar que o presente trabalho pode ser complementado em diversos aspectos. Alguns deles são descritos a seguir.

Neste trabalho investigamos três formas de construir a árvore de pacotes durante o leilão combinatório. Outros algoritmos de agrupamento de alvos podem ser avaliados de forma a verificar se são capazes de fornecer uma alocação de menor custo.

No presente trabalho, uma das formas utilizadas para resolver o problema de intratabilidade do leilão combinatório foi através da limitação das possíveis combinações de pacotes através da construção de árvore de pacotes disjuntos. A restrição de disjunção de pacotes obriga que um mesmo alvo só possa pertencer a dois pacotes diferentes se um pacote estiver contido no outro. Trabalhos futuros podem investigar leilões combinatórios que permitem a construção de árvore de pacotes não disjuntos, permitindo maior flexibilidade na combinação dos alvos. No entanto, o problema de determinação do vencedor neste tipo de leilão seria mais complexo.

Além disso, a presente pesquisa pode ser complementada por meio do estudo de funções de utilidade mais robustas para serem aplicadas a problemas mais próximos da realidade. Para ambientes conhecidos e que contém obstáculos, como portas e corredores, algoritmos como o A* e suas variações são capazes de computar lances considerando estes obstáculos. Por outro lado, em ambientes desconhecidos, onde há incertezas nos custos para se alcançar os alvos, se faz necessário a investigação

de funções de custo que sejam capazes de lidar com estas incertezas, bem como as incertezas relacionadas a atuação do robô, permitindo que a computação de lances durante o leilão seja capaz de acomodar estas incertezas.

Neste trabalho, assumimos que a comunicação é perfeita e ilimitada. No entanto, este cenário de comunicação perfeita não se configura em sistemas reais. Dessa forma se faz necessário investigar o impacto que a limitação das comunicações causa nos mecanismos de leilão propostos neste trabalho.

Referências Bibliográficas

Basilico, N. & Amigoni, F. (2011). Exploration strategies based on multi-criteria decision making for search and rescue autonomous robots. In *Proc. of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 99−106, Taipei, Taiwan.

Berhault, M.; Huang, H.; Keskinocak, P.; Koenig, S.; Elmaghraby, W.; Griffin, P. & Kleywegt, A. (2003). Robot exploration with combinatorial auctions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1957−1962, Las Vegas, USA.

Brumitt, B. L. & Stentz, A. (1996). Dynamic mission planning for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2396−2401, Minneapolis, USA.

Burgard, W.; Moors, M.; Fox, D.; Simmons, R. & Thrun, S. (2000). Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 476−481, San Francisco, USA.

Burgard, W.; Moors, M.; Stachniss, C. & Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21:376−386.

Campbell, A. & Wu, A. (2011). Multi-agent role allocation: issues, approaches, and multiple perspectives. *Autonomous Agents and Multi-Agent Systems*, 22:317−355.

Cao, Y. U.; Fukunaga, A. S. & Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7−27.

Cavalcante, R. C.; Bittencourt, I. I.; da Silva, A. P.; Silva, M.; Costa, E. & Santos, R. (2012a). A survey of security in multi-agent systems. *Expert Systems with Applications*, 39:4835−4846.

Cavalcante, R. C.; Noronha, T. F. & Chaimowicz, L. (2012b). A local search approach for improving multi-robot routing in exploration missions. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)*, pp. 85−90. Fortaleza, Brazil.

Chaimowicz, L.; Sugar, T.; Kumar, V. & Campos, M. F. M. (2001). An architecture for tightly coupled multi-robot cooperation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2992−2997, Seoul, Korea.

Colares, R. & Chaimowicz, L. (2012). Planning for simultaneous localization and mapping using topological information. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)*, pp. 214−219, Fortaleza, Brazil.

Cramton, P.; Shoham, Y. & Steinberg, R. (2005). *Introduction to Combinatorial Auctions*. MIT Press.

de Hoog, J.; Cameron, S. & Visser, A. (2009). Role-based autonomous multi-robot exploration. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pp. 482−487, Los Alamitos, USA.

Dias, M. B.; Zlot, R.; Kalra, N. & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. In *Proceedings of the IEEE*, volume 94, pp. 1257−1270.

Dissanayake, M. W. M. G.; Newman, P.; Clark, S.; Durrant-Whyte, H. F. & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17:229−241.

Dudek, G.; Jenkin, M. & Milios, E. (2002). A taxonomy for multirobot systems. *Autonomous Robots*, 3:375−397.

Ekici, A.; Keskinocak, P. & Koenig, S. (2009). Multi-robot routing with linear decreasing rewards over time. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 958−963, Kobe, Japan.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22:46−57.

Farinelli, A.; Farinelli, R.; Iocchi, L. & Nardi, D. (2004). Multi-robot systems: A classification focused on coordination. *IEEE Transactions on Systems, Man and Cybernetics*, 34:2015−2028.

Fox, D.; Ko, J.; Konolige, K.; Limketkai, B.; Schulz, D. & Stewart, B. (2006). Distributed multi-robot exploration and mapping. In *Proceedings of the IEEE*, volume 94, pp. 1325−1339.

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

Gerkey, B. P. & Matarić, M. J. (2002). Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18:758−768.

Gerkey, B. P. & Matarić, M. J. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pp. 3862−3868, Taipei, Taiwan.

Gerkey, B. P. & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. In *International Journal of Robotics Research*, volume 23, pp. 939−954.

Gerkey, B. P.; Vaughan, R. & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th International Conference on Advanced Robotics*, pp. 317−323.

Goebel, P. (2006). http://www.pirobot.org.

Hanna, H. (2005). Decentralized approach for multi-robot task allocation problem with uncertain task execution. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 535−540, Edmonton, Canada.

Haumann, A. D.; Listmann, K. D. & Willert, V. (2010). Discoverage: A new paradigm for multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 929−934, Anchorage, USA.

Jap, S. D. (2002). Online reverse auctions: Issues, themes, and prospects for the future. In *Journal of the Academy of Marketing Science*, volume 30, pp. 506−525.

Johnson, D. S. & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In *Local Search in Combinatorial Optimization*, pp. 215−310.

Kaleci, B.; Parlaktuna, O.; Ozkan, M. & Kirlik, G. (2010). Market-based task allocation by using assignment problem. In *IEEE International Conference on Systems Man and Cybernetics (SMC)*, pp. 135−141, Chicago, USA.

Karp, R. (1977). Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Mathematics of Operations Research*, pp. 209−224.

Ko, J.; Stewart, B.; Fox, D.; Konolige, K. & Limketkai, B. (2003). A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pp. 3232−3238, Las Vegas, USA.

Koenig, S.; Keskinocak, P. & Tovey, C. (2010). Progress on agent coordination with cooperative auctions. In *Proceedings of AAAI Conferece on Artificial Intelligence*, pp. 1713−1717, Atlanta, USA.

Koenig, S.; Tovey, C.; Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A. J.; Meyerson, A. & Jain, S. (2006). The power of sequential single-item auctions for agent coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1625−1629, Boston, USA.

Kuipers, B. & Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47−63.

Lagoudakis, M. G.; Berhault, M.; Koenig, S.; Keskinocak, P. & Kleywegt, A. J. (2004). Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pp. 698−705, Sendai, Japan.

Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Koenig, S.; Tovey, C.; Meyerson, A. & Jain, S. (2005). Auction-based multi-robot routing. In *Proceedings of the International Conference on Robotics: Science and Systems*, pp. 343−350, Cambridge, USA.

Marjovi, A.; Nunes, J. G.; Marques, L. & de Almeida, A. (2009). Multi-robot exploration and fire searching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1929−1934, St. Louis, USA.

Matarić, M. J.; Nilsson, M. & Simsarin, K. T. (1995). Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pp. 556−561, Pittsburgh, USA.

Nanjanath, M. & Gini, M. (2006). Dynamic task allocation for robots via auctions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2781−2786, Orlando, USA.

Rothkopf, M. H.; Pekec, A. & Harstad, R. M. (1998). Computationally manageable combinatorial auctions. *Management Science*, 44.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1−54.

Sariel, S.; Balch, T. & Erdogan, N. (2009). Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. *IEEE/ASME Transactions on Mechatronics*, 14:198−206.

Shiroma, P. M. & Campos, M. F. M. (2009). Comutar: A framework for multi-robot coordination and task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4817−4824, St. Louis, USA.

Simmons, R.; Apfelbaum, D.; Burgard, W.; Fox, D.; M., M.; Thrun, S. & Younes, H. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 852−858, Austin, USA.

Soares, M. B.; Campos, M. F. M.; Dutra, D. A.; da Silva Campos, V. C. & Pereira, G. A. S. (2007). Hybrid mobile robot navigational strategy for efficient data collection in sparsely deployed sensor networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 2833−2838.

Spaan, M. T. J.; Gonçalves, N. & Sequeira, J. (2010). Multirobot coordination by auctioning pomdps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1446−1451, Anchorage, USA.

Thomas, G. & Williams, A. B. (2009). Sequential auctions for heterogeneous task allocation in multiagent routing domains. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1995−2000, San Antonio, USA.

Thrun, S. & Bücken, A. (1996). Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 944−950, Portland, USA.

Viguria, A. & Howard, A. M. (2010). Probabilistic analysis of market-based algorithms for initial robotic formation. *The International Journal of Robotics Research*, 29:1154−1172.

Wagner, I. A.; Lindenbaum, M. & Bruckstein, A. M. (1998). Robotic exploration, brownian motion and electrical resistance. In *Proceedings of the 2nd International*

Workshop on Randomization and Approximation Techniques in Computer Science, pp. 116−130. London, UK.

Wurm, K. M.; S., C. & Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1160−1165, Nice, France.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 146−151, Monterey, CA.

Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pp. 47−53, Minneapolis, USA.

Zhang, K.; Collins, E. G. & Barbu, A. (2010). A novel stochastic clustering auction for task allocation in multi-robot teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3300−3307, Taipei, Taiwan.

Zheng, X. & Koenig, S. (2009a). K-swaps: Cooperative negotiation for solving task-allocation problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 373−379, Pasadena, USA.

Zheng, X. & Koenig, S. (2009b). Negotiation with reaction functions for solving complex task allocation problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4811−4816, St. Louis, USA.

Zheng, X.; Koenig, S. & Tovey, C. (2006). Improving sequential single-item auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2238−2244, Beijing, China.

Zlot, R. & Stentz, A. (2005). Complex task allocation for multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1515−1522, Barcelona, Spain.

Zlot, R.; Stentz, A.; Dias, M. B. & Thayer, S. (2002). Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3016−3023, Washington, USA.