Universidade Federal de Juiz de Fora

Faculdade de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Cláudio Maia Alves José

Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes

Juiz de Fora

2015

Cláudio Maia Alves José

Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas Eletrônicos, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientador: André Luís Marques Marcato, D.Sc.

Juiz de Fora

2015

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

M.A.José, Cláudio.
Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes / Cláudio Maia Alves José. – 2015.
97 f. : il.

Orientador: André Luís Marques Marcato, D.Sc.
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, 2015.

1. Múltiplos robôs. 2. Robô de serviço. 3. Robô de recarga. 4. Otimização I. L.M.Marcato, André. II. Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes.

Cláudio Maia Alves José

Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas Eletrônicos, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Aprovada em 20 de Fevereiro de 2015

BANCA EXAMINADORA

Professor André Luís Marques Marcato, D.Sc.
Orientador
Universidade Federal de Juiz de Fora

Professor Tiago Pereira do Nascimento, D.Sc.
Universidade Federal da Paraíba

Professor Ivo Chaves da Silva Jr, D.Sc.
Universidade Federal de Juiz de Fora

Professor Leonardo Rocha Olivi, D.Sc.
Universidade Federal de Juiz de Fora

Dedico este trabalho à minha família.

AGRADECIMENTOS

Meus agradecimentos a Deus, por ter me dado saúde e disposição no desenvolvimento deste trabalho. Agradeço também a minha amada família pelo carinho, pelo apoio e por jamais deixar faltar amor em minha vida.

Meus sinceros agradecimentos ao meu orientador, professor André Luís Marques Marcato, pelo excelente convívio e pelo grande profissionalismo sempre.

Agradeço a todos os professores que fizeram parte da minha vida.

Muito obrigado a todos os amigos do GRIn (*Grupo de Robótica Inteligente*), por se mostrarem tão companheiros sempre. Um agradecimento especial ao meu amigo Alexandre Menezes, por todo o companheirismo ao longo deste caminho.

Aos amigos e companheiros de trabalho (*Cefet-RJ*, UnED Petrópolis) pela compreensão, pela amizade fora do comum e por todo incentivo em cada momento desta maravilhosa jornada.

Obrigado a todos os que, de alguma forma, participaram e torceram pelo meu sucesso nesta caminhada.

"Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível"

(Charles Chaplin)

RESUMO

Múltiplos robôs autônomos executam tarefas que necessitam de uma cooperação entre os mesmos. Este trabalho trata um problema de coordenação de robôs móveis terrestres, os quais estão divididos em dois grupos: os robôs de serviço e os robôs de recarga. Os múltiplos robôs de serviço devem percorrer caminhos fechados que se interceptam em determinados pontos. Um controlador de alto nível atua diretamente nas velocidades médias destes veículos, evitando possíveis colisões e garantindo a segurança em suas tarefas. Os caminhos são percorridos em ciclos, os quais devem ser comensuráveis e de caráter persistente, ou seja, executados num horizonte de planejamento "infinito", o que ultrapassa as limitações de cargas de suas baterias. Para isso, é introduzido um grupo de robôs dedicados que atua no processo de troca das baterias, tarefa esta designada aos robôs de recarga. A estratégia utilizada para que todos os robôs de serviço sejam recarregados é baseada em grafo. A coordenação dos múltiplos robôs (serviço e recarga) é resolvida por meio de dois otimizadores, ambos implementados com o *solver* LINGO, integrados ao ROS (*Robot Operating System*) utilizando a linguagem C++. Um dos otimizadores coordena o movimento dos robôs de serviço com o objetivo de evitar colisões entre os mesmos. Os resultados gerados nesta primeira etapa de otimização, são utilizados para que os dois grupos robóticos estejam em sintonia durante o processo de recarga. Os caminhos percorridos pelos robôs de serviço são constituídos de pontos nos quais podem ocorrer o contato destes com os robôs de recarga. Desta maneira, a segunda parte da otimização consiste em determinar os caminhos ótimos a serem traçados pelos robôs de recarga. Este problema é resolvido por meio de um programa linear inteiro misto (*MILP*), o qual tem por objetivo minimizar o tempo global para a tarefa de recarga.

Palavras-chave: Múltiplos robôs, Robô de serviço, Robô de recarga, Otimização.

ABSTRACT

Autonomous multi-robots can be used to perform tasks that require cooperation between them. This work consists at the coordination problem of land mobile robots, divided into two groups: working robots and recharge robots. The working multi-robots must travel by closed paths that intersect each other. A high-level controller acts directly on average speeds of these vehicles, avoiding possible collisions and ensuring security in their tasks. The paths are traversed in cycles, which should be commensurate and persistents, executed on a infinite planning horizon, which overcomes the limitations of charges your batteries. For this is added one dedicated group of robots that operates in the battery exchange process. This group is called recharge robots. The strategy used to recharge all working robots is based on graph. The coordination of multiples robots is resolved through two optimizers, both implemented with LINGO solver, integrated with ROS (*Robot Operating System*) using the C++ language. One of optmizers coordinates the movement of working robots in order to avoid collisions between them. The results obtained in this first optimization step are used for the two robotic groups, became they must remain in line during the charging process. Thus, the second part of the optimization is to determine the optimum paths to be traced by the recharge robots. This problem is solved by a mixed integer linear program (*MILP*), with the objective of minimizing the overall time for recharging task.

Key-words: Multi-Robot, Working robot, Recharge robot, Optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IFR	Federação Internacional de Robótica (International Federation of Robotics)
ISO	Organização Internacional para Padronização (International Organization for Standardization)
VANT	Veículo Aéreo Não Tripulado
AUV	Veículo Subaquático Autônomo (Autonomous Underwater Vehicle)
ROS	Robot Operating System
AIS	Sistema Imunitário Artificial (Artificial Immune Systems)
IAIN	Idiotopic Artificial Immune Network
RS	Robô de Serviço
MILP	Mixed Integer Linear Program
PC	Ponto de Colisão
PR	Ponto de Recarga
RR	Robô de Recarga
TPS	Tempos de Percurso Subcaminhos
PI	Ponto Incial
FOB	Função Objetivo
DAG	Grafo Acíclico Dirigido (Directed Acyclic Graph)
GRIn	Grupo de Robótica Inteligente
ICR	Centro de Rotação Instantânea (Instantaneous Centre of Rotation)
SAIL	Stanford Artificial Intelligence Laboratory
BSD	Berkeley Software Distribution
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

DLL	Dynamic Link Library
GPL	General Public License
OSFR	Open Source Robotics Foundation

LISTA DE SÍMBOLOS

$\forall$	Para todo
$\in$	Pertence
$\leq$	Menor ou igual
θ	Teta
γ	Gama
ω	Ômega
Δ	Delta maiúsculo
$\geq$	Maior ou igual
ξ	Xi
λ	Lâmbda
τ	Tau
$\neq$	Diferente
δ	Delta minúsculo

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O mundo moderno está cada vez mais associado ao desenvolvimento e aplicações de novas tecnologias. Por exemplo, a grande quantidade de produção e a precisão exigida nas indústrias as tornam fortemente dependentes de uma automatização em seus processos construtivos. A robótica desempenha um importantíssimo papel nesta e em diversas outras questões fundamentais ao desenvolvimento do mundo, em aspectos tecnológicos e até mesmo humanos.

O termo robô é derivado da palavra Tcheca *robota*, cujo significado é "trabalho forçado", e pode ser definido de diversas maneiras. Segundo a *Federação Internacional de Robótica* (IFR), um robô é definido como: "Uma máquina que pode ser programada para executar tarefas que envolvam ações de manipulação, e em alguns casos ações de locomoção, sob um controle automático". Para a *Associação Australiana de Robótica*, uma máquina é considerada um robô quando possui alguma forma de mobilidade e pode ser programada para executar várias tarefas, realizando-as automaticamente após sua programação [1].

De acordo com a *Organização Internacional de Padronização* (ISO), os robôs devem possuir pelo menos três eixos móveis programáveis, sendo definidos apenas para ambientes industriais. A *ISO 8373: 1994*, por exemplo, define robô como: "Um manipulador multifuncional, controlado automaticamente, reprogramável, com três ou mais eixos, que pode estar fixo em determinado local ou ser móvel, utilizado em aplicações de automação industrial" [2].

Com o advento dos estudos a respeito dos robôs, a palavra robótica passou a ser altamente utilizada. Este termo foi designado por Isaac Asimov, um escritor e cientista russo, que em uma de suas obras, a partir de 1939, estabelece as seguintes leis para a robótica [3]:

Primeira Lei – "Um robô não pode ferir um ser humano ou, permanecendo passivo, deixar um ser humano exposto ao perigo".

Segunda Lei – "O robô deve obedecer às ordens dadas pelos seres humanos, exceto se tais ordens estiverem em contradição com a primeira lei".

Terceira Lei – "Um robô deve proteger sua existência na medida em que essa proteção não estiver em contradição com a primeira e segunda leis".

A robótica integra diversas áreas do conhecimento, como as engenharias elétrica, eletrônica, mecânica e de computação, além de levar em consideração o comportamento de seres humanos, de animais e de insetos, utilizando-os como auxílio no desenvolvimento

de algoritmos, principalmente no campo da inteligência artificial.

Todas as definições de robô citadas acima possuem em comum o fato de as máquinas serem programáveis e poderem realizar tarefas com uma certa autonomia após uma sequência de comandos. As definições são bastante concisas também na questão que envolve a manipulação de objetos diversos, sendo que, neste âmbito, os robôs são tratados como manipuladores. Tais robôs começaram a ser utilizados há aproximadamente cinquenta anos e são caracterizados por possuírem seu espaço de trabalho limitado, não podendo se deslocarem no ambiente ao qual estão inseridos, já que estas bases robóticas são fixas.

Estas máquinas são amplamente adotadas nas indústrias, em setores de montagem, manuseio e embalagem dos mais diversos tipos de produtos. O crescente avanço tecnológico da área permite a sua utilização em tarefas cuja alta precisão é inerente a um resultado satisfatório, como, por exemplo, na montagem e confecção de placas de circuito impresso, onde existe a necessidade de se manusear pequenos componentes eletrônicos bem como soldá-los de forma precisa. Na Figura 1, pode-se observar robôs pertencentes a esta classe.

Fonte: [4] e [5]

Figura 1 – Robôs Manipuladores

Um outro grupo de robôs que também tem crescido bastante e que se enquadra neste trabalho, são os robôs móveis, que, como o próprio nome sugere, são plataformas robóticas que podem se mover nos ambientes de aplicação. Enquadram-se, nesta categoria, robôs que se movem sobre a terra, em ambientes subaquáticos e também através do ar, conforme é visto na Figura 2.

Estes veículos são dotados de sensores que os auxiliam quanto à percepção do mundo real, permitindo-os tomarem as suas próprias decisões previamente programadas pelo homem. Devido a esta característica, estes veículos podem ser chamados de robôs autônomos, podendo movimentar-se em ambientes completamente ou parcialmente conhecidos, bem como em locais totalmente desconhecidos, utilizando-se de técnicas para as construções de seus próprios mapas e com isso se orientando adequadamente.

As aplicações com robôs autônomos têm crescido bastante nas mais diversas áreas de pesquisa. O fato de estes robôs atuarem em ambientes reais, contornando problemas

Fonte: [6], [7] e [8]

Figura 2 – Robôs Móveis

ocasionados pela dinâmica destes locais como: variações de temperatura, variações de luminosidade e obstáculos estáticos ou dinâmicos, garante o seu emprego nas indústrias, onde podem ser utilizados no transporte de diferentes tipos de materiais, e ainda executar tarefas em setores que oferecem riscos à integridade física do homem.

Veículos aéreos não tripulados (VANTs) são muito empregados no monitoramento de regiões, detectando focos de incêndio, derramamento de óleo e contribuindo para a segurança de determinadas localidades [9]. Pode-se utilizar veículos subaquáticos autônomos (AUVs) quando se deseja observar rios e oceanos em aplicações como controle arqueológico, biologia marinha, exploração de navios naufragados e supervisão de barragens.

Entre os grandes desafios encontrados no uso dos veículos autônomos terrestres, aéreos e subaquáticos, estão o planejamento de suas trajetórias no ambiente em questão [10], a tecnologia construtiva, o uso de sensores adequados a cada situação e o desenvolvimento de controladores para garantir o sucesso das operações.

Uma questão de fundamental importância em aplicações robóticas está ligada ao uso de robôs autônomos atuando de forma cooperativa. Todas as tarefas de monitoramento e exploração de ambientes, anteriormente citadas, são realizadas de forma mais rápida e abrangente quando desempenhadas cooperativamente por múltiplos robôs.

A autonomia de energia destes robôs é muito baixa, principalmente em se tratando de VANTs. Em função disso, vários trabalhos vêm sendo desenvolvidos objetivando um "aumento" desta autonomia por meio de reabastecimento ou troca de baterias utilizando outros robôs atuando de forma cooperativa [11], [12] e [13].

Este trabalho de pesquisa trata do problema de cooperação entre dois grupos de robôs autônomos terrestres. O objetivo é fazer com que alguns robôs, denominados robôs de serviço, percorram seus caminhos em movimentos coordenados para que não haja colisões entre os mesmos, num horizonte de planejamento que ultrapassa as limitações de cargas das suas baterias. Desta forma, é introduzido um grupo de robôs dedicados, denominados robôs de recarga, fazendo com que as tarefas executadas pelos robôs de serviço sejam consideradas persistentes.

1.2 MOTIVAÇÃO

A coordenação de múltiplos robôs autônomos tem interessado grandemente a comunidade científica e as indústrias [14]. Dar o máximo de autonomia a grupos de robôs que atuam em conjunto, significa realizar tarefas e atingir metas com mais qualidade, com menos gastos de tempo e de energia. Devido ao avanço das pesquisas na área, atualmente é possível utilizar concomitantemente diferentes grupos de robôs móveis. Pode-se, por exemplo, fazer uso de veículos terrestres com VANTs em operações conjuntas, otimizando o alcance dos objetivos [15].

O presente trabalho aborda dois tipos de coordenação. Trata-se de dois grupos distintos de veículos autônomos terrestres que atuam de forma cooperativa. Os robôs de serviço devem ter suas velocidades controladas de forma a nunca estarem ao mesmo tempo em zonas de possíveis colisões com outros veículos pertencentes ao mesmo grupo, executando, desta forma, ciclos em seus caminhos, os quais são conhecidos previamente.

A ideia é que estes caminhos sejam percorridos em tempos prolongados mesmo que a bateria de cada robô de serviço se esgote, como em uma aplicação de vigilância contínua ou na exploração de um determinado local [16]. Neste sentido, verifica-se uma importante contribuição visto que muitos dos trabalhos que fazem uso de veículos autônomos nas chamadas tarefas persistentes, ou de longo prazo de duração, não tratam o problema conjuntamente [17] e [9].

A realização das trocas de baterias dos robôs de serviço exige uma boa coordenação entre estes e o grupo responsável pela recarga. O encontro entre um robô de recarga e seu respectivo robô de serviço, deve ser realizado em posições e momentos específicos. Esta sincronização e coordenação global dos múltiplos robôs constituem o ponto focal do trabalho, pois permitem que as tarefas dos robôs de serviço sejam realizadas em um longo prazo.

A técnica empregada foi proposta em [11] e utiliza como estratégia para o encontro dos grupos de robôs uma abordagem baseada em grafo. Este método se aplica muito bem ao problema a ser discutido no Capítulo 3, já que o otimizador utilizado na coordenação dos robôs de serviço conforme mostrado em [18] fornece, além dos perfis de velocidade

desses veículos, os instantes de tempos exatos em que cada um desses robôs passam por pontos discretos, escolhidos de acordo com a necessidade da aplicação, ao longo de seus caminhos.

1.3 OBJETIVOS

O principal objetivo do trabalho é estabelecer uma estratégia para que todos os veículos autônomos inseridos no grupo de serviço, sejam recarregados ao longo da execução de suas tarefas.

Seja qual for a aplicação, é muito importante trabalhar a questão da autonomia de energia dos veículos. Com isso, torna-se clara a relevância do desenvolvimento de trabalhos que utilizam técnicas para a coordenação de robôs. O conceito de cooperação deve receber uma atenção especial e é incorporado ao objetivo central do presente trabalho.

Como um objetivo secundário, porém não menos importante, destaca-se a utilização do ROS (*Robot Operating System*) na implementação prática. Fazer a integração do *solver* de otimização LINGO com o ROS utilizando a linguagem de programação C++, também faz parte dos objetivos.

1.4 REVISÃO BIBLIOGRÁFICA

A cooperação entre grupos de robôs autônomos tem inspirado o desenvolvimento de diversos trabalhos, cujo eixo temático principal envolve a questão do gerenciamento da energia despendida. O objetivo primordial é fazer com que as funções designadas a um determinado grupo de robôs sejam realizadas em um horizonte de longo prazo.

Como mencionado anteriormente, a autonomia de energia destes veículos é, geralmente, baixa. As baterias de LiPo (*Lithium Polymer*) atualmente disponíveis, por exemplo, garantem uma autonomia de apenas alguns minutos de vôo para um típico VANT. Em aplicações de monitoramento, exploração de ambientes ou em qualquer tarefa cuja continuidade é fundamental, isto é um fator determinante.

O trabalho [19] aborda um problema de pesquisa e cobertura de um ambiente desconhecido. São utilizados veículos para a tarefa de busca e veículos para a tarefa de cobertura. Os robôs responsáveis pela busca de alvos no ambiente atualizam seus mapas baseado nas medições de seus sensores. O objetivo destas máquinas é otimizar a quantidade de informações coletadas pela equipe. Os veículos de serviço se espalham de forma otimizada pelo ambiente para realizar a cobertura. O sucesso da missão de cobertura da região é garantido pelo princípio de invariância de LaSalle. Os veículos de serviço têm as suas configurações modificadas pelas atualizações dos mapas fornecidos pelos robôs de busca.

Monitorar de forma contínua um determinado ambiente, implica em minimizar o tempo entre as visitas dos veículos autônomos a uma mesma região de interesse. Com este intuito, Nigam & Kroo (2008) [17] realizam um estudo no qual são utilizados múltiplos veículos aéreos. O uso de veículos aéreos não tripulados para desempenhar esta função é justificado pelas suas confiabilidade e eficiência, uma das razões para o aumento das pesquisas em controle e inteligência artificial. A metodologia empregada por [17] consiste em estender uma política de controle semi-heurístico de um único veículo para múltiplos VANTs por meio de dois métodos.

Em um dos casos é abordada uma política de controle reativo no uso de um único VANT, obtendo um resultado interessante quando o número de veículos torna-se grande. É desenvolvido, primeiramente, um estudo considerando um único veículo e apenas duas células de visita. O objetivo é fazer com que o robô escolha qual das células visitar primeiro, para depois realizar os deslocamentos necessários à continuidade da operação. Cada célula possui uma idade, significando o tempo decorrido desde a última observação, e está situada a uma determinada distância do veículo de monitoramento. A combinação linear entre estes parâmetros auxilia na tomada de decisões para o VANT, sendo que o objetivo é minimizar a idade máxima de cada célula. Para que o ambiente seja monitorado por múltiplos veículos, estende-se esta metodologia, sendo a região subdividida em células que são visitadas pelos VANTs de acordo com a política de controle descrita.

A outra metodologia consiste em particionar de maneira ótima o ambiente a ser monitorado. Cada subespaço é alocado, também de forma ótima, a um dos VANTs utilizados e, para esta finalidade, é utilizada uma heurística baseada em procedimentos de leilões. O objetivo é minimizar a idade máxima de cada subespaço, com vários veículos atuando em monitoramento paralelo. Mesmo que a função dos veículos seja o monitoramento persistente, não há um planejamento quanto à questão da energia em nenhum dos métodos utilizados.

Em uma outra aplicação de monitoramento, Suji et al. (2007) [9] utiliza uma equipe de VANTs para supervisionar incêndios florestais em uma determinada região. As atribuições aos robôs de serviço são realizadas também através de procedimentos baseados em leilões. Um controlador foi desenvolvido para que os agentes de serviço fiquem igualmente espaçados pelo ambiente a ser monitorado. O trabalho também não aborda a questão da recarga dos robôs.

Como estratégia e possível solução para este típico problema de energia, pode-se citar o trabalho [12]. Fuji et al. (2013) desenvolve uma plataforma autônoma capaz de realizar a troca das baterias de VANTs, permitindo que uma tarefa de monitoramento seja realizada continuamente. Como principais contribuições do trabalho estão os resultados de testes experimentais e a descrição, com desenhos esquemáticos, da plataforma de recarga desenvolvida. Como resultados relata-se noventa por cento de eficiência durante o pouso

dos veículos e sucesso absoluto no processo autônomo de recarga.

Em uma mesma linha de pesquisa, Swieringa et al. (2010) [13] implementa uma plataforma para a recarga de helicópteros de pequeno porte, deixando como sugestão para trabalhos futuros o desenvolvimento de um mecanismo que transporte a plataforma até o veículo aéreo, com o objetivo de aumentar a durabilidade de sua operação.

Uma vantagem da utilização de veículos autônomos aéreos nestas missões é o fato de se desprezar os obstáculos presentes no solo. Entre as desvantagens estão a limitação da abrangência dos sensores com a altitude de vôo dos veículos e as questões referentes à dinâmica destes robôs.

Uma questão muito importante quando se aplicam robôs em tarefas cooperativas é o tratamento de possíveis colisões entre os mesmos. Os trabalhos [20], [21], [22] e [23] abordam as missões dos robôs sob este aspecto.

Em [20], os robôs fazem uso de seus próprios sensores para evitarem choques com obstáculos, ou seja, não há um planejamento centralizado para a comunicação entre os robôs. Os veículos podem parar ou até mesmo desviar de seus trajetos originais para que não ocorra uma colisão. No trabalho [21], os robôs devem percorrer caminhos fechados em ambiente susceptíveis à variações. Como no trabalho [20], os veículos podem parar a fim de que as colisões sejam evitadas.

No trabalho [22], a coordenação dos múltiplos robôs, livre de colisões, é realizada através da alteração dos tempos de início das trajetórias de cada um dos robôs envolvidos na missão. Em [21], é resolvido um problema de otimização para a garantia de segurança na missão dos robôs. São analisadas restrições de aceleração e de velocidade para os veículos. O trabalho [18], cuja técnica é utilizada no desenvolvimento do presente trabalho, aborda a coordenação de múltiplos veículos aéreos não tripulados, de asa fixa. Os robôs devem percorrer caminhos fechados, que se interceptam. Estes caminhos são conhecidos e os veículos não podem interromper seus movimentos e nem desviarem de suas rotas para evitar colisões. É resolvido um programa linear inteiro misto, a fim de se controlar as velocidades médias dos robôs. Além disso, os veículos devem percorrer os seus caminhos de forma periódica.

Os robôs móveis terrestres são muito utilizados não só na exploração e no monitoramento de ambientes, mas também em tarefas que envolvem a manipulação de objetos em ambientes perigosos para os seres humanos. Nesta linha de raciocínio, Hossam Meshref (2013) [24] traz à tona o conceito de cooperação entre robôs autônomos, treinando-os para que sejam utilizados em situações reais, em ambientes nocivos aos seres humanos. Como um possível ambiente de aplicação, Hossam Meshref cita o local do acidente com um reator nuclear na cidade de Fukushima, no Japão, em 11 de março de 2011.

A metodologia de controle cooperativo adotada no trabalho [24] é baseada em

estudos biológicos, mais precisamente nos Sistemas Imunitários Artificiais (AIS), *Artificial Immune Systems*, no qual é utilizada uma rede IAIN, *Idiotopic Artificial Immune Network*. A técnica AIS é promissora no estudo de aprendizagem de robôs e impulsionará novas pesquisas na área. Como trabalhos futuros, Hossam Meshref deixa sugestões em relação a adição de obstáculos no espaço de trabalho dos veículos, e a realização de vários testes para se verificar mais profundamente o desempenho deste tipo de aprendizagem, além de relatar bons resultado obtidos no trabalho.

No campo da exploração de ambientes desconhecidos, Burgard et al. (2000) [25] utiliza múltiplos robôs autônomos de forma cooperativa com o objetivo de minimizar o tempo global da exploração. Para isto, utilizam-se vários pontos de destino espalhados pelo ambiente, de forma que os robôs são levados a cobrirem diferentes áreas, de acordo com o alcance dos seus sensores. A cada ponto de destino é associado um custo e, à medida que um veículo se dirige a um ponto, esta localidade é excluída das tarefas dos outros robôs envolvidos na operação. O time de robôs deve escolher diferentes pontos durante a exploração e a modelagem adotada é probabilística.

Para monitorar ou fazer a varredura de determinados ambientes sujeitos à mudanças, Smith et al. (2011) [26] utiliza múltiplos robôs autônomos com velocidades individualmente controladas. O ambiente é modelado como um campo com localizações finitas e seu crescimento ou decrescimento se dá de forma linear, estando diretamente associado às regiões que estão dentro ou fora do alcance dos robôs, que se movem em caminhos fechados. O controle das velocidades de cada robô evita que haja um crescimento ilimitado destes campos.

Embora os trabalhos a respeito dos robôs terrestres, citados anteriormente, visem aplicações com uma maior durabilidade de tempo, não há uma abordagem em relação à recarga destes veículos durante as operações.

Derenick et al. (2011) [27] também retrata a questão do monitoramento de ambientes, mapeados por diagramas de Voronoi baseados em centro de gravidade, por múltiplos robôs de forma persistente, porém com estratégias para recarregá-los. O ambiente de trabalho é dotado de bases fixas, às quais os robôs se dirigem nos casos em que suas baterias estão em um nível considerado crítico para a continuidade da operação. Quando um robô se dirige à sua estação de recarga, os outros robôs se ajustam de forma cooperativa para que a região continue sendo monitorada. A desvantagem deste sistema é que se todos os robôs atingirem seu nível crítico de energia, ao mesmo tempo, o ambiente ficará sem monitoramento.

Litus et al. (2009) [28] utiliza uma estratégia diferente para a recarga dos robôs que estão realizando as tarefas. É designado apenas um robô para auxiliar nesta missão, com os outros se organizando de forma cooperativa para serem recarregados em sequência. São propostas heurísticas para melhorar o desempenho na operação de recarga, visto que,

sendo utilizado apenas um robô de recarga, o problema se assemelha ao famoso caso do caixeiro viajante, estudado no desenvolvimento de técnicas de otimização.

O trabalho [11] propõe uma estratégia de recarga baseada em grafo. A técnica utilizada permite o uso de múltiplos robôs para auxiliar na recarga de um grupo de robôs de serviço. Os veículos responsáveis pela tarefa de recarga possuem velocidade constante e se encontram com os robôs de serviço em pontos destinados à operação de recarga. São realizadas simulações para um caso com vários robôs de recarga e para um caso com apenas um robô de recarga. Neste último caso, é feita uma comparação entre a solução por um programa linear inteiro misto e por um procedimento heurístico. Além disso, Mathew et al. (2013) demonstra que o problema é caracterizado como *NP-hard*.

1.5 ORGANIZAÇÃO DO TRABALHO

Após uma contextualização do tema abordado e algumas discussões sobre as pesquisas desenvolvidas na área, o restante do trabalho é organizado da forma descrita a seguir.

No Capítulo 2 são abordados os fundamentos teóricos necessários à compreensão do trabalho como um todo. São discutidos, de forma sucinta, conceitos sobre grafos, otimização, robôs móveis terrestres, visão computacional e ROS. As discussões são realizadas buscando sempre fazer uma ligação entre a teoria e o problema geral tratado neste trabalho.

Em algumas das seções são resolvidos exemplos para uma melhor compreensão da ferramenta utilizada. Na seção de otimização, é dada uma atenção especial ao *solver* LINGO utilizado no tratamento dos problemas de otimização. A seção ROS também recebe um maior detalhamento, por ser a ferramenta que integra todas as técnicas utilizadas.

O Capítulo 3 é dedicado aos robôs de serviço. São discutidos a formação dos caminhos percorridos pelos veículos e o controle de suas velocidades médias. A modelagem matemática deste problema de otimização é descrita, explicando-se todas as restrições utilizadas.

A integração de todo o problema considerado no trabalho é realizada no Capítulo 4. Primeiramente, é apresentado o grupo de robôs responsável pela tarefa de recarga, ou seja, que permite o planejamento de realização das tarefas a longo prazo, pelos veículos de serviço.

Na seção de metodologia, é apresentada toda a formulação matemática que envolve o otimizador das tarefas de recarga. O tratamento do problema *MILP* fica mais claro com a resolução de um caso exemplo, no qual todas as equações de restrições e função objetivo são desenvolvidas. O algoritmo de otimização, nesta etapa do problema, é explicado em uma seção separada. O seu detalhamento é feito por meio de um fluxograma e da descrição,

passo a passo, do código desenvolvido no *software* LINGO (11.0).

O Capítulo 5 é reservado aos resultados obtidos nas implementações, juntamente com as suas discussões. São considerados três casos, visando sempre as aplicações práticas em cenários menores. As duas primeiras situações consideram apenas os resultados dos otimizadores. São informados, desta forma, os mapas que devem ser utilizados pelos robôs de recarga, sem uma preocupação com erros de localização, por exemplo. O último caso leva em consideração o tratamento dos robôs, como, por exemplo, os controles utilizados, além da utilização do ROS para gerenciar toda a simulação. Embora seja mais completo que os casos anteriores, ainda não é uma aplicação totalmente prática.

As conclusões finais e considerações sobre trabalhos futuros encontram-se no Capítulo 6. No Apêndice A, é apresentado um exemplo didático sobre como se modelar e resolver um pequeno problema de otimização em LINGO.

Além disso, todos os capítulos contêm uma seção especial para conclusões, com exceção do Capítulo 5, já que os resultados são devidamente comentados e discutidos em sua apresentação. O objetivo é que os temas abordados fiquem mais claros e autoexplicativos.

1.6 PUBLICAÇÃO DECORRENTE

- TEIXEIRA, A. M ; JOSÉ, C. M. A. et al. Coordenação Ótima de Múltiplos Robôs de Serviço e de Recarga em Tarefas Persistentes. In: IEEE. *International Conference on Industry Applications Induscon, 2014 IEEE International Conference on.* [S.l.], 2014. p. 849–854.

1.7 CONCLUSÕES

Utilizar robôs em tarefas diversas é uma alternativa inteligente e de suma importância, face à necessidade de desenvolvimento e aperfeiçoamento das civilizações. Para que os objetivos gerais das missões designadas a estas máquinas sejam otimizados, é importante que os veículos tenham o máximo de autonomia durante a execução das mesmas. Neste caso, desenvolver e aplicar técnicas objetivando a utilização ótima de múltiplos robôs atuando de maneira cooperativa é fundamental. Na seção que aborda os objetivos do trabalho, foram citadas várias destas técnicas, que serão descritas com os devidos detalhes nos capítulos seguintes.

A utilização dos robôs autônomos com esta finalidade pode garantir que as tarefas sejam executadas persistentemente. Isto é uma alternativa vital, visto que a autonomia de energia destes veículos é consideravelmente baixa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo abordar, de maneira geral, os aspectos teóricos necessários para uma melhor compreensão do trabalho. As ferramentas são descritas de forma isolada e, sempre que possível, é feita uma breve conexão destas com o trabalho.

2.1 TEORIA BÁSICA SOBRE GRAFOS

De uma forma geral, um grafo é formado por uma tripla ordenada $G(V, A, c)$, na qual V é um conjunto não vazio, denominado de conjunto de vértices do grafo, A é um conjunto de pares não ordenados de elementos, não necessariamente distintos, de V. A ligação entre dois vértices recebe o nome de aresta e c pode ser utilizado para explicitar o grau de relacionamento entre estes vértices [29]. A Figura 3 mostra a formação de um grafo simples.

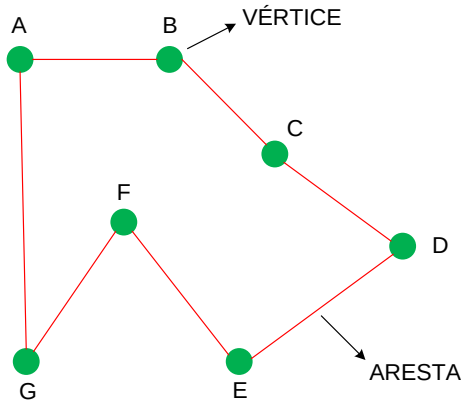
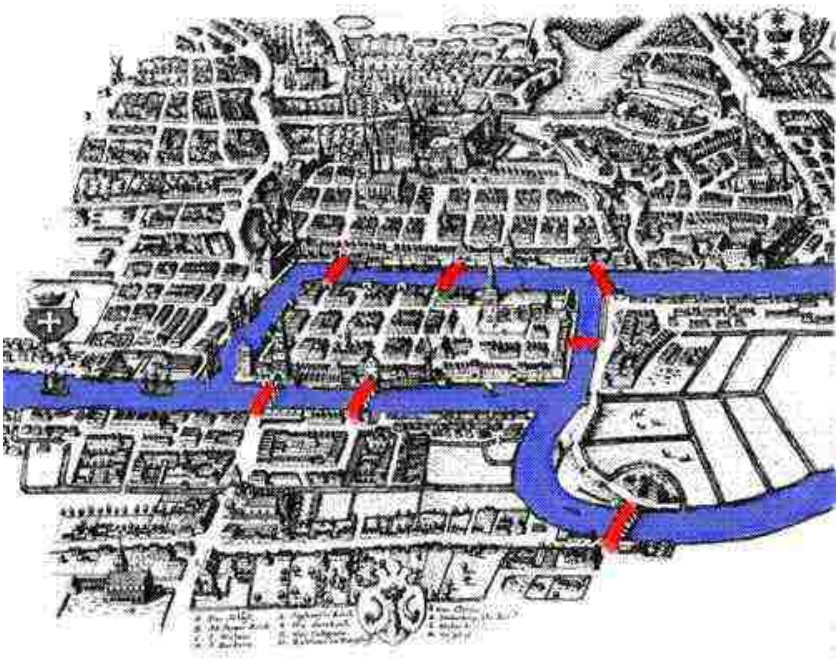


Figura 3 – Grafo Simples

A origem da teoria de grafos se deu no século XVIII, mais precisamente no ano de 1735, com o *Problema das Pontes de Königsberg* [30].

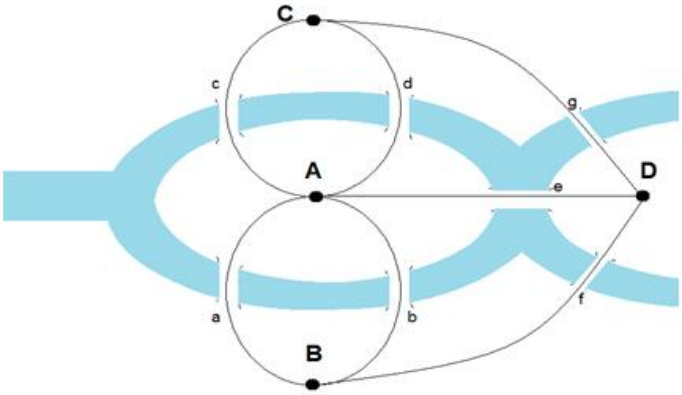
A cidade de *Königsberg*, atual Caliningrado (região da Prússia), é formada por um complexo de ilhas ligadas entre si por sete pontes. A Figura 4 ilustra a região descrita. Naquela época, os habitantes da cidade discutiam se era possível a realização de um percurso que atravessasse toda a cidade, passando por todas as pontes uma única vez.

Este problema foi solucionado por Leonhard Euler através da criação do primeiro grafo da história. O grafo representativo pode ser visto na Figura 5, e a conclusão é de que não é possível a obtenção de um circuito Euleriano, como ficaram conhecidos os problemas desta natureza, para este grafo, já que o mesmo possui todos os vértices ligados a um número ímpar de arestas.



Fonte: [31]

Figura 4 – Cidade de Königsberg



Fonte: [32]

Figura 5 – Grafo: *Problema das Pontes de Königsberg*

Uma vantagem de se utilizar grafo é o grande poder de abstração que esta ferramenta proporciona. Diversos problemas podem ser solucionados utilizando, por exemplo, um mesmo algoritmo. Dado um determinado problema, é feita uma modelagem baseada em grafo e aplicado um algoritmo que se enquadre na situação analisada. As aplicações da teoria de grafos englobam áreas diversas, pode-se encontrar o emprego dessas técnicas em redes de comunicação, linhas de transmissão e distribuição de energia elétrica, controles de malhas viárias, robótica, automação, confiabilidade e outras. Exemplos de aplicações

de grafos podem ser vistos em [33], [34] e [11].

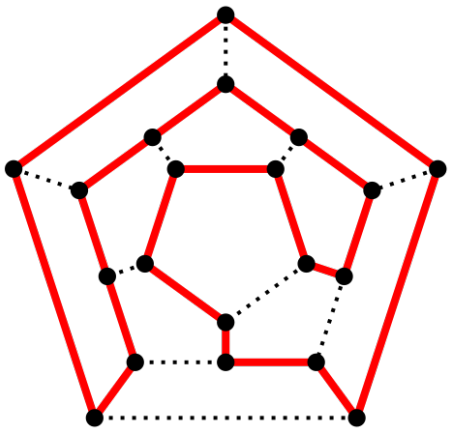
Os grafos podem ser direcionados ou não, o que implica em associar ou não uma direção às arestas. Mesmo que, por definição, o conjunto de arestas seja formado por pares ordenados de elementos não necessariamente distintos, a técnica utilizada para a coordenação dos dois grupos robóticos, neste trabalho, impõe a distinção entre estes elementos. A modelagem da parte do problema, baseada em grafo, pode ser vista na Seção 4.1, Capítulo 4.

Nas subseções seguintes serão abordados dois problemas clássicos no estudo da teoria de grafos, pela sua semelhança e importância na utilização da técnica de otimização para um dos grupos robóticos utilizados neste trabalho.

2.1.1 **Grafo Hamiltoniano**

Um grafo G é dito Hamiltoniano quando possui um caminho que contém todos os vértices de G. Este caminho deve iniciar e terminar num mesmo vértice, passando por todos os vértices de G uma única vez.

O matemático irlandês William Hamilton, em 1859, criou um jogo denominado de "*The Icosian Game*" [35]. O objetivo era passar por todos os vértices de um dodecaedro uma única vez, Figura 6. Aos vértices foram dados nomes de cidades importantes e o jogador teria de começar e terminar em uma mesma cidade.



Fonte: [36]

Figura 6 – Dodecaedro: *Caminho Hamiltoniano*

Um *ciclo Hamiltoniano* se difere de um *ciclo Euleriano*, visto no *Problema das Pontes de Königsberg*, pelo fato de que neste deve existir um caminho que englobe todas as arestas do grafo. Já no *ciclo Hamiltoniano* deve haver a passagem por todos os vértices do grafo. O problema do *caminho Hamiltoniano* está diretamente ligado ao problema do *caixeiro viajante*, visto a seguir.

2.1.2 **O Problema do Caixeiro Viajante**

Um grafo, como já mencionado, é constituído de vértices e arestas, podendo a estas ser atribuídos pesos. Por exemplo, em uma malha rodoviária que interliga algumas cidades, as arestas representam as rodovias e seus pesos são os custos das viagens.

O caso do *caixeiro viajante* é um problema de *grafo Hamiltoniano*. A questão é determinar um percurso, no qual parte-se de uma cidade e visita-se todas as outras uma única vez, retornando à cidade de origem, de modo que o custo total desta viagem seja mínimo. O trabalho [37] relata uma abordagem na resolução deste problema.

Encontrar *ciclos Hamiltonianos* é mais complexo que encontrar *ciclos Eulerianos*, principalmente quando existem pesos atribuídos às arestas, pois tem-se, agora, que encontrar dentre os caminhos Hamiltonianos, o de menor custo. Nesta linha de raciocínio existem diversos algoritmos de otimização, desde problemas lineares inteiros mistos (*MILPs*) até procedimentos heurísticos, conforme pode ser visto nos trabalho [38].

Neste trabalho, o problema de coordenação global dos robôs de serviço e de recarga se assemelha ao problema do *caixeiro viajante*, quando a tarefa de recarga é designada a um único robô. Neste contexto, o robô de recarga é o vendedor, as cidades são os pontos destinados às recargas dos robôs de serviço e os pesos são, por exemplo, as distâncias ou os comprimentos das arestas (rodovias). O objetivo é que o robô de recarga auxilie todos os robôs de serviço no processo de troca de baterias, saindo de um ponto inicial e percorrendo um caminho no qual cada robô de serviço é visitado uma única vez, retornando ao ponto de origem. Este caminho deve ser otimizado de modo que a distância total seja mínima. A metodologia é descrita no Capítulo 4 com mais detalhes.

Para casos em que se utilizem mais robôs destinados à operação de recarga, devem-se encontrar caminhos ótimos para todos estes veículos, porém, não há a necessidade de que cada robô visite todos os robôs de serviço. Portanto, o algoritmo utilizado e explicado no Capítulo 4, soluciona este problema por completo.

2.2 OTIMIZAÇÃO

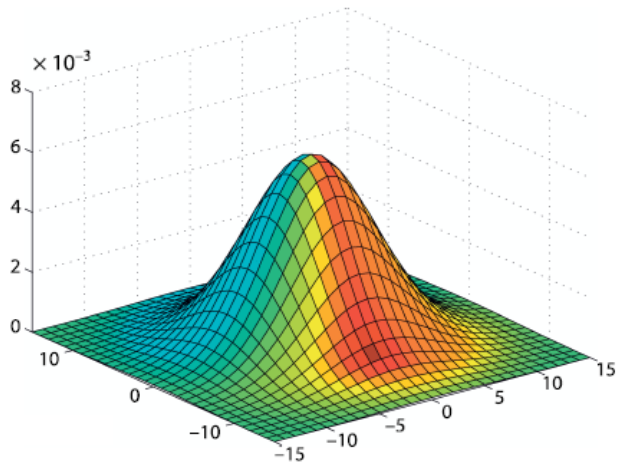
O conceito de otimização se aplica a qualquer campo de atividade. Otimizar, em linhas gerais, significa buscar um melhor desempenho, obter o maior rendimento possível de um determinado processo, dados os respectivos pontos de vista (funções de custo).

Quando se deseja melhorar um processo, seja a produtividade de uma indústria ou até mesmo questões relacionadas a tempo ou a custos, é necessário ter um objetivo bem definido. À partir da descrição e modelagem do problema, procura-se estabelecer prioridades e empregar as melhores técnicas a fim de se alcançarem os melhores rendimentos. A opção por solucionar um problema através da otimização deve ser realizada após um estudo e uma análise qualitativa, evoluindo para uma análise quantitativa a fim de se

escolher o melhor método aplicável [39].

Todo problema de otimização é composto por uma "função objetivo". O grau de desempenho do processo analisado é medido pelo valor desta função, a qual é minimizada ou maximizada dependendo do objetivo em questão. Isto justifica o grande número de áreas que fazem uso das técnicas de otimização, como a biologia, a química, a administração, as ciências físicas e as engenharias.

Seja qual for o objetivo, minimizar ou maximizar, os problemas de otimização podem fazer uso de equações, inequações ou qualquer outro artifício para restringir o conjunto de soluções. Em outras palavras, a minimização ou a maximização seguem a um conjunto de “regras”, denominadas de restrições. A Figura 7 mostra o gráfico de uma possível função objetivo, com um ponto de máximo bem definido. Tem-se, desta forma, o mapeamento da função no qual se pode definir uma região viável, ter uma estimativa inicial e avaliar a sua sensibilidade em relação às variáveis do problema.



Fonte: [40]

Figura 7 – Exemplo de uma função (*Gaussiana*) com um máximo global

Os problemas de otimização podem ser resolvidos de maneira *off-line*, quando se deseja projetar algum equipamento, ajustar modelos estáticos ou dinâmicos, sintetisar algum processo, entre outras aplicações. Pode-se também empregar técnicas de otimização em modo *on-line*, principalmente em sistemas que fazem uso de controle adaptativo e controle ótimo. No presente trabalho, os algoritmos de otimização são executados *off-line*. Porém, pode-se perfeitamente implementá-los de forma integrada na prática, fazendo com que os robôs utilizados tomem as suas decisões com os algoritmos e métodos de controle sendo executados *on-line*, o que constitui um objetivo futuro.

O *solver* de otimização utilizado nas implementações deste trabalho é descrito na subseção seguinte. São mostradas, de forma objetiva, algumas das funcionalidades do

software utilizadas nos códigos.

2.2.1 *Software* LINGO

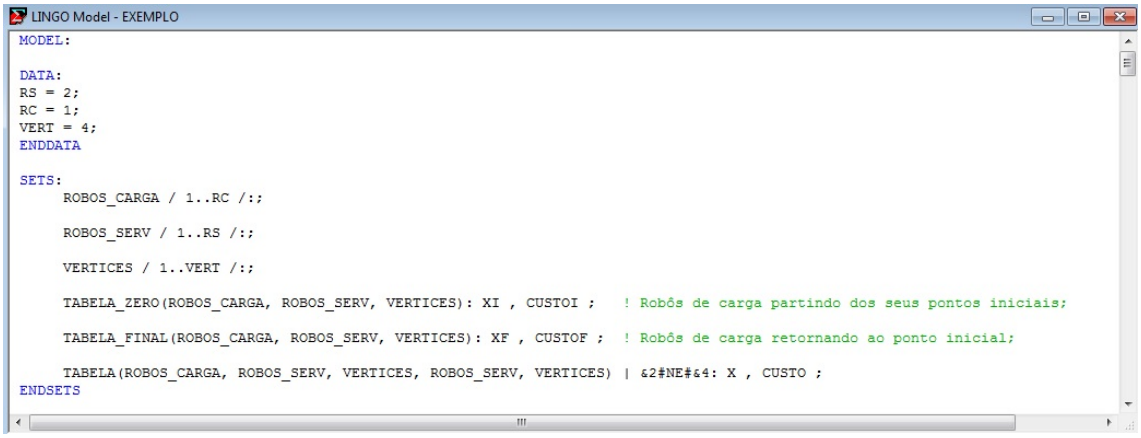
A coordenação ótima dos múltiplos robôs é realizada através da solução de dois programas de otimização. Neste trabalho, os dois otimizadores são implementados no *software* LINGO (11.0). Este *software* faz parte do projeto de um sistema chamado LINDO [41], o qual é responsável pelo desenvolvimento de alguns *softwares* destinados a resolução dos mais variados tipos de problemas de otimização.

Através do LINGO pode-se modelar problemas matemáticos de uma forma simples e eficiente. Sua linguagem é baseada em conjuntos, sendo assim, os modelos são expressos de maneira bastante compacta e podem ser resolvidos iterativamente por meio de *scripts* internos do programa. Este *software* de modelagem pode ser utilizado para solucionar e auxiliar na busca de melhores respostas para problemas lineares e não-lineares. Além disso, é indicado também em aplicações de programação inteira mista, a qual possui variáveis contínuas e inteiras no mesmo problema. O LINGO utiliza, para a resolução de problemas lineares, o método simplex revisado com o produto da inversa. Os modelos inteiros são resolvidos pelo método *branch and bound*. Em modelos lineares inteiros há um considerável pré-processamento no sentido de se restringir as regiões viáveis, o que melhora o tempo para a resolução dos modelos de programação inteira.

A modelagem de um problema em LINGO é divida em seções. A seção conjuntos (*SETS*) tem por objetivo relacionar grupos de objetos que serão utilizados na descrição da função objetivo e das restrições. Existem conjuntos primitivos e derivados. Na implementação do algoritmo que otimiza a tarefa do grupo de robôs responsável pela recarga, por exemplo, faz-se o uso destes dois tipos de conjuntos. A Figura 8 mostra um exemplo de como se usar algumas destas seções (*SETS* e *DATA*). O problema aqui utilizado como exemplo será detalhado no Capítulo 4, cabendo, neste momento, apenas uma familiarização e entendimento do *solver* utilizado.

Nota-se a presença de vários conjuntos. O conjunto denominado de *ROBOS_SERV* é primitivo, ou seja, formado apenas pelos objetos robôs de serviço, não podendo ser reduzido posteriormente. Já o conjunto *TABELA_ZERO* é formado pela associação de diferentes conjuntos, constituindo desta maneira um conjunto derivado. O número de objetos pertencentes a cada um destes conjuntos é determinado de acordo com a aplicação, especificado pelo próprio usuário.

A seção *DATA*, vista na mesma figura, é responsável pela entrada de dados. Neste caso exemplo, pode-se digitar os números de robôs de serviço (*RS*), de robôs de recarga (*RC*) e de vértices (*VERT*) que constituem o problema analisado. Portanto, na seção *SETS*, cada conjunto já está condicionado a um problema genérico. No caso da Figura 8, o cenário é constituído por dois robôs de serviço, um robô de recarga e quatro vértices.



Fonte: *print screen* do *software* de otimização LINGO

Figura 8 – *Software* LINGO: seções *DATA* e *SETS*

É importante ressaltar que, neste exemplo, cada membro dos conjuntos derivados possuem atributos que são utilizados diretamente na resolução do problema. Analisando, por exemplo, o conjunto *TABELA*, pode-se observar os atributos *X* e *CUSTO*, responsáveis pela habilitação e pela alocação de pesos a cada uma das arestas do problema. O mesmo acontece com os conjuntos derivados *TABELA_ZERO* e *TABELA_FINAL*. Na Tabela 1 são especificados os conjuntos derivados presentes na Figura 8, com os seus respectivos membros.

Tabela 1 – Conjuntos derivados e seus respectivos membros

CONJUNTOS DERIVADOS	**MEMBROS (DISPOSIÇÃO)**
TABELA	(ROBOS_CARGA,ROBOS_SERV,VERTICES,ROBOS_SERV,VERTICES)
TABELA_ZERO	(ROBOS_CARGA,ROBOS_SERV,VERTICES)
TABELA_FINAL	(ROBOS_CARGA,ROBOS_SERV,BERTICES)

O LINGO trabalha também com operadores aritméticos, operadores lógicos e com operadores relacionais. Na Figura 8, ainda na seção *SETS*, o conjunto *TABELA* possui um operador relacional do tipo &2#NE#&4. Esta operação retorna *TRUE* quando os operandos &2 e &4 são diferentes (NE: *Not Equal*). Os parâmetros &2 e &4 referem-se ao conjunto *ROBOS_SERV*, e neste caso, variam de 1 a 2.

Para escrever as equações necessárias ao problema de otimização, é preciso utilizar-se de funções do LINGO que atuam sobre os conjuntos predefinidos. Funções que utilizam somatórios, como as que são utilizadas no problema de otimização para os robôs de recarga, são implementadas iniciando-se com o comando @SUM. Este operador faz com que todo o conjunto em questão seja percorrido, retornando o somatório dos valores dos atributos associados ao mesmo. No Capítulo 4, a função objetivo do problema de otimização de recarga é apresentada (Equação 4.3). Neste caso, o operador retorna o somatório dos pesos

associados a cada uma das arestas do grafo representativo.

As restrições do problema também podem ser escritas de forma simples e direta.

No Apêndice A, é apresentado um problema exemplo, retirado do manual de instruções do LINGO (11.0), para uma melhor compreensão deste *solver*.

2.2.1.1 *Integração do Solver LINGO com C++*

Na subseção anterior foram apresentadas e exemplificadas apenas algumas das diversas funções contidas no LINGO. Este *software* possui uma extensa biblioteca de funções que facilitam a resolução dos modelos de otimização [42].

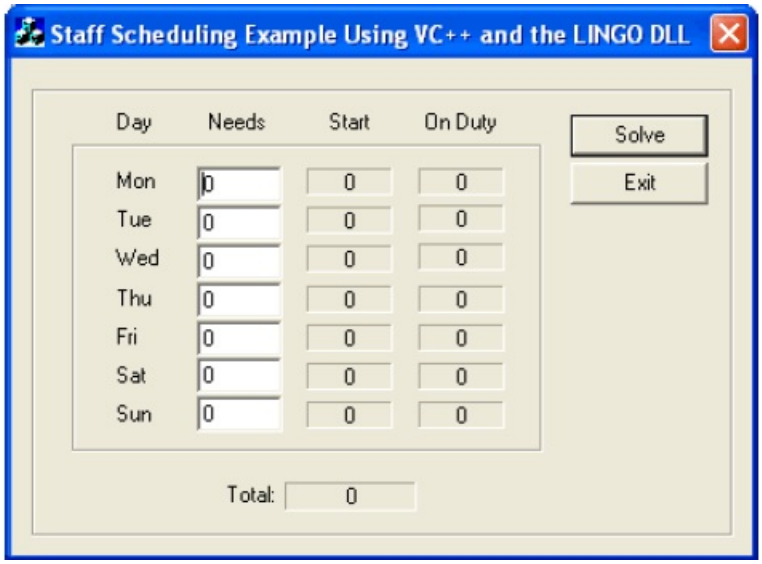
A biblioteca padrão de vínculo dinâmico DLL (*Dynamic Link Library*) no *Windows* é utilizada pelo LINGO. Desta maneira, é possível integrar com facilidade o *solver* de otimização com a linguagem de programação C++.

Normalmente os subproblemas resolvidos em uma aplicação, precisam compartilhar dados. Em outras palavras, um programa pode estar gerando como resultados os dados de entrada para outro programa. Existe uma função no LINGO denominada de @POINTER, que pode ser utilizada na seção *DATA*, por exemplo, a fim de se estabelecer um canal de comunicação entre o LINGO e os programas desenvolvidos na linguagem C++.

Ao chamar a LINGO DLL, é passada uma lista de ponteiros de memória. A maneira correta de utilização da função descrita é: @POINTER(n), onde n se refere ao enésimo ponteiro da lista [42]. Para uma melhor compreensão, imagine um exemplo de um estabelecimento comercial que funcione sete dias na semana. Os funcionários contratados deverão trabalhar cinco dias e folgar dois dias consecutivos, recebendo todos o mesmo salário semanal. Alguns dias são mais movimentados e, baseado em sua experiência, o administrador do negócio imagina quantos trabalhadores são necessários em um determinado dia da semana. O administrador deve determinar quantos funcionários começam a trabalhar em cada dia na semana, de forma a minimizar o número total de empregados [42].

Como o interesse é a demonstração de como se utilizar a função @POINTER, não será descrito o modelo de otimização completo. A Figura 9 mostra uma caixa de diálogo criada com o Visual C++ e o Visual Basic. Na coluna *Needs*, o usuário digita o número de pessoas necessárias e aperta o botão Solver. Neste momento, os dados digitados são extraídos a partir da caixa de diálogos e transferidos para o modelo desenvolvido em LINGO. Em outras palavras, a previsão de necessidade de pessoal constitui a lista de ponteiros de memória.

As colunas *Start* (funcionários que devem iniciar a cada dia) e *On Duty* (funcionários de plantão) deverão ser preenchidas com os valores obtidos após uma execução do modelo LINGO. O mesmo acontece com o campo *Total*, reservado para se computar o número de empregados necessários no início de cada dia.



Fonte: [42]

Figura 9 – Caixa de Diálogo Antes da Resolução do Modelo em LINGO

Um trecho do modelo é descrito na Figura 10.

```
DATA:
    NEEDS = @POINTER( 1);
    @POINTER( 2) = START;
    @POINTER( 3) = ONDUTY;
    @POINTER( 4) = OBJECTIVE;
    @POINTER( 5) = @STATUS();
ENDDATA
```

Fonte: [42]

Figura 10 – Exemplo de Utilização da Função @POINTER

NEEDS, *START* e *ONDUTY* são atributos de um conjunto utilizado no modelo. Seus valores ocupam as posições 1, 2, e 3 da localização de memória, respectivamente. Observa-se que o atributo *NEEDS* está ao lado esquerdo da igualdade. Isto se deve ao fato de o LINGO necessita dos valores digitados pelo usuário na caixa de diálogos, para que o modelo seja resolvido. *OBJETIVE* se refere ao valor da função objetivo a ser minimizada, ocupando a posição 4 da localização de memória. A função @STATUS é utilizada para o status do processo de solução do modelo. A Figura 11 mostra os códigos retornados com as suas devidas interpretações.

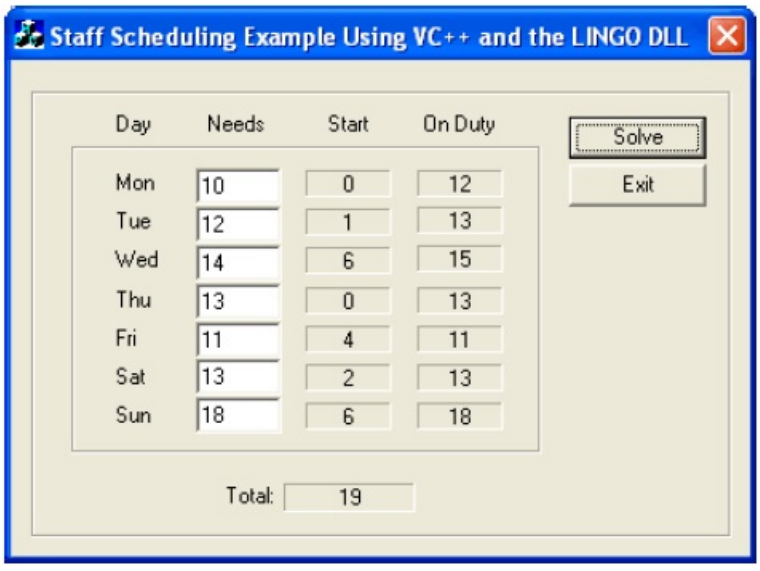
Em muitos casos, quando não são retornados os códigos 0, 4, 6 ou 8, não há a exportação dos dados para os locais de memória. Isto se torna claro, visto que nestas condições a solução encontrada possui grandes chances de não ser verdadeira.

@STATUS Code	Interpretation
0	*Global Optimum* - The optimal solution has been found.
1	*Infeasible* - No solution exists that satisfies all constraints.
2	*Unbounded* - The objective can be improved without bound.
3	*Undetermined* - The solution process failed.
4	*Feasible* - A feasible solution was found that may, or may not, be the optimal solution.
5	*Infeasible or Unbounded* - The preprocessor determined the model is either infeasible or unbounded. Turn off presolving and re-solve to determine which.
6	*Local Optimum* - Although a better solution may exist, a locally optimal solution has been found.
7	*Locally Infeasible* - Although feasible solutions may exist, LINGO was not able to find one.
8	*Cutoff* - The objective cutoff level was achieved.
9	*Numeric Error* - The solver stopped due to an undefined arithmetic operation in one of the constraints.

Fonte: [42]

Figura 11 – Interpretação dos Códigos Retornados pela Função @STATUS

Finalmente, a Figura 12 mostra a caixa de diálogo devidamente preenchida com os dados exportados pelo LINGO.



Fonte: [42]

Figura 12 – Caixa de Diálogo Após da Resolução do Modelo em LINGO

2.3 ROBÔS MÓVEIS TERRESTRES

Nesta seção, faz-se uma abordagem geral sobre uma das plataformas robóticas móveis mais utilizadas na academia. Este trabalho é voltado para os robôs móveis autônomos terrestres, e a familiarização das pessoas com os veículos de quatro rodas acaba justificando o grande uso destes robôs.

Os robôs de serviço que realizam as tarefas de monitoramento podem ser modelados como uniciclo ou biciclo, dependendo do número de rodas. A Figura 13 mostra um típico robô, o qual é utilizado como base para os estudos envolvidos neste trabalho. O veículo é desenvolvido pela MobileRobots® e é denominado de Pioneer 3-DX. Possui duas rodas dianteiras, controladas por motores DC reversíveis, e uma roda "livre" localizada na parte traseira do veículo. Esta plataforma contém componentes para sensoriamento e navegação nos ambientes.

Fonte: [43]

Figura 13 – Robô Pioneer 3-DX

Este tipo de base robótica é sub-atuada, ou seja, possui um número de atuadores menor que o número de graus de liberdade. Isto acontece pelo fato de não existir uma aceleração lateral, seu tipo de roda se move apenas para frente ou para trás, podendo estes veículos, no máximo, desenvolverem uma aceleração rotacional que os permitem realizar curvas e percorrer diversos tipos de caminhos.

Existem robôs móveis, cujas rodas possuem, além dos movimentos tradicionais, um suporte para a locomoção lateral. Uma base robótica deste tipo pode ser vista na Figura 14. Este robô pode ser diretamente aplicado às tarefas de recarga, já que possui um braço mecânico para atuar na troca das baterias dos robôs de serviço.

Fonte: [44]

Figura 14 – Robô KUKA

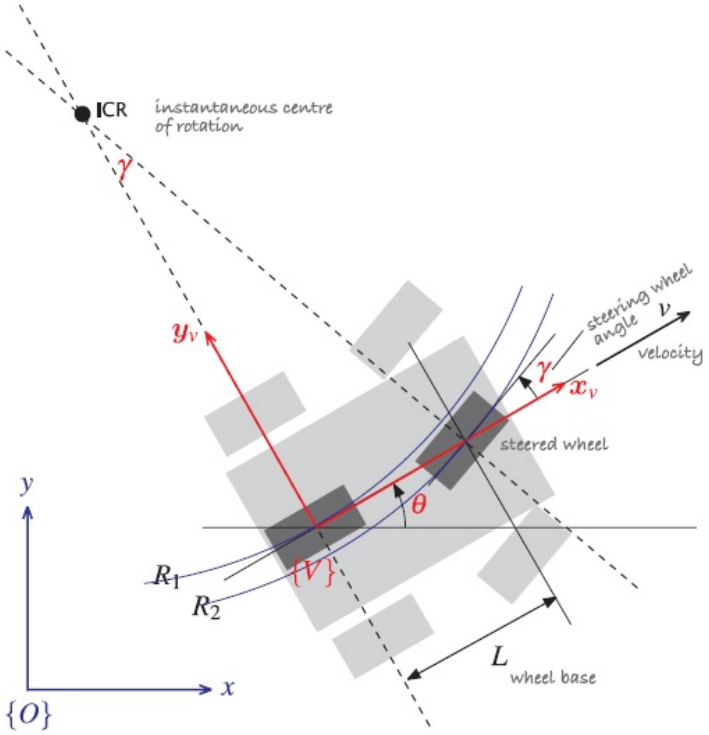
2.3.1 Robô *Car-like*: modelo cinemático

Em robótica, veículos do tipo *car-like* pertencem a classe dos chamados sistemas não-holonômicos. Este tipo de sistema é caracterizado por possuir limitações de movimento, o que leva a uma restrição não integrável, ou seja, não se consegue reduzi-la a uma restrição de posição [40]. Por exemplo, estes veículos podem se locomover para frente ou para trás, através de uma velocidade linear v_x. Podem desenvolver uma velocidade angular v_θ para realizar curvas. Porém possuem uma limitação de movimento em relação ao eixo y, de forma que v_y é igual a zero. Isto caracteriza uma restrição não integrável e, por tanto, o sistema é não-holonômico.

O modelo utilizado para a representação do robô "*Car-like*" é conhecido como biciclo. A Figura 15 ilustra este veículo inserido em um ambiente 2-D. A aproximação do modelo biciclo é representada pelos retângulos em cinza escuro. Nesta figura, o sistema de coordenadas {V}, em vermelho, está relacionado ao veículo. O eixo x aponta para a frente do robô e a sua velocidade é v. A velocidade na direção do eixo y é igual a zero, devido às limitações provocadas pela conservação do momento angular, ou seja, o veículo não se desloca lateralmente.

O sistema global de coordenadas {O}, no qual são representados os parâmetros que configuram o robô (x, y, θ), é denotado pela cor azul na figura. Além disso, ICR (*Instantaneous Centre of Rotation*) é o ponto de referência quando o veículo realiza um caminho circular. Os raios R_1 e R_2 ligam ICR a cada uma das rodas, traseira e dianteira, respectivamente. Desta maneira, a velocidade angular do robô é dada pela equação 2.1.

$$\dot{\theta} = \frac{v}{R_1} \tag{2.1}$$



Fonte: [40]

Figura 15 – Modelo Biciclo e Seus Eixos Coordenados

Com:

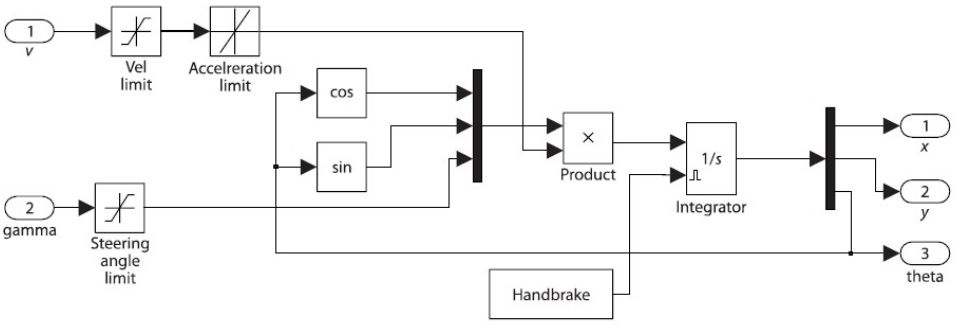
$$R_1 = \frac{L}{\tan\gamma}$$

O ângulo γ é limitado mecanicamente. O menor valor de R1 implica no máximo ângulo de rotação. O comprimento do veículo é dado por L. As velocidades do robô no sistema geral de coordenadas são dadas por:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\gamma \end{aligned} \tag{2.2}$$

O modelo biciclo pode ser esquematizado no Simulink®, do MATLAB®, de acordo com a Figura 16 [40].

Este modelo cinemático implementa o conjunto de equações 2.2. A velocidade linear v e o ângulo γ são limitados. O *reset* do bloco *Integrator* e a condição inicial são



Fonte: [40]

Figura 16 – Modelo Biciclo No Simulink®

fornecidos por uma constante através do bloco *Handbrake*. As Figuras 17 e 18 mostram um exemplo do funcionamento do modelo cinemático.

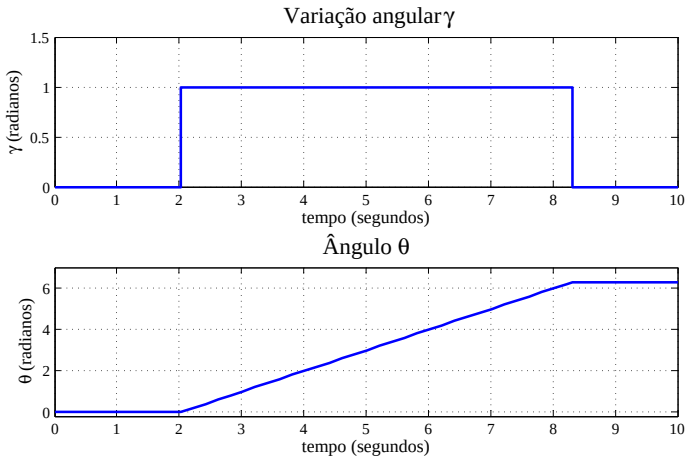


Figura 17 – Sinais de Entrada e Saída Angulares em um Modelo Biciclo

Neste pequeno exemplo, é adotada uma velocidade constante de 1m/s para o robô. Na Figura 17 tem-se o sinal de entrada γ e a correspondente posição de saída θ. Nota-se que mantendo um ângulo γ de 1 radiano, o robô tem uma variação θ de aproximadamente 6,28 radianos. A Figura 18 mostra o deslocamento do veículo no plano xy.

2.3.2 **Robô Uniciclo: modelo cinemático**

A plataforma Pioneer 3-DX pode ser modelada cinematicamente como uniciclo. As equações são bem parecidas com as do modelo biciclo discutido anteriormente, porém, o modelo uniciclo permite que o robô mude a sua orientação mesmo com a velocidade linear (v_x) igual a zero. Isso acontece devido a presença de uma roda "livre", localizada na parte dianteira do veículo, que serve somente de apoio para o robô. Na Figura 19 tem-se o

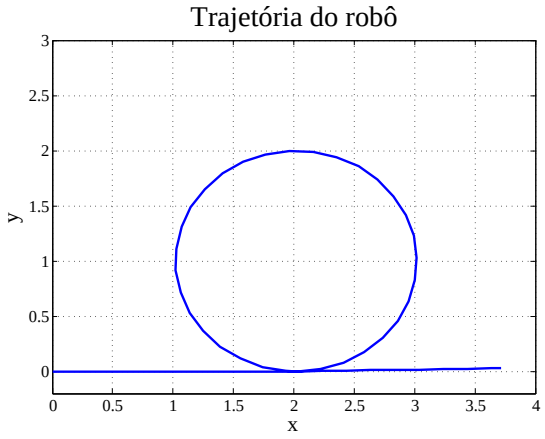


Figura 18 – Posição do Robô no Plano xy

esquema básico do modelo uniciclo. As velocidades linear e angular (v e ω) são as entradas e, como saídas, têm-se a posição e a orientação (x, y, θ).

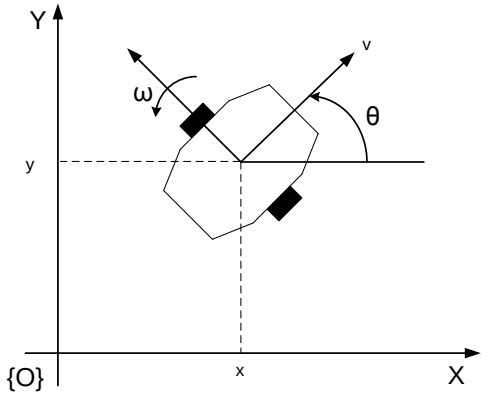


Figura 19 – Modelo Uniciclo e Seus Eixos Coordenados

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{2.3}$$

No capítulo de resultados e discussões é tratado objetivamente o controlador implementado nas simulações. Na prática pode-se utilizar, por exemplo, técnicas de fusão sensorial para que os robôs tenham as suas posições estimadas com uma maior precisão. Neste sentido, pode ser interessante o uso do filtro de Kalman Estendido para a redução de incertezas na estimação das posições.

2.4 ROS (ROBOT OPERATING SYSTEM)

A grande diversidade de plataformas robóticas existentes faz com a tarefa de se desenvolver *softwares* robustos, de aplicações gerais, seja muito difícil. Isto acontece porque estes robôs possuem *hardwares* diferenciados, variados tipos de sensores, de atuadores e características muito específicas. Desta maneira, torna-se complicado um grupo de uma determinada instituição ou laboratório desenvolver, por conta própria, todas as ferramentas necessárias para o perfeito funcionamento dessa ampla variedade de dispositivos [45].

O ROS foi desenvolvido com o intuito de impulsionar e facilitar o desenvolvimento da robótica, com uma filosofia de cooperação. Seu desenvolvimento se deu, originalmente, em 2007 por SAIL (*Stanford Artificial Intelligence Laboratory*). Em 2008 passou a ser gerenciado pela *Willow Garage*, onde teve seus conceitos amplamente divulgados. Através da colaboração de pesquisadores com bastante experiência na área, deram-se as criações de seus pacotes e softwares fundamentais, desenvolvidos em campo aberto com uma licensa *open-source* BSD (*Berkeley Software Distribution*) permissiva [45]. Atualmente, o desenvolvimento e a distribuição do ROS são apoiados por uma organização independente sem fins lucrativos, "*Open Source Robotics Foundation*"(OSFR).

O *framework* ROS torna a forma de lidar, com diferentes plataformas e suas peculiaridades, muito mais simples e flexível. Grupos de pesquisas no mundo inteiro podem contribuir para o desenvolvimento de *softwares* de aplicações robóticas, cada qual com a sua especialidade, seja visão computacional, mapeamento de ambientes internos, uso de mapas para a navegação, entre outros [45]. Pode-se, então, reaproveitar códigos desenvolvidos por outros grupos ou contribuir também com novas descobertas e aplicações. Esta é uma vantajosa característica, que faz com que este *software*, que abrange uma coleção de ferramentas, bibliotecas e convenções para os mais variados robôs existentes no mercado [45], seja utilizado em todo o mundo. Além da família *Pioneer*, ROS oferece suporte a uma série de robôs. A Figura 20 mostra algumas das plataformas suportadas por este *framework*.

Fonte: [45]

Figura 20 – Algumas Plataformas Robóticas Suportadas pelo ROS

Da esquerda pra direita, tem-se o humanóide *Aldebaran Nao*, o manipulador *Shadow*

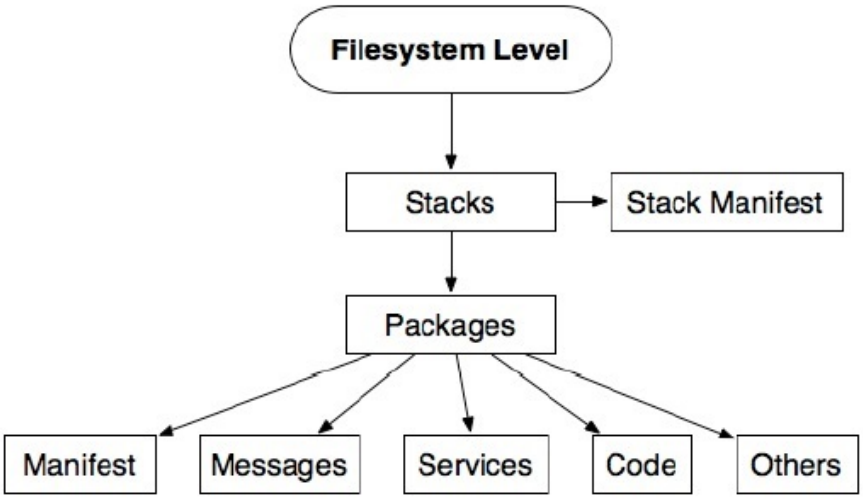
Dexterous Hand, o veículo aéreo *Erle-copter* e a plataforma omnidirecional *Neobotix mpo-500*. Devido a toda esta variedade, uma série de dispositivos sensores e atuadores são diariamente incrementados para serem suportados pelo ROS [46].

Uma importante característica do ROS é que suas ferramentas básicas podem ser aprendidas e utilizadas por entusiastas em robótica, não só por grupos de pesquisas ou companhias industriais. Em outras palavras, mesmo pessoas com pouquíssima experiência em robótica podem desenvolver pequenas aplicações em seus próprios protótipos. Completos tutoriais de instalação e uso do ROS podem ser encontrados em [45]. Nas próximas subseções são apresentadas algumas de suas características e ferramentas, bem como a sua arquitetura.

2.4.1 **O sistema de arquivos ROS**

Apesar do nome *Robot Operating System*, ROS não é considerado um sistema operacional. Entretanto, apresenta alguns recursos de um sistema operacional padrão, como: abstração de *hardware*, controle de dispositivos de baixo nível, implementação de funcionalidades de uso comum, trânsito de mensagem entre processos, além do gerenciamento de pacotes [46].

Sua arquitetura é baseada em grafos com uma topologia centralizada, sendo que os nós são os processos que utilizam o ROS. A estrutura geral do sistema de arquivos em ROS pode ser vista na Figura 21.



Fonte: [46]

Figura 21 – Sistema de Arquivos no ROS

Observa-se a estruturação em pastas, sendo composto basicamente por *Stacks* e *Packages*. Os *Stacks* são agrupamentos de pacotes, que podem ter funcionalidades diversas.

Uma de suas principais funções é simplificar o compartilhamento de códigos entre usuários do mundo inteiro, característico essencial do ROS. Os arquivos intitulados de *manifest* contêm, tanto para os *Packages* quanto para os *Stacks*, informações sobre suas licensas, suas dependências, *flags* de compilação, no caso dos pacotes, entre outras.

Nos *Packages* encontram-se os arquivos de configurações, os códigos fonte dos programas para os processos executados durante as aplicações, ou seja, o mínimo necessário para o gerenciamento de um projeto em robótica com o ROS. Entre as diversas linguagens de programação suportadas pelo ROS, estão o C++ e o Python. Como descrito anteriormente, os processos controlados neste trabalho são implementados em C++.

É neste nível da organização que se encontram as mensagens que circulam entre os nós, fazendo com que as aplicações funcionem exatamente como projetadas. A linguagem de descrição das mensagens é simplificada para interpretar os dados utilizados. Os arquivos de mensagens possuem extensão `.msg` e ficam armazenadas na pasta `msg/`. O desenvolvedor do projeto pode criar as suas próprias mensagens e também utilizar mensagens predefinidas pelo ROS. A Figura 22 mostra alguns tipos de mensagens junto com as suas interpretações em C++ e em Python.

Primitive type	Serialization	C++	Python
bool	Unsigned 8-bit int	uint8_t	bool
int8	Signed 8-bit int	int8_t	int
uint8	Unsigned 8-bit int	uint8_t	int
int16	Signed 16-bit int	int16_t	int
uint16	Unsigned 16-bit int	uint16_t	int
int32	Signed 32-bit int	int32_t	int
uint32	Unsigned 32-bit int	uint32_t	int
int64	Signed 64-bit int	int64_t	long
uint64	Unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ASCII string (4-bit)	std::string	string
time	Secs/nsecs signed 32-bit ints	ros::Time	rospy.Time
duration	Secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

Fonte: [46]

Figura 22 – Mensagens Padrões em ROS

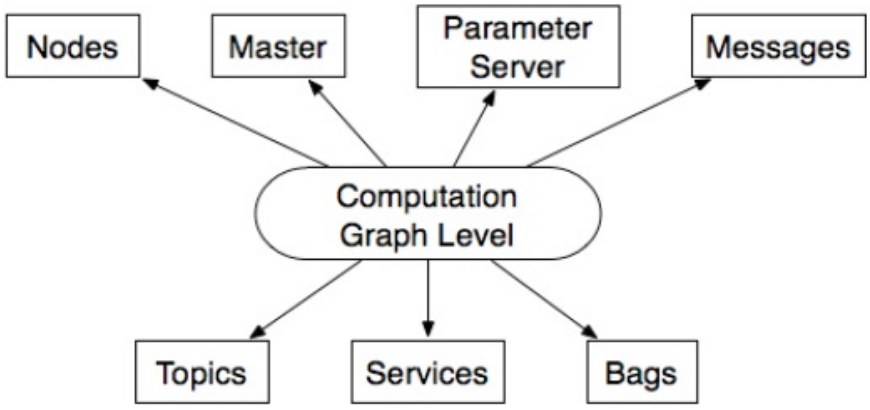
Observa-se que as mensagens são constituídas de campos e constantes. Por exemplo, uma mensagem que informe a velocidade de um robô pode ser definida por `float32 vel`, onde `vel` é uma constante que representa o dado velocidade e `float32` é o campo que explicita o tipo de dado. Um tipo especial de arquivo em ROS é o *Header*, utilizado para

adicionar características como *timestamp* e *frame* [46], como, por exemplo, uma mensagem do tipo `string frame_id`. Isto possibilita gravar dados sobre o que acontece com os robôs ao longo das execuções dos processos. Dar-se-á uma atenção especial a este assunto na próxima subseção, devido a sua importância em aplicações práticas e até mesmo em ambientes para simulações.

Ainda na Figura 21 observa-se a presença de uma estrutura chamada *Service*, um subdiretório (`srv/`) do *Package*. Esta é uma outra forma de comunicação entre os nós, representada através de requisições e respostas entre os próprios nós. Este tipo de arquivo possui a extenção `.srv`.

2.4.2 **Conceitos básicos: comunicação na rede do ROS**

Além dos sistemas de arquivos, visto anteriormente, o ROS possui também um nível de estrutura denominado de *Computation Graph Level*, como pode ser visto na Figura 23.



Fonte: [46]

Figura 23 – Nível *Computation Graph Level*

Nos *Bags* ficam armazenados importantes dados que podem ser utilizados no desenvolvimento e teste de algoritmos. São coletados, por exemplo, dados de sensores, os quais são complicados de serem coletados e podem auxiliar muito na apuração de resultados experimentais.

Todos os processos em uma determinada aplicação são conectados em rede. Em função desta característica, todos os nós podem interagir entre si, visualizando informações que estão sendo enviadas na rede, enviando e recebendo também estas informações de dados. O servidor de parâmetros contém dados permanentes responsáveis pela configuração e inicialização dos nós. Normalmente, um sistema robótico completo possui muitos nós

executando diversos processos, é uma alternativa melhor do que se usarem poucos nós controlando muitos processos [46].

O conceito de nó *Master* é fundamental para o entendimento do trâmite de dados através da rede criada por ROS. Sem este nó principal é impossível estabelecer uma comunicação entre os processos envolvidos, serviços e mensagens, por exemplo. Uma importante característica é que esses processos podem ser executados em computadores diferentes, conectados em rede.

A Figura 24 ilustra esta forma de comunicação centralizada. O nó em vermelho, denominado de `roscore` (*Master*), é um processo único que gerencia a comunicação entre todos os nós. Este processo principal é definido por uma variável de ambiente denominada *ROS_MASTER_URI*.

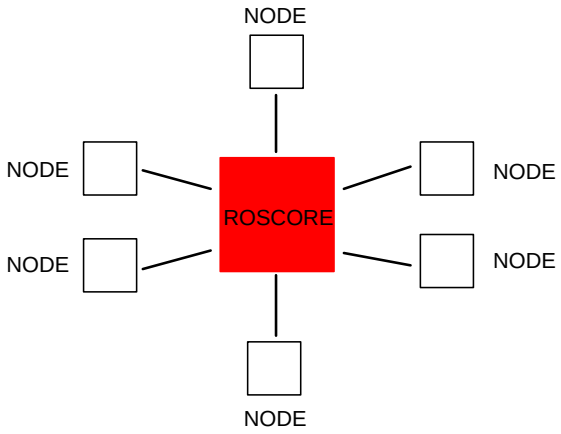


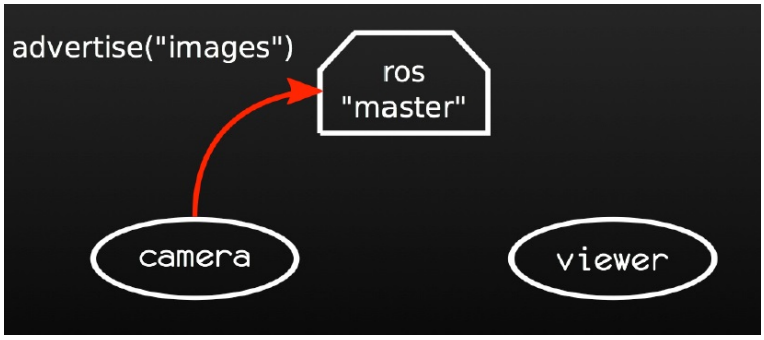
Figura 24 – Sistema Geral de Comunicação Entre os Nós por Mensagens

Ao enviar um dado através da rede, diz-se que o nó está publicando em um Tópico (*Topics* na Figura 23). Isto ocorre porque cada mensagem deve ter um nome associado à mesma para ser transmitida pela rede. Para que um nó receba mensagens publicadas por outros nós, é necessário que o mesmo se inscreva no Tópico de interesse. O gerenciamento completo desta rede criada pelo ROS é realizado pelo `roscore`.

As Figuras 25, 26, 27 e 28 ilustram um exemplo básico de comunicação entre dois processos. Trata-se da simples reprodução de imagens de uma câmera, que pode ser a *webcam* de um *laptop*, por exemplo.

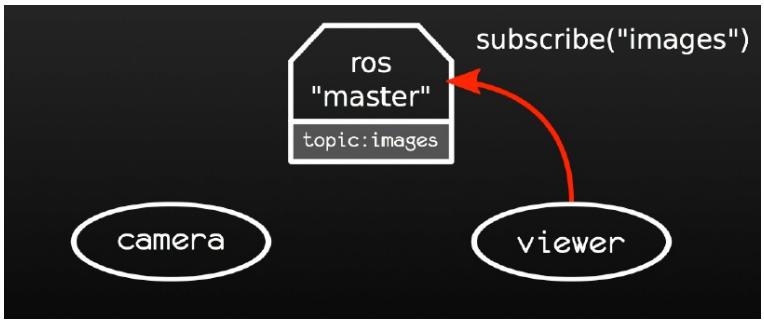
O sistema é constituído por três nós: o `roscore`, o nó `camera` e o nó `viewer`. A comunicação é realizada de acordo com seguinte sequência de operações:

1. O nó `camera` envia uma solicitação ao `roscore` informando o Tópico a ser publicado na rede. Neste caso com o nome de "`images`";



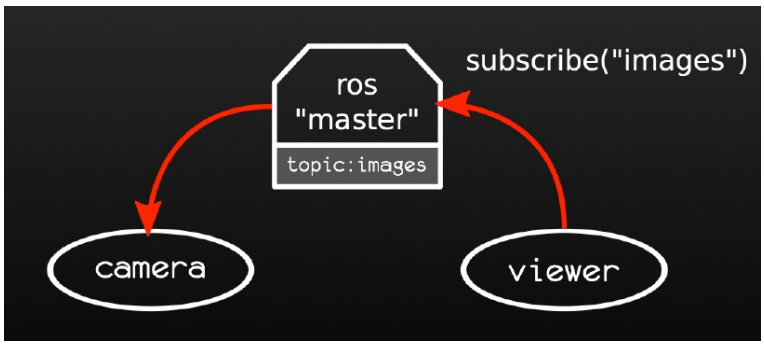
Fonte: [47]

Figura 25 – Primeiro Passo para a Comunicação Entre Dois Nós



Fonte: [47]

Figura 26 – Segundo Passo para a Comunicação Entre Dois Nós

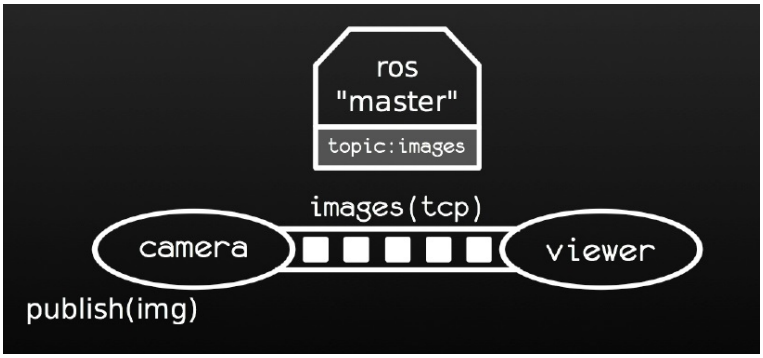


Fonte: [47]

Figura 27 – Terceiro Passo para a Comunicação Entre Dois Nós

2. O nó "*Master*" registra o Tópico e o coloca à disposição de todos os nós que desejem receber informações sobre esta mensagem;

3. Os nós interessados, no caso apenas o nó `viewer`, se inscrevem no Tópico, enviando uma solicitação ao `roscore`;



Fonte: [47]

Figura 28 – Quarto Passo para a Comunicação Entre Dois Nós

4. Por fim, a comunicação é estabelecida e as imagens circulam normalmente pela rede;

Como especificado anteriormente, não é necessário que os nós sejam executados em um mesmo computador. A transmissão de dados utilizados pelo ROS pode ser realizada pelo protocolo TCP/IP (*Transmission Control Protocol* e *Internet Protocol*) ou pelo protocolo UDP (*User Datagram Protocol*).

Outra forma de transmissão de dados em ROS, é a comunicação direta entre os nós. Neste caso, existem os Serviços (*Services*), que já foi mencionado anteriormente. O seu funcionamento é análogo ao de uma estrutura cliente-servidor, muito utilizada em redes de computadores. Uma ilustração desta forma de comunicação entre os nós pode ser vista na Figura 29.

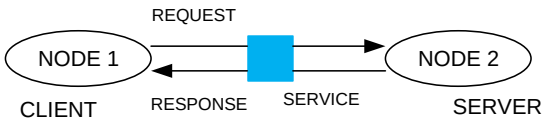


Figura 29 – Comunicação Baseada em Serviços

2.4.3 **A Comunidade ROS**

O ROS fornece os recursos necessários para que haja a troca de experiências entre os desenvolvedores de *softwares*, sejam pesquisadores, estudantes ou até mesmo iniciantes na robótica. Fazem parte destes recursos [46]:

- Distribuições: a cada seis meses, aproximadamente, é lançada uma nova versão do ROS. O papel destas versões atualizadas é semelhante ao das distribuições Linux.

- Repositórios: qualquer instituição pode desenvolver seus próprios robôs e contribuir com os seus softwares. ROS conta com uma rede federada de repositórios.
- ROS Wiki: local onde os usuários podem fazer as suas próprias documentações a respeito do ROS. Neste ambiente, pode-se fazer atualizações e até mesmo correções de outros documentos, além de escrever tutoriais. Este recurso pode ser acessado em [45].
- Fórum de discussão: canal de comunicação entre os usuários do ROS. Neste local, encontram-se as perguntas mais freqüentes e é possível a interação entre os usuários para a troca de todos os tipos de informações a respeito do ROS. Pode-se ter acesso a esta ferramenta em [48].

2.5 CONCLUSÕES

Este capítulo é de fundamental importância para uma melhor compreensão do trabalho. Foram abordados temas de diversas áreas, as quais podem ser integradas em uma grande aplicação.

As duas primeiras seções fazem uma ligação direta com o problema abordado neste trabalho. Os conceitos de grafo descritos abrangem os aspectos mais importantes para o entendimento das tarefas dos robôs de recarga. Na seção dedicada a otimização, dá-se uma atenção especial ao *solver* utilizado. Esta parte do trabalho está fortemente ligada ao Capítulo 4, no qual é descrito detalhadamente o algoritmo de otimização utilizado na tarefa de recarga.

Como a ideia é a aplicação em robôs autônomos terrestres, dedicou-se uma pequena seção para este tipo de veículo. Representaram-se, cinematicamente, os modelos biciclo e uniciclo. Um pequeno exemplo de funcionamento do modelo biciclo foi apresentado.

Visando a integração entre as técnicas utilizadas neste trabalho, apresentou-se o *framework* ROS, uma ferramenta cada vez mais utilizada em robótica. Por meio deste *software* pode-se gerenciar diversos nós, cada qual com a sua finalidade na aplicação global. Os programas de otimização têm os seus próprios nós, comunicando-se com outros programas em C++ utilizando a função de integração descrita.

3 COORDENAÇÃO DOS ROBÔS DE SERVIÇO

A proposta principal do trabalho é dar suporte para que as tarefas desempenhadas por um grupo de robôs sejam persistentes. Como mencionado anteriormente, os robôs de serviço são responsáveis pela exploração, monitoramento ou vigilância contínua dos ambientes de aplicações.

No desenvolvimento deste trabalho, são utilizados robôs de serviço cuja tarefa é simplesmente percorrer os seus caminhos fechados, como num monitoramento de um grande ambiente. O problema é exposto e modelado matematicamente nas seções seguintes, focando na coordenação dos veículos de forma que as tarefas sejam realizadas com segurança.

3.1 ABORDAGEM DO PROBLEMA: POSSÍVEIS COLISÕES

A coordenação dos veículos no desempenho das tarefas se faz necessária devido à topologia dos caminhos utilizados. A Figura 30 ilustra um típico ambiente ao qual se aplicam as técnicas descritas no trabalho. É interessante ressaltar que o grupo de robôs de recarga será introduzido e detalhado no capítulo seguinte, para facilitar a compreensão e o objetivo do trabalho como um todo.

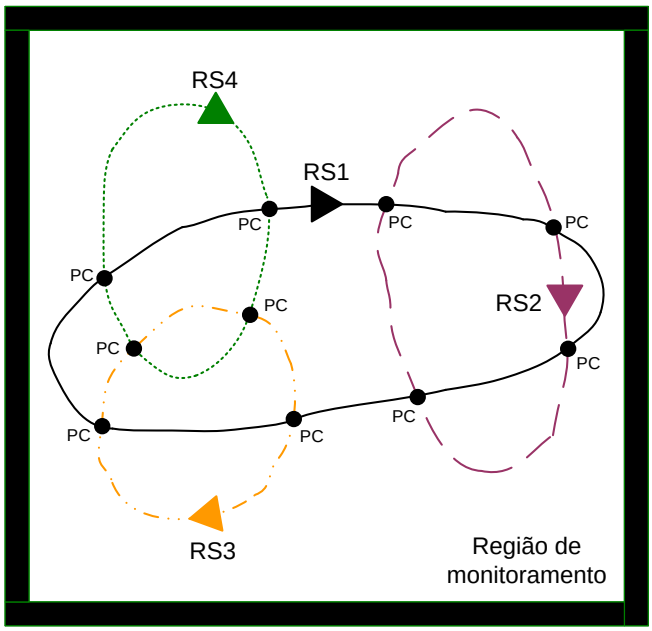


Figura 30 – Robôs de Serviço em Ambiente de Trabalho

De acordo com a ilustração, têm-se regiões de um ambiente monitoradas por quatro robôs de serviço *(RS1, RS2, RS3 e RS4)*.

Cada veículo percorre um caminho fechado e completamente conhecido. Pode-se observar pontos, denominados PC (Ponto de Colisão). Desta forma, para que a tarefa de monitoramento seja realizada com segurança, é necessário que um controlador de alto nível atue no sentido de se evitarem colisões entre os veículos. O objetivo é que os robôs percorram os seus devidos caminhos, ciclicamente, sem alterações de rotas.

Os subcaminhos, compreendidos pelos trechos entre dois pontos de colisões sucessivos nas rotas dos robôs, têm de ser percorridos em um determinado intervalo de tempo. O controlador deve atuar na velocidade de cada veículo para assegurar que não ocorrerá nenhum choque [18].

3.2 MODELAGEM MATEMÁTICA

Os perfis de velocidades são gerados para todos os veículos ao longo de seus percursos. Cada subcaminho é percorrido com uma velocidade média, ajustada de acordo com um algoritmo de otimização. Isto implica na realização das tarefas com o máximo de segurança quanto à questão de possíveis colisões.

Com as velocidades médias determinadas para cada subcaminho, têm-se também os intervalos de tempo para que os robôs os percorram, visto que os percursos são completamente conhecidos. O tempo de percurso para cada veículo, em um subcaminho, é encontrado de forma que uma margem de segurança de tempo para uma possível colisão seja maximizada. A técnica utilizada neste ponto foi proposta em [18]. Busca-se maximizar o menor intervalo de tempo Δt entre a passagem dos robôs por cada um dos pontos de colisão.

Os perfis de tempo e velocidades médias são obtidos através da resolução de um programa linear inteiro misto (MILP), cuja estrutura de otimização é modelada matematicamente a seguir.

Deseja-se maximizar Δt sujeito a:

- **Limites de Margem de Tempo:**

$$|t_j[i_1, k_1] - t_j[i_2, k_2]| \geq \Delta t \tag{3.1}$$

$$t_j[i_1, 0] - t_j[i_2, 0] \geq \Delta t - \xi[i_1, i_2, j]\, B[i_1, i_2, j] \tag{3.2}$$

$$t_j[i_1, 0] - t_j[i_2, 0] \leq -\Delta t + \xi[i_1, i_2, j]\,(1 - B[i_1, i_2, j]) \tag{3.3}$$

$$\forall j \in \mathcal{J}, \forall i_1, i_2 \in \mathcal{S}_j, B[i_1, i_2, j] \in \{0, 1\}$$

Onde:

$t_j[i,k]$: tempo que o robô i alcança o ponto j no ciclo k.

$\mathcal{J}$: conjunto dos pontos de colisão j.

$\mathcal{S}_j$: conjunto de robôs i que podem colidir no ponto j.

Nas Inequações 3.2 e 3.3, há uma variável binária B, responsável por determinar qual delas estará ativa durante o processo de otimização. A constante ξ (estimador a priori) recebe um alto valor para que as duas restrições sejam atendidas durante a execução do programa de otimização. Esse artifício é necessário devido ao fato de que a restrição de desigualdade 3.1 gera duas restrições exclusivas entre si.

Quando $B\left[i_1, i_2, j\right]$ recebe o valor lógico 1, tem-se o robô i_2 passando pelo ponto de colisão j primeiro que i_1. O contrário acontece quando B assume o valor lógico 0.

- **Periodicidade e Comensurabilidade de Ciclos:**

$$t_{l(i)}\left[i,1\right] - t_{l(i)}\left[i,0\right] = \lambda_i T \qquad \forall i \in \mathcal{S}_j \tag{3.4}$$

Onde:

$l(i)$: função que mapeia o robô i ao ponto inicial.

λ_i: número natural.

T: base de tempo.

Na matemática, duas quantidades são ditas comensuráveis quando a razão entre elas é igual a um número racional.

A restrição 3.4 é de suma importância para a garantia de segurança no desempenho da tarefa. Estabelecida uma base de tempo para todo o conjunto, esta equação fornece os tempos para que cada robô de serviço complete uma volta em torno de seu caminho.

Ciclos periódicos e incomensuráveis trazem uma incerteza no funcionamento do sistema. Neste caso, a ausência de colisões não pode ser garantida [18].

- **Restrições de Velocidade para Pontos no Mesmo Ciclo:**

$$\tau_{\min}\left[i,j\right] \leq t_{s(j,i)}\left[i,0\right] - t_j\left[i,0\right] \leq \tau_{\max}\left[i,j\right] \qquad \forall i \in \mathcal{R}, \forall j \in \mathcal{Y}_i \tag{3.5}$$

Onde:

$\mathcal{R}$: conjunto dos robôs i.

$\mathcal{Y}_i$: conjunto dos pontos de colisão presentes no caminho do robô i.

$s(j,i)$: função que retorna o próximo ponto na sequência.

$t_{s(j,i)}\,[i,0]$: tempo que o robô i alcança o próximo ponto em sua sequência.

A restrição 3.5 estabelece que cada robô de serviço, em seu percurso, deve atravessar cada subcaminho em intervalos de tempo compreendidos entre um valor mínimo (τ_{min})e um valor máximo (τ_{max}).

- **Restrições de Velocidade para o Último Ponto:**

$$t_{l(i)}\,[i,1] - t_{a(l(i),i)}\,[i,0] \geq \tau_{\min}\,[i, a(l(i),i)] \tag{3.6}$$
$$t_{l(i)}\,[i,1] - t_{a(l(i),i)}\,[i,0] \leq \tau_{\max}\,[i, a(l(i),i)] \tag{3.7}$$
$$\forall i \in \mathcal{R}, \forall j \in \mathcal{Y}_i$$

Onde:

$a(.,i)$: função que retorna o ponto imediatamente anterior na sequência dos pontos presentes no caminho do robô i.

Há uma particularidade em relação ao último ponto da sequência, pois, neste caso, o próximo ponto pertence ao ciclo posterior. Por tanto, estas inequações são responsáveis pela restrição do intervalo de tempo referente ao último subcaminho do percurso.

- **Condição Inicial:**

$$-T + \Delta t \leq t_j\,[i_1,0] - t_j\,[i_2,0] \leq T - \Delta t \tag{3.8}$$
$$\forall j \in \mathcal{J}, \forall i_1, i_2 \in \mathcal{S}_j$$

Como uma condição inicial, um robô de serviço não deve passar duas vezes consecutivas por um ponto de colisão j, sem que todos os outros veículos pertencentes a $\mathcal{S}_j$ tenham passado ao menos uma vez por este ponto.

- **Δt não Negativo:**

$$\Delta t \geq 0 \tag{3.9}$$

- **Convenção:**

$$t_{l(1)}\left[1,0\right] = 0 \tag{3.10}$$

Convenciona-se que os robôs de serviço iniciem suas tarefas no ponto inicial com tempo igual a zero.

3.3 CONCLUSÕES

Os robôs móveis autônomos podem ser usados nas mais diversas tarefas, como monitoramento, exploração e vigilância de ambientes. O ideal é que todas estas missões sejam realizadas com um longo prazo de duração, evitando-se ao máximo as interrupções.

Visando a aplicação de uma técnica para aumentar a durabilidade das tarefas desempenhadas por um grupo robótico, denominado robôs de serviço, este capítulo abordou uma possível situação prática. Os robôs de serviço percorrem seus caminhos, os quais são constituídos por pontos onde poderiam ocorrer colisões entre os veículos, devendo, desta forma, ter os seus movimentos coordenados com precisão.

Apresentou-se uma estrutura para otimizar a realização da missão, com o objetivo de maximizar o menor intervalo de tempo correspondente à passagem dos veículos por um mesmo ponto de colisão (PC). As restrições de igualdade e desigualdade foram devidamente explicadas, ressaltando a importância de cada uma no problema descrito.

Conclui-se que, dentre as restrições impostas ao problema, a questão da comensurabilidade entre os ciclos realizados pelos veículos em seus caminhos, é imprescindível para garantir o sucesso das missões quanto a questão da segurança dos veículos. Esta restrição é fundamental para que as tarefas sejam planejadas em um longo horizonte, com a possibilidade de planejamento baseado em ciclos que se repetem. As tarefas desempenhadas pelos múltiplos robôs podem ser consideradas persistentes. Artifícios matemáticos, como a introdução de variáveis binárias a fim de auxiliar no controle das velocidades, também são fundamentais na solução de problemas como este.

A determinação dos perfis de tempo e velocidade é o ponto chave para a continuidade do trabalho. O próximo capítulo aborda uma estratégia para se coordenar os dois grupos robóticos, ressaltando-se a importância e a aplicação dos tempos obtidos nesta primeira etapa de otimização.

4 CENÁRIO COMPLETO: GRUPOS ROBÓTICOS EM TAREFAS COOPERATIVAS

Na Seção 1.4, que trata de uma revisão da literatura, é apresentada uma série de aplicações para os robôs autônomos móveis. Em vários dos casos abordados, percebe-se uma clara intenção de que as tarefas desempenhadas pelos veículos utilizados sejam realizáveis em um longo prazo de duração. Porém, muitos dos trabalhos não abordam uma necessidade de reabastecimento ou de recarga dos robôs.

Este capítulo descreve detalhadamente uma técnica que consegue suprir de maneira eficiente esta questão, em casos semelhantes ao explicitado no capítulo anterior. As chamadas tarefas persistentes ou missões com planejamento de horizonte "infinito" ganham, enfim, o suporte necessário para que os objetivos sejam alcançados.

O algoritmo de otimização utilizado para resolver o problema de colisões entre os veículos responsáveis pelo serviço de monitoramento, fornece um precioso histórico de dados relacionados aos robôs de serviço. O perfil de tempo para cada um destes veículos permite a obtenção do momento exato em que os mesmos passam por diferentes pontos em seus caminhos.

Esta ferramenta e a forma como estes dados são gerados, justificam a técnica de recarga utilizada neste trabalho [11]. Tem-se, agora, o cenário completo, com dois grupos de robôs trabalhando de forma cooperativa em busca de um objetivo comum.

4.1 INTRODUÇÃO DOS ROBÔS DE RECARGA

No Capítulo 3 foi mencionado apenas um dos grupos robóticos pertencentes a uma operação de monitoramento, tratou-se dos robôs de serviço. A Figura 31 descreve um cenário completo de operação. Pode-se observar três caminhos, cada qual monitorado por um robô de serviço. Destacam-se, nesta nova situação, os robôs de recarga (RR) e a introdução de novos pontos de interesse ao longo dos caminhos dos veículos de serviço. Os chamados subcaminhos ficam, nesta nova abordagem, compreendidos entre pontos de colisão (PC) e também entre pontos de recarga (PR).

Para que as tarefas executadas pelos robôs de serviço se estendam ao maior tempo possível, é necessário que haja recarga de baterias, tarefa realizada pelos robôs de recarga. O desafio é garantir que esses robôs se encontrem com os robôs de serviço nos pontos de recarga, pré-definidos no planejamento inicial do caminho. Para isso, será implementado um algoritmo de otimização que encontrará a rota com o menor tempo de execução para a tarefa de recarga [11]. A discretização do ambiente com a escolha de pontos de recarga específicos torna possível a representação da tarefa dos robôs de recarga por um grafo.

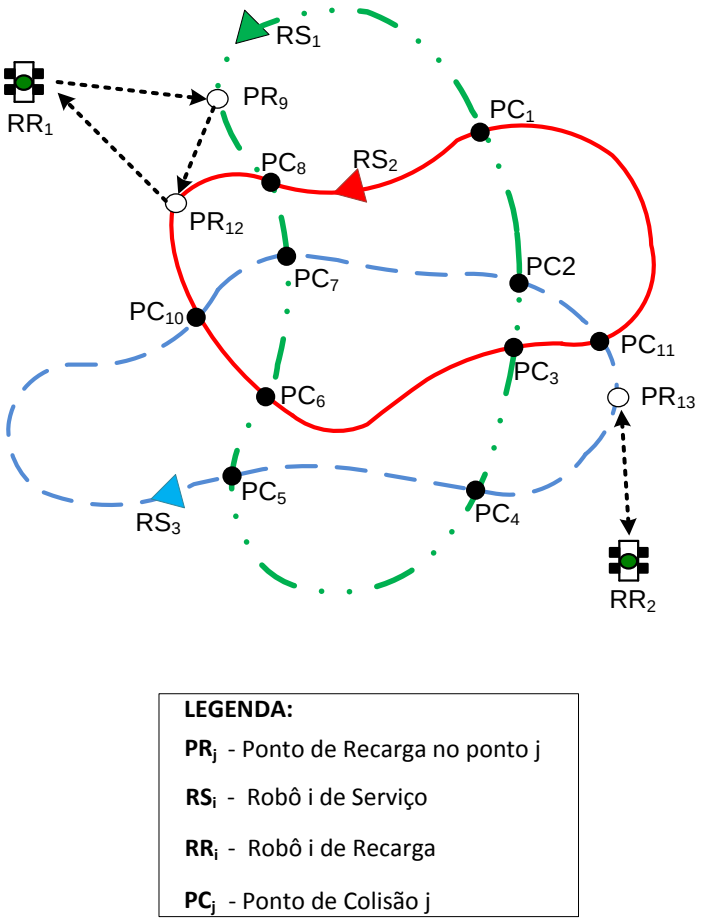


Figura 31 – Tarefa de Monitoramento: cenário completo

4.2 METODOLOGIA UTILIZADA

A solução global do problema pode ser dividida em duas estruturas de otimização. A primeira atua somente sobre os robôs de serviço e é responsável por controlar as suas velocidades médias a fim de maximizar o intervalo de tempo Δt, descrito anteriormente. Já a segunda age somente sobre os robôs de recarga, minimizando uma função objetivo a fim de encontrar um caminho ótimo para que seja realizada essa missão.

A Figura 32 ilustra o relacionamento entre os dois níveis de otimização. Os tempos de percursos subcaminhos (TPS), percorridos pelos robôs de serviço, são utilizados para determinar os custos de cada aresta do grafo. A partir destes custos, é possível determinar a viabilidade dos caminhos a serem seguidos pelos robôs de recarga.

Com a discretização dos pontos de carregamento, pertencentes aos respectivos caminhos de cada robô de serviço, uma abordagem baseada em grafo auxilia na formulação de um programa linear inteiro misto, que otimize as rotas de todos os robôs de recarga [11].

A Figura 33 mostra uma situação com dois robôs de serviço em caminhos diferentes

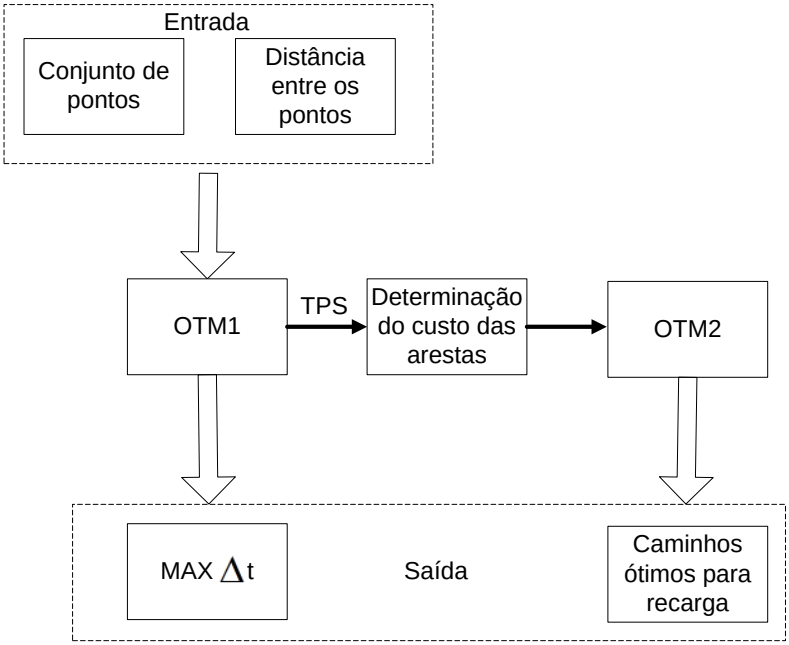


Figura 32 – Estruturas de Otimização

e um robô utilizado para recarga. Os vértices que constituem o grafo representativo, são formados pelos pontos de recarga indicados em cada caminho, onde podem ocorrer o contato entre um robô de serviço e o robô de recarga. Observa-se que cada robô de serviço possui o seu próprio conjunto de vértices, e que não há uma intersecção entre os pontos destes conjuntos. Esta situação ocorrerá mesmo que os robôs de serviço estejam se movendo em caminhos iguais, cada qual, neste caso, podendo ser recarregado em uma região específica.

As ligações entre os vértices de conjuntos diferentes formam as arestas direcionadas do grafo por meio das quais é planejada a rota para o robô de recarga. Na Figura 33 pode-se observar todas as combinações possíveis de arestas. É importante salientar que, cada robô de serviço, deverá ser atendido uma única vez durante o trajeto do veículo responsável pelo processo de recarga. Desta forma, estes robôs serão auxiliados em apenas um de seus pontos destinados a tal operação. Além disso, o ponto de partida do robô de recarga corresponde ao vértice inicial do grafo, podendo ser ligado a todos os pontos de recarga. Uma aresta apresenta uma travessia factível, quando o robô de recarga chega ao vértice de destino em tempo hábil para encontrar o seu respectivo robô de serviço.

A fim de que seja escolhido o melhor caminho para o robô de recarga, são atribuídos custos não negativos às arestas, visando a minimização de uma certa função objetivo. Deseja-se, neste trabalho, minimizar o tempo total para a recarga. Cada custo associado a sua devida aresta é calculado pela diferença entre os instantes de tempos em que os robôs de serviço cruzam os vértices envolvidos. Não é considerada a possibilidade de colisão

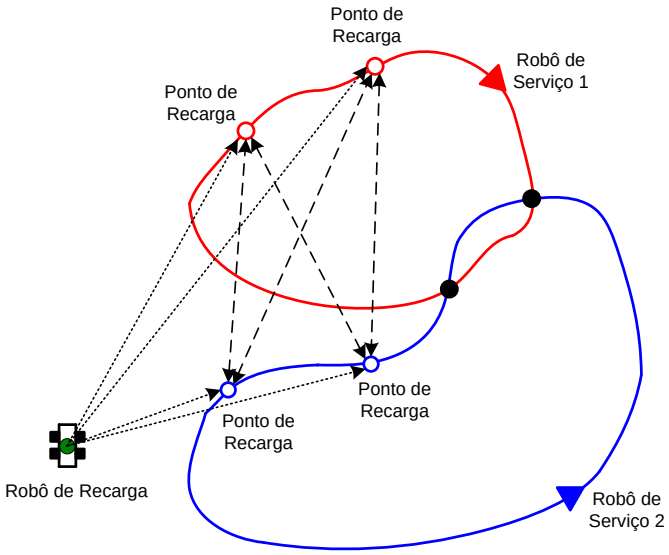


Figura 33 – Arestas do Grafo

entre robôs de recarga e robôs de serviço, ou entre os próprios robôs de recarga.

4.2.1 **Formulação MILP**

A formulação do programa linear inteiro misto, proposta por Mathew et al. [11], pode ser descrita da seguinte maneira:

Seja um grafo $G = (V, A, c)$. V representa o conjunto de agrupamentos de vértices $(V_0, V_{RS_1}, V_{RS_2}, ..., V_{RS_w})$, sendo w o número de robôs de serviço e V_0 o conjunto dos pontos iniciais dos robôs de recarga. A é o conjunto de arestas e $c : A \rightarrow \Re$ são os custos, já discutidos anteriormente.

Considere ainda os vértices v_m e v_n , em que $m \in V_{r_1}$, $n \in V_{r_2}$, $r_1 \neq r_2$ e $r_1, r_2 \in \{RS_1, ..., RS_w\}$. Define-se uma variável binária de decisão $x_{mn\delta} \in \{0, 1\}$, $\delta \in \{1, ..., u\}$, onde u refere-se ao número de robôs de recarga, com $x_{mn\delta} = 1$ quando o vértice v_n é visitado após v_m. Os custos de cada aresta associados às variáveis de decisão são denotados por c_{mn}, recebendo valores altos quando não há uma travessia factível. Nas equações a seguir, o índice 0 para cada robô de recarga RR_u, denota as suas posições iniciais. Um vértice artificial indexado por d é utilizado para caracterizar o fim de cada rota.

- **Condições para a factibilidade de uma aresta:**

$$\frac{||PR_{r2,n} - PR_{r1,m}||}{v} \leq t_{r2,n} - t_{r1,m} \tag{4.1}$$

$$\frac{||PR_{r2,n} - PR_{r1,m}||}{v} \leq t_{r2,n} - (t_{r1,m} + \Delta t_{recarga}) \tag{4.2}$$

Nas desigualdades 4.1 e 4.2, v é a velocidade, constante, do robô de recarga. Uma aresta é considerada factível, quando o tempo para que o robô a percorra não ultrapassa o limite que garanta o seu encontro com o robô de serviço em questão.

A condição mostrada em 4.1 deve ser utilizada quando se considera uma recarga instantânea, ou seja, em situações simuladas em que basta que um robô de recarga se aproxime de um robô de serviço para que este seja recarregado. Porém, quando se trata de uma aplicação prática, deve ser considerado um tempo ($\Delta t_{recarga}$) para que ocorra a recarga do veículo de serviço, de acordo com a desigualdade mostrada em 4.2. Isto afeta na determinação da factibilidade de uma aresta, alterando os valores dos custos a serem analisados na resolução do problema de otimização, detalhado a seguir.

- **Estrutura de Otimização**
- *Função Objetivo:*

$$\min \sum_{\delta=1}^{u} \sum_{m \in V} \sum_{n \in V} c_{mn} x_{mn\delta} \tag{4.3}$$

sujeito a:

$$\sum_{n \in V \setminus V_0} x_{0n\delta} = 1 \qquad \forall \delta \in \{1, ..., u\} \tag{4.4}$$

$$\sum_{m \in V \setminus V_0} x_{md\delta} = 1 \qquad \forall \delta \in \{1, ..., u\} \tag{4.5}$$

$$\sum_{\delta=1}^{u} \sum_{n \in V_r} \sum_{m \in V} x_{mn\delta} = 1 \qquad \forall r \in \{RS_1, ..., RSw\} \tag{4.6}$$

$$\sum_{\delta=1}^{u} \sum_{m \in V_r} \sum_{n \in V} x_{mn\delta} = 1 \qquad \forall r \in \{RS_1, ..., RSw\} \tag{4.7}$$

$$\sum_{m,n \in V} x_{mf\delta} - x_{fn\delta} = 0 \qquad \forall \delta \in \{1, ..., u\} \quad \forall f \in V \backslash V_0 \tag{4.8}$$

$$x_{mn\delta} \in \{0, 1\} \qquad \forall m, n \in V \quad \forall \delta \in \{1, ..., F\} \tag{4.9}$$

Deseja-se minimizar a função objetivo 4.3, utilizando variáveis binárias de decisão 4.9. As restrições deste problema, garantem que os robôs de recarga saiam de seus vértices iniciais (ponto de partida) com destino a um dos pontos de carregamento, retornando à um vértice artificial d após toda a operação. O que pode ser visto nas Equações 4.4 e 4.5, respectivamente. É necessário também que cada robô de serviço seja atendido uma única vez, conforme as restrições 4.6 e 4.7. Para assegurar a escolha correta das rotas percorridas por estes robôs, comparam-se os graus de entrada e de saída em todos os vértices do grafo, de acordo com a Equação 4.8.

4.2.2 **Exemplo básico de aplicação**

Para uma melhor compreensão da estrutura de otimização descrita, segue um exemplo simples, o qual é composto por dois robôs de serviço e um robô de recarga. O objetivo deste exemplo é mostrar as equações de restrições de forma mais detalhada e reforçar o entendimento do método utilizado.

A Figura 34 mostra o cenário considerado neste exemplo. São dois robôs de serviço realizando suas tarefas de monitoramento em dois caminhos distintos. Cada percurso possui dois pontos destinados à operação de recarga, tarefa esta designada a um único robô (RR).

Como descrito anteriormente, a tarefa de recarga é realizada de forma que o veículo responsável pela operação percorra um caminho considerado ótimo. Neste exemplo, os custos atribuídos às arestas foram gerados aleatoriamente, com o intuito de demonstrar a técnica. Na prática, os custos podem ser associados ao tempo, conforme mencionado, ao consumo de combustível ou à distância percorrida pelo robô de recarga. A Tabela 2 mostra os custos de cada aresta. O vértice v_0 se refere ao ponto inicial do veículo responsável pela operação de recarga.

A estrutura de otimização é detalhada a seguir de acordo com a simbologia adotada na subseção anterior:

- **Função Objetivo:**

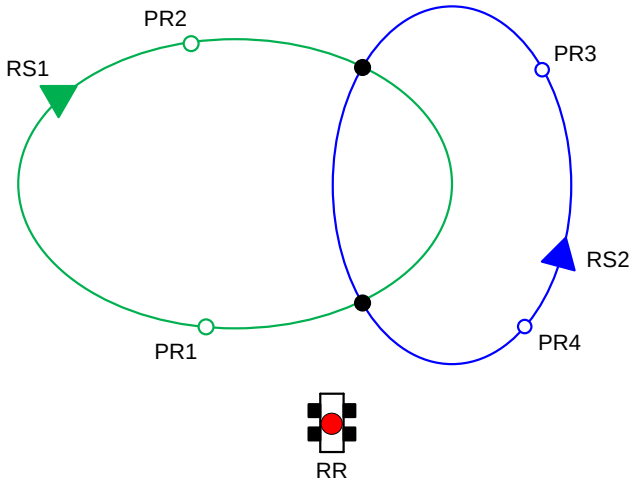


Figura 34 – Formulação MILP: exemplo didático

Tabela 2 – Custos para as arestas

ARESTA	CUSTO
$v_0 - PR_1$	2
$v_0 - PR_2$	4
$v_0 - PR_3$	3
$v_0 - PR_4$	3
$PR_1 - PR_3$	4
$PR_1 - PR_4$	8
$PR_2 - PR_3$	8
$PR_2 - PR_4$	10

$$
\begin{aligned}
\min \quad & c_{111}x_{111} + c_{121}x_{121} + c_{211}x_{211} + c_{221}x_{221} && m \in V_{r1} \\
& + c_{111}x_{111} + c_{121}x_{121} + c_{211}x_{211} + c_{221}x_{221} && m \in V_{r2} \\
& + c_{011}x_{011} + c_{021}x_{021} && n \in V_{r1} \\
& + c_{011}x_{011} + c_{021}x_{021} && n \in V_{r2} \\
& + c_{1d1}x_{1d1} + c_{2d1}x_{2d1} && m \in V_{r1} \\
& + c_{1d1}x_{1d1} + c_{2d1}x_{2d1} && m \in V_{r2}
\end{aligned} \tag{4.10}
$$

A equação é escrita de forma que, ao final de cada linha, especifica-se o conjunto ao qual pertencem os índices (m e n) de cada vértice do grafo. Esta é uma representação bem coerente e será utilizada em todas as equações, visto que os conjuntos de vértices particionados são mutuamente exclusivos. Desta forma, m e n não podem pertencer a um mesmo ponto de recarga.

- **Restrições (1):**

$$x_{mn\delta} \in \{0,1\} \qquad (4.11)$$

$$\begin{aligned} x_{011} + x_{021} \qquad & n \in V_{r1} \\ +x_{011} + x_{021} = 1 \qquad & n \in V_{r2} \end{aligned} \qquad (4.12)$$

$$\begin{aligned} x_{1d1} + x_{2d1} \qquad & m \in V_{r1} \\ +x_{1d1} + x_{2d1} = 1 \qquad & m \in V_{r2} \end{aligned} \qquad (4.13)$$

A restrição 4.11 define $x_{mn\delta}$ como uma variável binária.

As restrições 4.12 e 4.13 garantem que o robô de recarga parta do seu ponto inicial v_0 e retorne a um vértice artificial v_d, respectivamente. Nas aplicações práticas, v_d pode ser o próprio vértice inicial. A Figura 35 (a e c) mostra as possibilidades de escolha para o robô de recarga no que se refere às restrições 4.12 e 4.13, respectivamente. Na Figura 35 (b e d) tem-se as respectivas arestas determinadas pelo algoritmo.

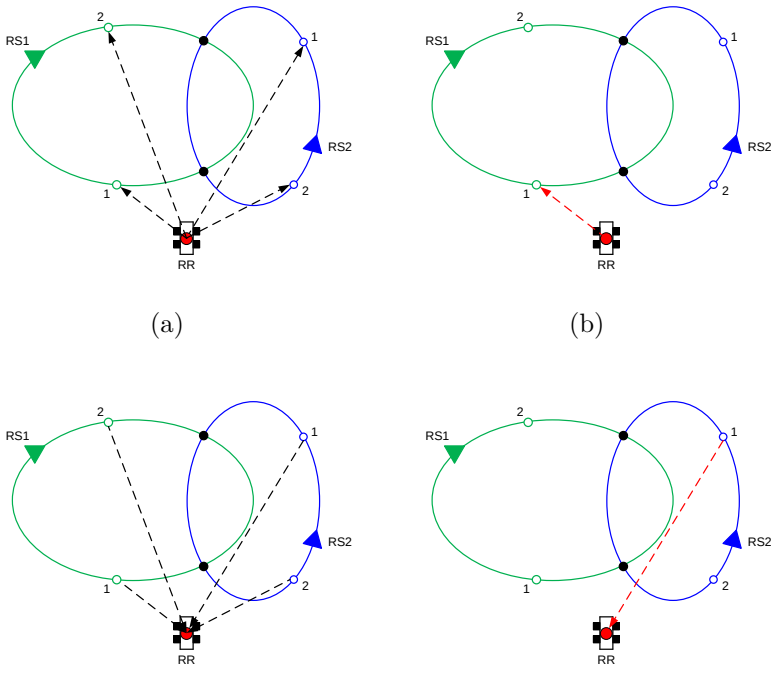


Figura 35 – Exemplo Didático: restrições (1)

- **Restrições (2):**

$$x_{011} + x_{021} + x_{111} + x_{121} + x_{211} + x_{221} = 1 \qquad n \in V_{r1}; m \in V \backslash V_{r1} \qquad (4.14)$$

$$x_{011} + x_{021} + x_{111} + x_{121} + x_{211} + x_{221} = 1 \qquad n \in V_{r2}; m \in V \backslash V_{r2} \qquad (4.15)$$

As restrições 4.14 e 4.15 forçam o robô de recarga a visitar apenas um dos vértices pertencentes a cada caminho. A Equação 4.14, por exemplo, garante que o veículo se dirige a um único vértice de V_{r1} durante a sua tarefa de recarga.

Estas restrições também asseguram que um robô de recarga não pode sair de um vértice, após recarregar o robô de serviço em questão, e visitar outro vértice (ponto de recarga) pertencente ao mesmo caminho.

Estas situações podem ser vistas na Figura 36.

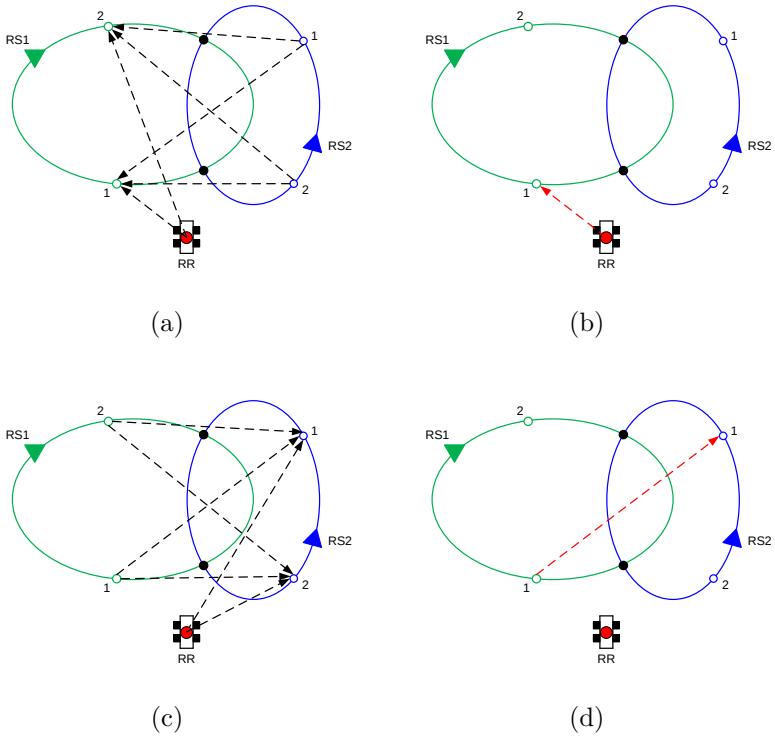


Figura 36 – Exemplo Didático: restrições (2), Equações 4.14 e 4.15

$$x_{101} + x_{201} + x_{111} + x_{121} + x_{211} + x_{221} = 1 \qquad m \in V_{r1}; n \in V \backslash V_{r1} \qquad (4.16)$$

$$x_{101} + x_{201} + x_{111} + x_{121} + x_{211} + x_{221} = 1 \qquad m \in V_{r2}; n \in V \backslash V_{r2} \qquad (4.17)$$

As Equações 4.16 e 4.17 complementam a ideia anterior. Porém, nestas restrições, está imposto ao robô de recarga partir de apenas um dos vértices (PR) pertencentes a cada caminho. A Figura 37 ilustra esta condição.

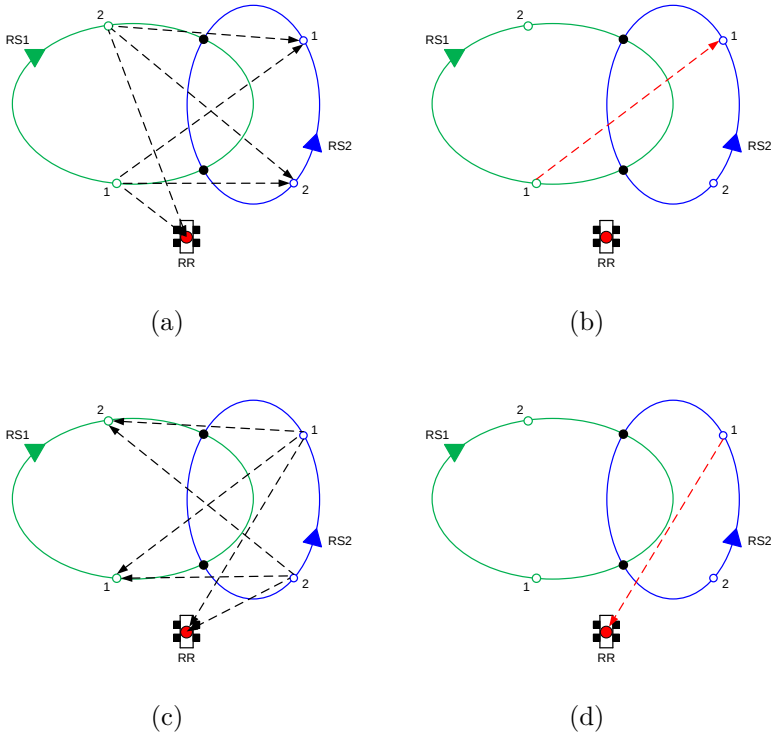


Figura 37 – Exemplo Didático: restrições (2), Equações 4.16 e 4.17

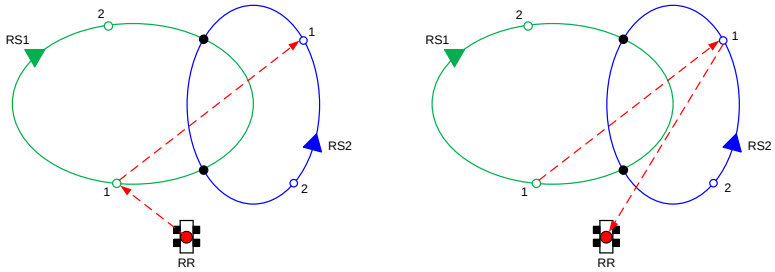
- **Restrições (3):**

$$(x_{011} - x_{111}) + (x_{021} - x_{211}) + (x_{011} - x_{121}) + (x_{021} - x_{221}) = 0$$
$$m \in V_0; f \in V_{r1}; n \in V_{r2} \tag{4.18}$$
$$(x_{111} - x_{101}) + (x_{121} - x_{201}) + (x_{211} - x_{101}) + (x_{221} - x_{201}) = 0$$
$$n \in V_0; f \in V_{r2}; m \in V_{r1} \tag{4.19}$$

O conjunto de restrições (3) consiste de equações que comparam os graus de entrada e de saída em cada vértice do grafo. Durante o caminho ótimo percorrido pelo robô de recarga ao longo de sua missão, as variáveis binárias ligadas às arestas que compõem este trajeto, recebem valor lógico 1. Nas Equações 4.18 e 4.19, o vértice f corresponde ao ponto de comparação dos graus (valores lógicos 0 ou 1). Quando um robô de recarga chega a um vértice f e parte deste para outro ponto de recarga no grafo, a subtração das variáveis binárias ligadas a estas duas arestas deve ser nula. Isto se torna bem claro, já que os graus (valores lógicos) de entrada e de saída em f são iguais a 1, neste caso.

A Figura 38 (a e b) refere-se às restrições 4.18 e 4.19 respectivamente.

- **Solução:**



(a) (b)

Figura 38 – Exemplo Didático: restrições (3)

De acordo com os custos arbitrados a cada aresta, as seguintes variáveis binárias recebem valor lógico igual a 1:

$$\begin{aligned} x_{011} \quad & n \in V_{r1} \\ x_{111} \quad & n \in V_{r2} \\ x_{101} \quad & m \in V_{r2} \end{aligned}$$

Isto corresponde as arestas das linhas 1, 3 e 5 da Tabela 2, cujos custos são de, respectivamente, 2, 3 e 4. A Figura 39 ilustra qual seria o caminho escolhido pelo robô de recarga.

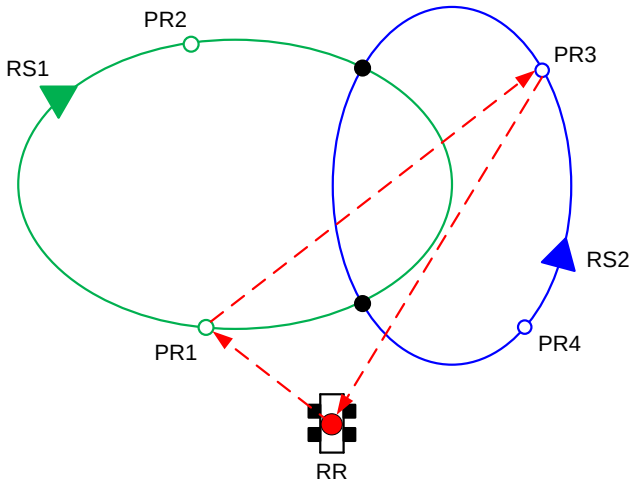
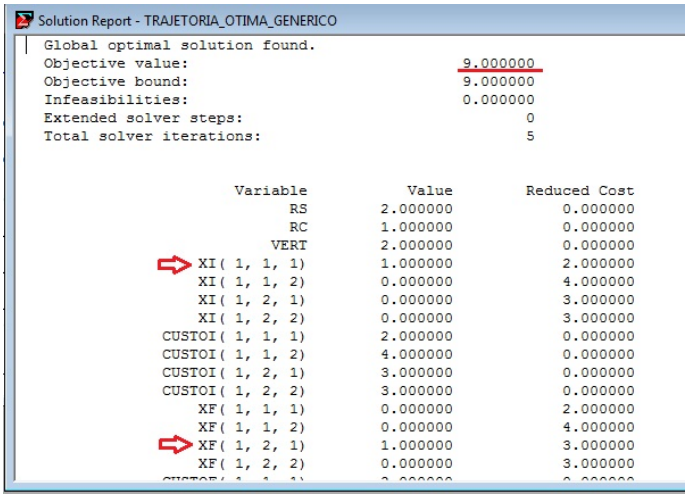


Figura 39 – Formulação MILP: solução do exemplo didático

Todas as outras variáveis binárias recebem valor lógico zero, isto faz com que o custo mínimo para a missão global do veículo de recarga seja de:

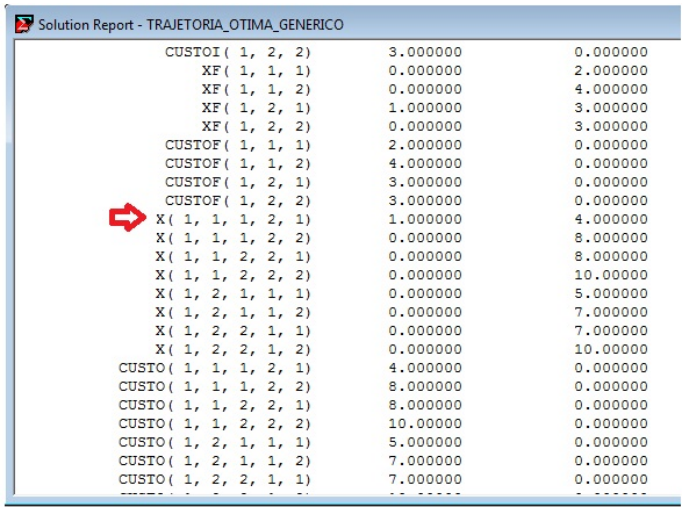
$$FOB = 2 + 4 + 3 = 9$$

O *software* de otimização LINGO fornece relatórios com o detalhamento de todas as equações de restrição, com os resultados das variáveis envolvidas e com o valor da função objetivo (FOB). As Figuras 40, 41 e 42 ilustram a situação para este caso exemplo.



Fonte: *print screen* do *software* de otimização LINGO

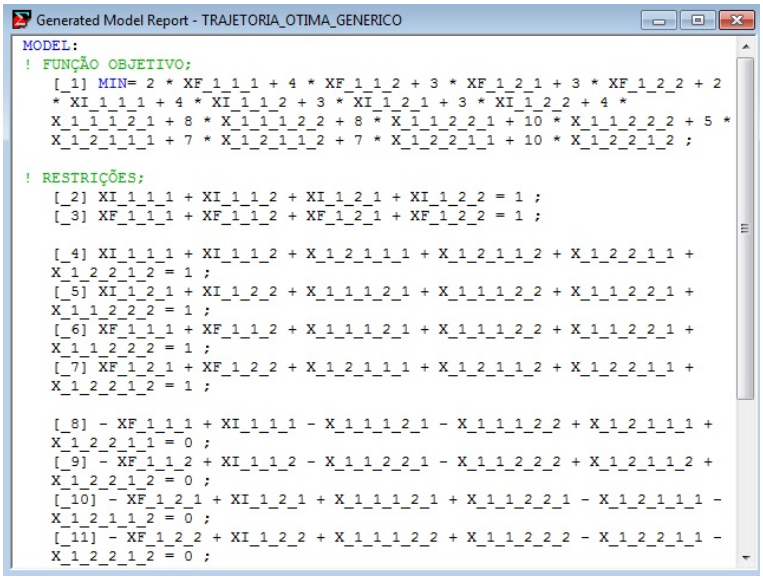
Figura 40 – Resultados: função objetivo e variáveis de decisão



Fonte: *print screen* do *software* de otimização LINGO

Figura 41 – Resultados: variáveis de decisão

O valor da função objetivo está destacado na Figura 40, juntamente com as variáveis binárias associadas as arestas que ligam v_0 a PR_1 e PR_3 a v_0. Na Figura 41, tem-se a



Fonte: *print screen* do *software* de otimização LINGO

Figura 42 – Função Objetivo e Equações de Restrições

complementação com a aresta que liga PR_1 a PR_3. Na Figura 42 tem-se um relatório com todas as equações de restrições para este problema.

4.3 O ALGORITMO DE OTIMIZAÇÂO

De uma forma geral, o processo de escolha para os melhores caminhos a serem seguidos pelos robôs de recarga pode ser visto na Figura 43.

Como entradas de dados manuais, têm-se: número de robôs de serviço, número de robôs de recarga, número de pontos de recarga, instantes de tempo em que os robôs de serviço passam pelos pontos destinados a uma possível recarga, comprimento de todas as arestas e as velocidades dos robôs de recarga.

O processamento realizado para se obterem os custos, nesta etapa, depende do que é almejado como objetivo na aplicação. Durante a apresentação da formulação matemática do programa linear inteiro misto, foi exposta uma condição de factibilidade para cada aresta pertencente ao grafo representativo. Os tempos a serem introduzidos no algoritmo são gerados, como já mencionado, durante a primeira etapa de otimização, a qual coordena os movimentos dos robôs de serviço através do controle de suas velocidades.

Devido ao fato de que as tarefas dos robôs de serviço, em seus respectivos caminhos, são realizadas ciclicamente, existe a necessidade de se incluir como entrada de dados o número de ciclos a serem executados por estes veículos. Desta maneira, os conjuntos de vértices que compõe os caminhos de cada veículo de serviço, são constituídos pelos pontos de recarga multiplicados pelo número de ciclos. O controle das velocidades dos robôs de

Figura 43 – Fluxograma: otimização para os robôs de recarga

serviço é definido para um ciclo, sendo repetido para os demais. A proposta é fazer com que o número de ciclos seja o maior possível, uma tarefa com horizonte de planejamento "infinito".

Neste trabalho, foi utilizado o *software* MATLAB® para a realização do cálculo dos custos, bem como a determinação das factibilidades das arestas. Quando, nas Inequações 4.1 e 4.2, os tempos limites são violados, atribuem-se altos valores para os pesos das respectivas arestas. O mesmo ocorre quando são encontrados valores de pesos negativos.

Em aplicações práticas, deve-se ter uma boa estimativa em relação ao consumo de energia por ciclo, para cada robô de serviço. Isto implica na determinação do momento exato em que os robôs de recarga devem partir de seus pontos iniciais. É necessário que todos os robôs de serviço consigam atingir os seus vértices, sem o risco de pararem por falta de bateria, durante a missão designada ao grupo robótico de recarga.

Após a determinação de todos os custos, é resolvido o problema de otimização. O otimizador, ao detectar todo o conjunto de possíveis caminhos, analisa as equações de restrições. Respeitadas todas as condições impostas, é realizado o cálculo do valor da função objetivo. A modelagem da função objetivo e das equações de restrições, pode ser vista a seguir.

- **Função Objetivo:**

$$\min \sum_{\delta=1}^{u} \sum_{m \in V} \sum_{n \in V} c_{mn} x_{mn\delta} \tag{4.20}$$

Modelagem utilizada no LINGO:

```
MIN = @SUM(TABELA(C,Sp,Vp,Sd,Vd):  CUSTO(C,Sp,Vp,Sd,Vd)*X(C,Sp,Vp,Sd,Vd))
        + @SUM(TABELA_ZERO(C,Sd,V): CUSTOI(C,Sd,V)*XI(C,Sd,V) +
         @SUM(TABELA_FINAL(C,Sp,V): CUSTOF(C,Sp,V)*XF(C,Sp,V));
```

Os conjuntos derivados já foram mencionados no Capítulo 2. Os parâmetros pertencentes a cada um dos conjuntos são interpretados da seguinte maneira:

O parâmetro representado pela letra C pertence ao conjunto primitivo *ROBOS_CARGA*, variando de 1 até o número de robôs de recarga na aplicação. O conjunto de robôs de serviço está aqui representado pela letra S. Como a análise é feita sempre aos pares de vértices, diferenciam-se os robôs de partida e de destino utilizando p e d, respectivamente. Portanto, um robô de recarga sempre parte de um robô de serviço (Sp) em direção a outro (Sd). A análise dos vértices do grafo é feita de forma análoga, sendo Vp o vértice de partida e Vd o vértice de destino.

O conjunto *TABELA* não leva em consideração, nesta implementação, os conjuntos de vértices iniciais e finais, que correspondem aos pontos iniciais e de destinos finais para os robôs de recarga. Desta forma, foram criados os conjuntos, também derivados, *TABELA_ZERO* e *TABELA_FINAL*, nos quais os termos *ZERO* e *FINAL* designam os pontos iniciais e finais do problema.

Os atributos CUSTO, CUSTOI e CUSTOF, junto com os parâmetros já discutidos, referenciam todos os custos previamente calculados, respeitadas as condições de factibilidades das arestas.

As variáveis de decisão denotadas por X, XI e XF, atribuem às arestas os níveis lógicos "0"ou "1"de forma que os conjuntos das arestas que constituem os caminhos ótimos recebem o valor lógico "1". Na simbologia matemática, significa dizer que x_{mn} recebe valor lógico "1"quando o robô de recarga δ visita o vértice n após visitar o vértice m.

- **Restrições:**

A declaração das variáveis binárias de decisão, neste trabalho, é feita da seguinte forma:

```
@FOR(TABELA(C,Sp,Vp,Sd,Vd):  @BIN(X(C,Sp,Vp,Sd,Vd)));
@FOR(TABELA_ZERO(C,Sd,V): @BIN(X(C,Sd,V)));
@FOR(TABELA_FINAL(C,Sp,V): @BIN(X(C,Sp,V)));
```

Desta maneira, são garantidas todas as possíveis combinações de arestas, atribuindo uma variável de decisão a cada uma destas arestas.

As restrições que garantem aos robôs de recarga partirem de e retornarem aos seus pontos iniciais são descritas, matematicamente, por:

$$\sum_{n \in V \setminus V_0} x_{0n\delta} = 1 \qquad \forall \delta \in \{1, ..., u\} \tag{4.21}$$

$$\sum_{m \in V \setminus V_0} x_{md\delta} = 1 \qquad \forall \delta \in \{1, ..., u\} \tag{4.22}$$

Suas respectivas modelagens em LINGO:

```
@FOR(ROBOS_CARGA(C):
 @SUM(TABELA_ZERO(CC,Sd,V) | CC#EQ#C: XI(CC,Sd,V)) = 1;
);

@FOR(ROBOS_CARGA(C):
 @SUM(TABELA_FINAL(CC,Sp,V) | CC#EQ#C: XF(CC,Sp,V)) = 1;
);
```

Nota-se o uso do operador lógico #EQ#, o qual retorna *TRUE* quando os operandos CC e C são iguais.

Para garantir que os robôs de recarga cheguem a apenas um dos pontos, destinados a esta finalidade, pertencentes aos caminhos de cada veículo de serviço, tem-se matematicamente:

$$\sum_{\delta=1}^{u} \sum_{n \in V_r} \sum_{m \in V} x_{mn\delta} = 1 \qquad \forall r \in \{RS_1, ..., RSw\} \tag{4.23}$$

Código utilizado:

```
@FOR(ROBOS_SERV(Sd):
 @SUM(TABELA(C,Sp,Vp,Sdd,Vd) | Sdd#EQ#Sd:  X(C,Sp,Vp,Sdd,Vd)) +
 @SUM(TABELA_ZERO(C,Sdd,V) | Sdd#EQ#Sd:  XI(C,Sdd,V)) = 1;
);
```

Analogamente, a restrição que define a partida dos robôs de recarga de apenas um dos pontos pertencentes aos caminhos é modelada da seguinte forma:

$$\sum_{\delta=1}^{u} \sum_{m \in V_r} \sum_{n \in V} x_{mn\delta} = 1 \qquad \forall r \in \{RS_1, ..., RSw\} \tag{4.24}$$

```
@FOR(ROBOS_SERV(Sp):
 @SUM(TABELA(C,Spp,Vp,Sd,Vd) | Spp#EQ#Sp:  X(C,Spp,Vp,Sd,Vd)) +
 @SUM(TABELA_FINAL(C,Spp,V) | Spp#EQ#Sp:  XF(C,Spp,V)) = 1;
);
```

O último conjunto de restrições, que compara os graus de entrada e de saída em cada vértice, pode ser obtido como segue:

Matematicamente:

$$\sum_{m,n \in V} x_{mf\delta} - x_{fn\delta} = 0 \qquad \forall \delta \in \{1, ..., u\}$$
$$\forall f \in V \backslash V_0 \tag{4.25}$$

Código LINGO:

```
@FOR(ROBOS_CARGA(C):
 @FOR(ROBOS_SERV(ROBSERV):
  @FOR(VERTICES(V):
   -@SUM(TABELA(CC,Sp,Vp,Sd,Vd)|Vp#EQ#V #AND# Sp#EQ#ROBSERV
   #AND# CC#EQ#C: X(CC,Sp,Vp,Sd,Vd))
   +@SUM(TABELA(CC,Sp,Vp,Sd,Vd)|Vd#EQ#V #AND# Sd#EQ#ROBSERV
   #AND# CC#EQ#C: X(CC,Sp,Vp,Sd,Vd))
```

```
    + XI(C,ROBSERV,V) - XF(C,ROBSERV,V) = 0;
   );
  );
 );
```

Finalmente, é informado a todos os robôs de recarga o caminho a ser percorrido. A cooperação entre os grupos robóticos é feita de forma que cada robô de recarga auxilie um grupo de robôs de serviço. O objetivo, neste trabalho, é que esta operação seja executada no menor tempo possível. Ao chegar num ponto de recarga antes do robô de serviço, o veículo responsável por sua recarga deve esperar até que o contato ocorra. Só após a troca da bateria é que o robô de recarga deve deixar o vértice em questão.

Ao retornarem aos seus pontos iniciais, os robôs de recarga ficam no aguardo de uma nova chamada às suas operações. Configurações como o grupo robótico atendido por cada robô de recarga, podem ser mudadas devido aos cálculos dos novos custos e às novas possibilidades de caminhos ótimos. Este método de busca por grafo acíclico dirigido (DAG - *Directed Acyclic Graph*) fornece um excelente encaixe entre as duas estruturas de otimização utilizadas.

4.4 CONCLUSÕES

Este capítulo é de fundamental importância para o desenvolvimento do trabalho, pois fornece os subsídios necessários à caracterização de persistência para as tarefas de um dos grupos robóticos.

Foi destacada a importância da utilização de uma técnica, para recarregar robôs que executam uma determinada tarefa. A introdução de um novo grupo robótico (RRs) requer uma estratégia de coordenação global dos veículos. O cenário analisado passa a ser o mais completo possível.

A solução empregada no trabalho foi detalhada com a descrição do programa linear inteiro misto, seguido de um exemplo de aplicação. Todas as equações de restrições juntamente com a função objetivo, são devidamente explicadas visando a compreensão do método de busca por DAG (*Directed Acyclic Graph*).

Por fim, fez-se uma conexão entre os dois otimizadores utilizados para coordenar todos os veículos envolvidos no cenário. O algoritmo que otimiza os caminhos a serem percorridos pelos robôs de recarga foi descrito através de um fluxograma (Figura 43). A comunicação entre as duas estruturas de otimização se dá na obtenção dos instantes de tempos em que os robôs de serviço passam pelos pontos de interesse, no caso, os pontos de recarga.

A estratégia de recarga utilizada pode ser aplicada em ambientes cujos caminhos

percorridos pelos robôs de serviço são plenamente conhecidos. Os custos analisados são escolhidos de acordo com a aplicação e, no caso deste trabalho, deseja-se minimizar o tempo total da operação de recarga. Portanto, é necessária uma alternativa para que os perfis de tempos sejam disponibilizados a este processo de otimização [18].

5 RESULTADOS E DISCUSSÕES

Foram realizados dois estudos de casos (CASO 1 e CASO 2) visando o experimento das técnicas de otimização descritas. A terceira parte dos resultados aborda algumas simulações, onde estas metodologias são aplicadas em robôs simulados.

Várias situações são consideradas com um aumento gradativo no detalhamento das simulações. Nos dois primeiros casos, são considerados os resultados obtidos em cada estrutura de otimização, sem levar em consideração o controle dos robôs envolvidos [49]. Desta forma, pode-se planejar em modo *off-line*, todos os caminhos a serem seguidos pelos robôs de recarga, ou seja, futuramente é informado aos robôs os seus caminhos ótimos. Este planejamento é importante, visto que todos os caminhos dos robôs de serviço são conhecidos. Além disso, os valores de $\lambda 1$ e $\lambda 2$ são escolhidos de forma empírica nos casos analisados. O último caso descreve o comportamento dos múltiplos robôs, em simulações realizadas no Stage/ROS.

Em todos os casos apresentados, considera-se que as recargas dos robôs de serviço são instantâneas. Isto significa que basta os robôs de recarga se aproximarem dos robôs de serviço para que estes sejam recarregados, simulando um contato real. A consideração de um tempo para recarga, $\Delta t_{recarga}$, implica em uma mudança apenas na maneira de efetuar o cálculo dos custos a serem atribuídos às arestas do grafo representativo em cada situação, conforme descrito na Seção 4.2.

5.1 CASO 1: ESTRUTURAS DE OTIMIZAÇÃO ($\lambda 1 = 3$ E $\lambda 2 = 1$)

Neste primeiro caso, são considerados dois robôs de serviço na realização do monitoramento, sendo auxiliados por um robô de recarga. São escolhidos diferentes valores de λ para cada um dos veículos, de forma que estes não completem os seus ciclos em um mesmo tempo.

A velocidade do robô de recarga é constante e considerada suficiente para que todas as arestas (caminhos) sejam factíveis. Desta forma, os comprimentos destes caminhos não são considerados na análise dos custos durante o último processo de otimização. A Figura 44 ilustra esta primeira situação.

O comprimento de um caminho é 3 vezes maior do que o outro. Considera-se então $\lambda 1 = 3$, fazendo com que RS_1 complete um ciclo em 90 segundos, e $\lambda 2 = 1$, com RS_2 levando um tempo de 30 segundos por ciclo. O tempo base é $T = 30$ segundos. A Tabela 3 revela os perfis de velocidade média dos robôs de serviço e os tempos em cada ponto dos caminhos, obtidos pela resolução do problema de possíveis colisões.

O custo total de recarga neste cenário é de 30 segundos. O caminho ótimo percorrido por RR segue as arestas B, E e A, nesta ordem, com RS_2 sendo recarregado no ponto

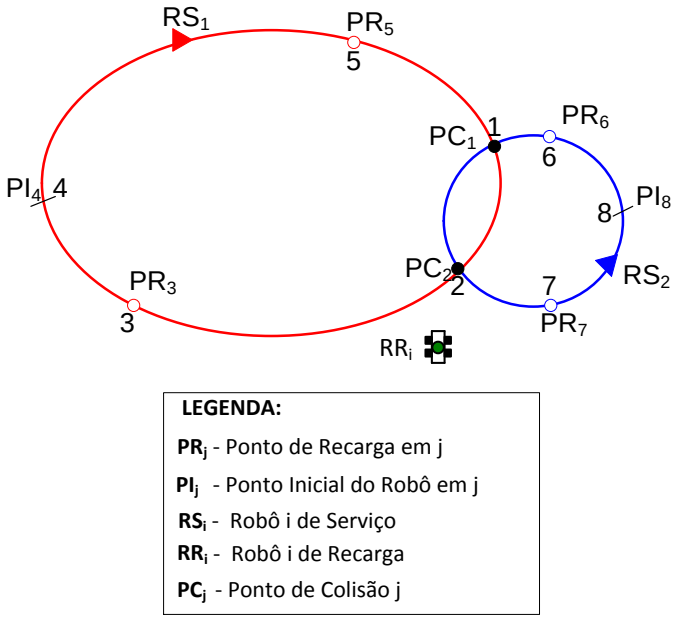


Figura 44 – Ambiente de Trabalho para a Missão Abordada no Caso 1

Tabela 3 – Tempo e velocidade média dos robôs de serviço (CASO 1)

PONTO	**CICLO**	**TEMPO (s)**		**VELOCIDADE (m/s)**	
		RS_1	RS_2	RS_1	RS_2
1	1	28	13	1	0,46
	2	118	43	1	0,46
2	1	34	19	1	0,63
	2	124	49	1	0,63
3	1	84	-	0,64	-
	2	174	-	0,64	-
4	1	0	-	0	-
	2	90	-		-
5	1	20	-	1	-
	2	110	-	1	-
6	1	-	3	-	1
	2	-	33	-	1
7	1	-	26	-	0,62
	2	-	56	-	0,62
8	1	-	0	-	0
	2	-	30	-	1

PR_6 e RS_1 recarregado em PR_5, no primeiro ciclo de cada robô, conforme a Figura 45.

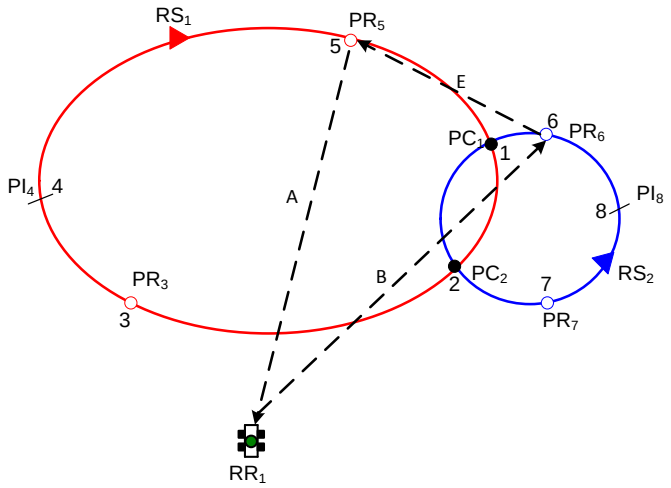


Figura 45 – Rota a ser seguida pelo Robô de Recarga para o Caso 1

5.2 CASO 2: ESTRUTURAS DE OTIMIZAÇÃO ($\lambda 1 = \lambda 2 = 1$)

Nesse exemplo, a distância por ciclo percorrida por RS_1 é de 16 metros, enquanto RS_2 tem uma excursão de 14 metros por ciclo. São utilizados $\lambda_1 = \lambda_2 = 1$, significando que o tempo para que cada veículo de serviço complete uma volta em torno de seus devidos caminhos, é igual. Foi escolhido um tempo de 30 segundos por ciclo. A Figura 46 ilustra o ambiente no qual o Caso 2 é analisado.

A Tabela 4 fornece os valores das velocidades médias (m/s) em todos os subcaminhos percorridos pelos robôs de serviço e os tempos gerados pelo algoritmo na primeira etapa de otimização, onde os pontos 1 e 2 merecem destaque por se tratarem de pontos de possíveis colisões. Para facilitar a compreensão, os pontos em questão estão devidamente numerados na Figura 46. Nota-se que existem pontos que pertencem a somente um dos caminhos. Portanto neste caso não há tempo e velocidades associados a eles.

Percebe-se que não há risco de colisão entre os robôs de serviço em nenhum dos dois pontos de colisão, uma vez que foi obtido um Δt de 15 segundos. Enquanto, por exemplo, RS_2 passa pelo ponto 1 em 4 segundos, RS_1 leva 19 segundos para alcançá-lo, com ambos os robôs ainda em seus primeiros ciclos. Os robôs passam pelos mesmos locais no ciclo posterior após 30 segundos, respeitando, desta forma, o período T estipulado no algoritmo para que o robô de serviço complete uma volta.

Pela Tabela 4 é possível avaliar através dos tempos que o robô de serviço RS_1 inicia sua tarefa no ponto 4, passando posteriormente pelos pontos 5, 1, 2 e 3 em sequência. Os valores dos tempos nos pontos 3, 5, 6 e 8 contribuem para a geração dos custos da função objetivo na próxima etapa de otimização.

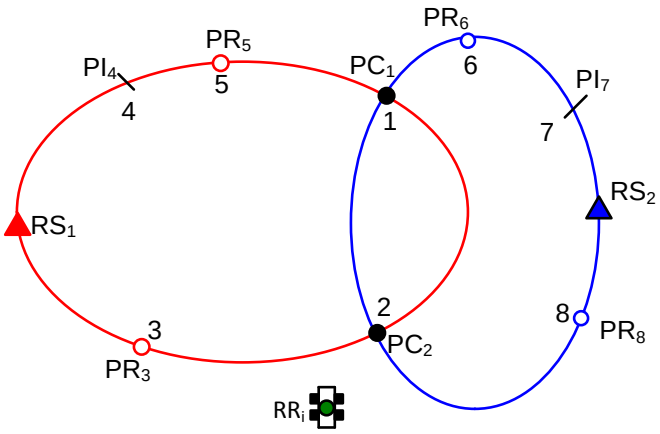


Figura 46 – Ambiente de Trabalho para a Missão Abordada no Caso 2

Tabela 4 – Tempo e velocidade média dos robôs de serviço (CASO 2)

PONTO	CICLO	TEMPO (s)		VELOCIDADE (m/s)	
		RS_1	RS_2	RS_1	RS_2
1	1	19	4	0,26	1
	2	49	34	0,26	1
2	1	24	9	0,42	0,72
	2	54	39	0,42	0,72
3	1	27	-	0,48	-
	2	57	-	0,48	-
4	1	0	-	0	-
	2	30	-	1	-
5	1	2	-	1	-
	2	32	-	1	-
6	1	-	1,5	-	1
	2	-	31,5	-	1
7	1	-	0	-	0
	2	-	30	-	1
8	1	-	25,5	-	0,37
	2	-	55,5	-	0,37

Como descrito na seção anterior, verifica-se a factibilidade das arestas, considerando o robô de recarga com uma velocidade constante de 1,3 m/s. A Tabela 5 fornece as

distâncias necessárias para esta análise.

Tabela 5 – Comprimento das arestas (CASO 2)

ARESTA	COMPRIMENTO (m)
$v_0 - PR_5$	3
$v_0 - PR_6$	3,5
$v_0 - PR_3$	1,2
$v_0 - PR_8$	1,3
$PR_5 - PR_6$	2,7
$PR_5 - PR_8$	5
$PR_3 - PR_8$	2,5
$PR_3 - PR_6$	5,5

O caminho ótimo escolhido pelo robô de recarga é mostrado na Figura 47 e abrange as arestas D, G e C, sendo que os robôs RS_2 e RS_1 são recarregados nos pontos PR_8 e PR_3, respectivamente, nesta mesma ordem e ambos ainda em seus primeiros ciclos. O custo total de recarga é de 28,92 segundos.

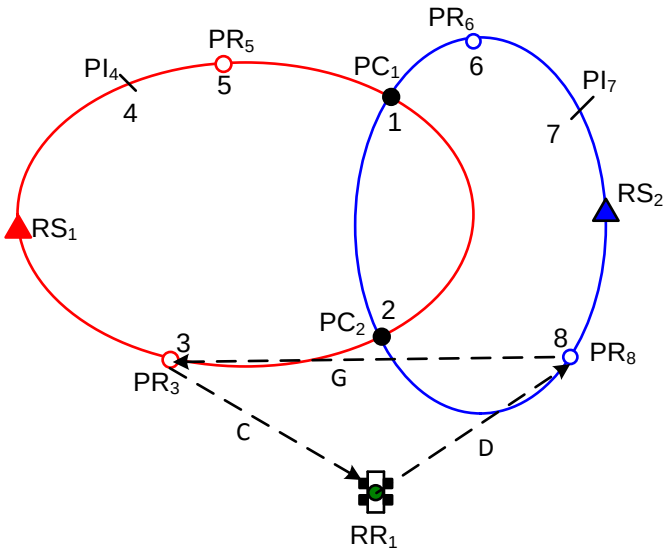


Figura 47 – Rota a ser seguida pelo Robô de Recarga para o Caso 2

5.3 SIMULAÇÕES: STAGE/ROS

Após a apresentação dos dois casos acima, nos quais foram avaliados os algoritmos de otimização aplicados no trabalho, são realizadas duas simulações com os múltiplos robôs coordenados.

Uma ilustração do cenário analisado pode ser vista na Figura 48. São utilizados dois robôs de serviço e um robô de recarga. Cada caminho é composto por dois pontos de recarga,

sendo que os pontos PR_1 e PR_2 correspondem aos pontos de recarga 1 e 2, respectivamente, do robô de serviço 1. Os pontos PR_3 e PR_4 correspondem, respectivamente, aos pontos de recarga 1 e 2 do robô de serviço 2. Além disso, os caminhos possuem pontos de colisões. Os comprimentos de todas as arestas do grafo representativo são dados na Tabela 6.

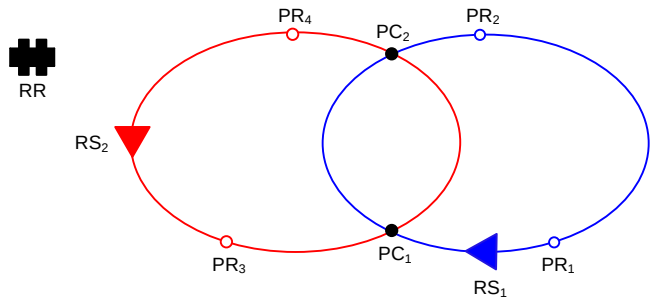


Figura 48 – Ilustração do Cenário Utilizado nas Simulações

Tabela 6 – Comprimento das arestas (Stage/ROS)

ARESTA	**COMPRIMENTO (m)**
$v_0 - PR_1$	7,46
$v_0 - PR_2$	6,02
$v_0 - PR_3$	3,82
$v_0 - PR_4$	5,00
$PR_1 - PR_3$	3,80
$PR_1 - PR_4$	3,05
$PR_2 - PR_3$	3,20
$PR_2 - PR_4$	1,00

Os cálculos destes comprimentos são realizados utilizando coordenadas próximas aos pontos de recarga, para que, durante as simulações, não ocorram choques entre o robô de recarga e os robôs de serviço. A Figura 49 ilustra essa situação.

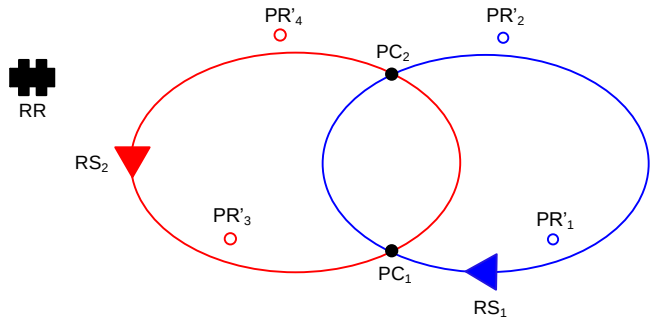


Figura 49 – Coordenadas Próximas aos Pontos de Recarga

As simulações foram realizadas no Stage/ROS, com a versão Fuerte do ROS em um sistema operacional Ubuntu 12.04. Utilizou-se apenas um computador com as seguintes configurações: processador Intel® Core™ i3, 2,10 GHz e 4,00 GB de memória RAM.

Os tempos gerados pelo primeiro programa de otimização são dados na Tabela 7. Estes valores correspondem aos momentos exatos em que os robôs de serviço passam por seus respectivos pontos de recarga. O intervalo de tempo maximizado (Δt) foi de 12,5 segundos.

Tabela 7 – Tempos Utilizados na Obtenção dos Custos para o Segundo Otimizador

PONTO	**INSTANTES DE TEMPO (s)**
PR_1	10,4
PR_2	26
PR_3	25,4
PR_4	13,2

As velocidades médias controladas dos robôs de serviço podem ser vistas nas Tabelas 8 e 9.

Tabela 8 – Velocidades médias controladas para RS_1

TRECHOS	**VELOCIDADES** RS_1 **(m/s)**
$PR_2 - PC_1$	0,33
$PC_1 - PC_2$	0,60
$PC_2 - PR_2$	0,59

Tabela 9 – Velocidades médias controladas para RS_2

TRECHOS	**VELOCIDADES** RS_2 **(m/s)**
$PR_3 - PC_1$	0,60
$PC_1 - PC_2$	0,48
$PC_2 - PR_3$	0,27

A Figura 50 mostra os robôs utilizados nas simulações, bem como os caminhos percorridos pelos robôs de serviço. O robô (cor azul) corresponde ao RS_1 e o robô (cor vermelha) representa RS_2. O robô (cor preta) é o robô de recarga (RR).

Para a tarefa de recarga são consideradas duas situações. Como o robô de recarga possui velocidade constante, considera-se em um primeiro momento, uma velocidade de 0,6 m/s. Em um segundo momento, com RR partindo do mesmo ponto inicial que na situação anterior, é considerada uma velocidade de 1,0 m/s. O ponto de partida do robô de recarga é escolhido de forma estratégica, visando um menor risco de colisão entre este veículo e os robôs de serviço.

A malha de controle utilizada para cada robô é vista na Figura 51. A implementação do controlador foi realizada em C++.

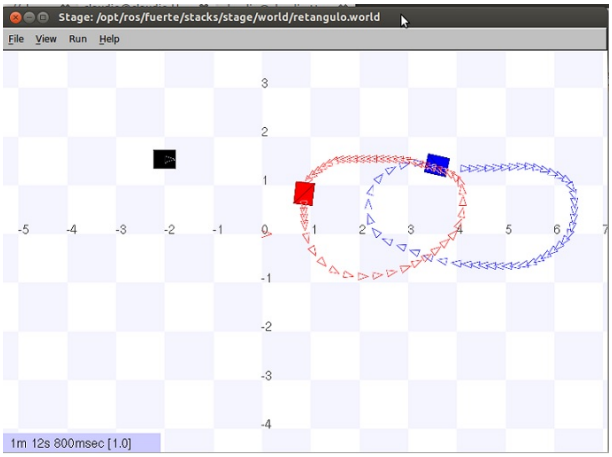
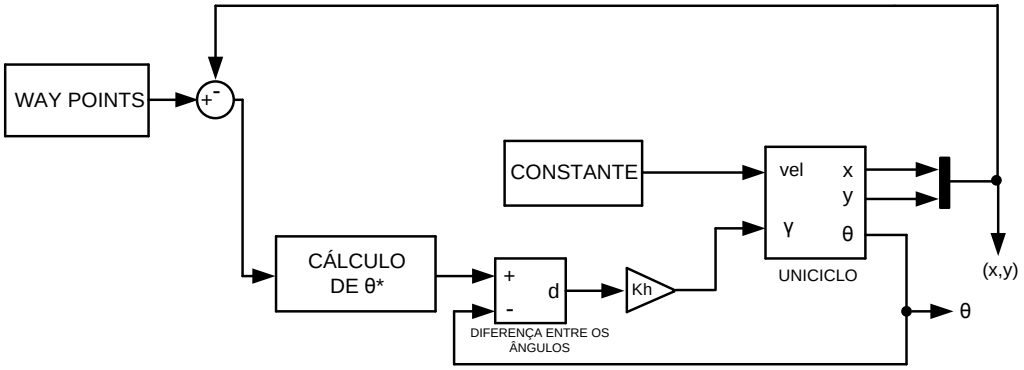


Figura 50 – Representação dos Múltiplos Robôs no Simulador



Fonte: adaptado de [40]

Figura 51 – Malha de Controle Utilizada nas Simulações

Os *way points* correspondem a uma sequência de coordenadas para que o robô percorra o caminho desejado. Como já discutido anteriormente, estes caminhos são conhecidos. As velocidades são constantes para os dois grupos robóticos. Para os robôs de serviço, os valores das velocidades em cada subcaminho correspondem aos resultados das velocidades médias obtidos através do primeiro otimizador, Tabelas 8 e 9.

O robô é orientado com um controlador proporcional. Esta orientação em direção ao ponto alvo é feita com um ganho atribuído ao resultado das diferenças angulares d $(\theta^* \ominus \theta)$. O ângulo θ^* é obtido de acordo com a Equação 5.1, a qual considera a coordenada atual e a coordenada alvo. Aplica-se, então, um ganho K_h $(K_h > 0)$ a essa diferença. O valor de K_h influencia diretamente no deslocamento angular do robô em um determinado caminho. Desta forma, são necessários ajustes em seu valor dependendo das curvas que compõe o seu caminho.

$$\theta^* = \tan^{-1} \frac{y^* - y}{x^* - x} \tag{5.1}$$

5.3.1 **Robô de recarga com velocidade constante de 0,6 m/s**

Considera-se, primeiramente, que o robô de recarga realize a sua missão com uma velocidade de 0,6 m/s. Os custos para cada aresta podem ser vistos na Tabela 10 .

Tabela 10 – Custos (segundos) atribuídos às arestas: $v = 0{,}6$ m/s

RR	v_0	R_{11}	R_{12}	R_{21}	R_{22}
v_0	X	∞	26	25,4	13,2
R_{11}	12,43	X	X	15	∞
R_{12}	10,03	X	X	∞	∞
R_{21}	6,37	∞	∞	X	X
R_{22}	8,33	∞	12,8	X	X

As notações utilizadas têm o seguinte significado:

v_0: ponto de partida de RR;
R_{ij}: robô de serviço i, ponto de recarga j;
X: inválido;

A Tabela 10 engloba todas as combinações de arestas (bidirecionais) possíveis. A sua leitura é realizada imaginando os sentidos dos deslocamentos de RR. A primeira coluna corresponde à partida de RR e a primeira linha, o seu destino.

Desta maneira, os valores denotados por "X", indicam que RR atende os robôs de serviço uma única vez durante a sua missão, como descrito nas restrições abordadas no Capítulo 4. Os valores de custos altos, representados por ∞, indicam que esta aresta é infactível durante a sua missão. O custo para que RR parta de seu vértice inicial em direção ao de recarga 2 de RS_2, por exemplo, é de 13,2 segundos. Os resultados obtidos no LINGO são representados nas Figuras 52 e 53.

As variáveis `XI(1,2,2)`, `X(1,2,2,1,2)` e `XF(1,1,2)` recebem valor lógico 1. Estes atributos correspondem aos conjuntos derivados descritos no Capítulo 2. Desta forma, o robô de recarga deve partir do vértice inicial v_0 em direção ao ponto de recarga 2 de RS_2, em seguida deslocar-se ao ponto de recarga 1 de RS_1, retornando à origem após a sua missão.

As Figuras 54, 55 e 56 mostram o caminho ótimo percorrido pelo robô de recarga. O tempo global para a tarefa de recarga, na simulação, foi de aproximadamente 37 segundos.

A pequena diferença entre o tempo determinado pelo LINGO, de 36,03 segundos, e o tempo no simulador Stage/ROS é justificada pelo fato de que, no LINGO, as arestas

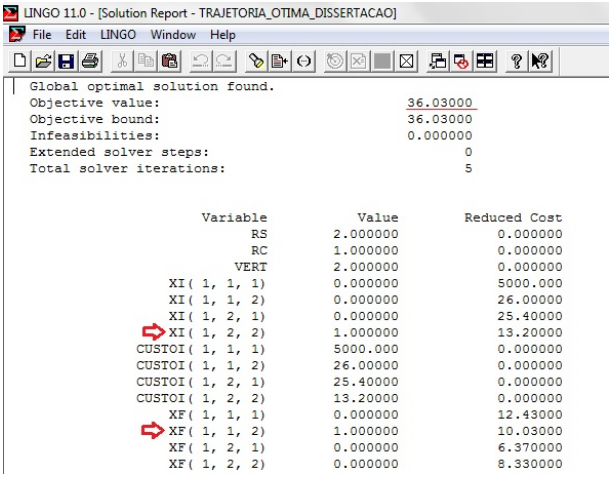


Figura 52 – Resultados LINGO (FOB e Arestas Inicial e Final): $v = 0{,}6$ m/s

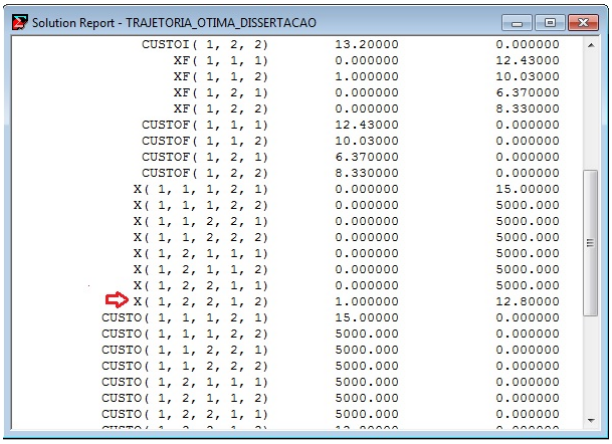


Figura 53 – Resultados LINGO (Aresta Intermediária): $v = 0{,}6$ m/s

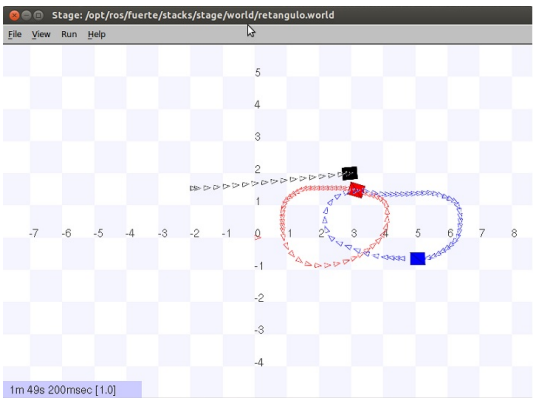


Figura 54 – Recarga de RS_2 em PR_4 ($v = 0{,}6$ m/s)

são perfeitamente retas. Porém no simulador o robô de recarga deve realizar algumas curvas durante o seu caminho. Resultados satisfatórios foram obtidos, através de testes na

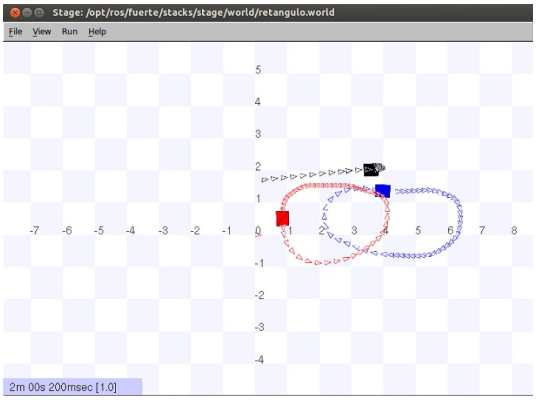


Figura 55 – Recarga de RS_1 em PR_2 (v = 0,6 m/s)

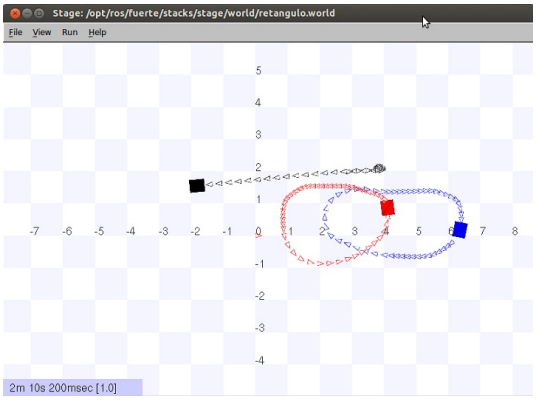


Figura 56 – Retorno de RR ao Vértice Inicial (v = 0,6 m/s)

simulação, considerando o ganho proporcional K_h igual a 1.

5.3.2 **Robô de recarga com velocidade constante de 1,0 m/s**

Nesta parte da simulação, é alterada apenas a velocidade do robô de recarga. O objetivo é testar as soluções encontradas pelo otimizador, com uma mesma configuração de controle para os robôs de serviço e com o robô de recarga partindo do mesmo vértice inicial do caso anterior.

O aumento da velocidade de RR causa uma reconfiguração nos valores dos custos, como mostra a Tabela 11

Tabela 11 – Custos (segundos) atribuídos às arestas: v = 1,0 m/s

RR	v_0	R_{11}	R_{12}	R_{21}	R_{22}
v_0	X	10,4	26	25,4	13,2
R_{11}	7,46	X	X	15	∞
R_{12}	6,02	X	X	∞	∞
R_{21}	3,82	∞	∞	X	X
R_{22}	5,00	∞	12,8	X	X

Os resultados obtidos no LINGO são apresentados nas Figuras 57 e 58.

O robô de recarga deve, agora, dirigir-se ao ponto de recarga 1 de RS_1, depois seguir em direção ao ponto de recarga 1 de RS_2 e retornar ao vértice inicial. Este caminho é representado, respectivamente, pelas variáveis de decisão `XI(1,1,1)`, `X(1,1,1,2,1)` e `XF(1,2,1)`.

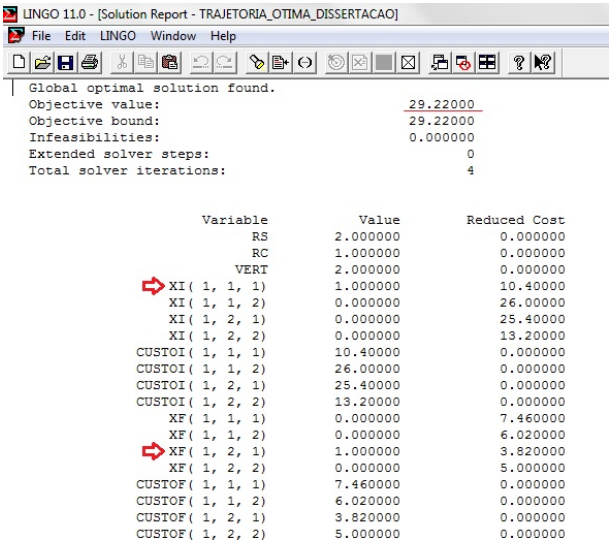


Figura 57 – Resultados LINGO (FOB e Arestas Inicial e Final): $v = 1{,}0$ m/s

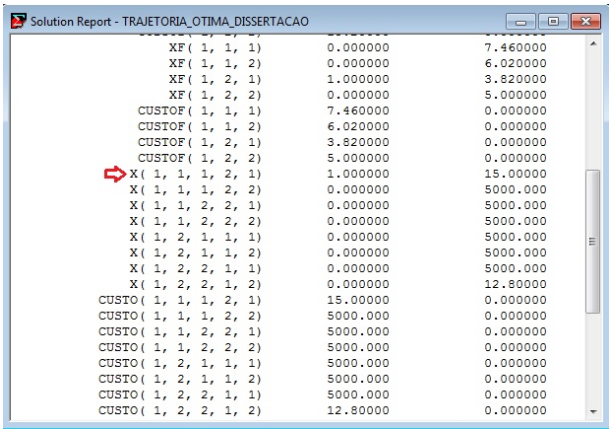


Figura 58 – Resultados LINGO (Aresta Intermediária): $v = 1{,}0$ m/s

O caminho ótimo percorrido por RR é descrito nas Figuras 59, 60 e 61. Neste caso, o tempo global para a tarefa de recarga no simulador foi de aproximadamente 31 segundos, muito próximo ao tempo determinado pelo LINGO, que foi de 29,22 segundos, de acordo com a Figura 57.

A mudança do caminho seguido por RR deve-se ao fato de que, com o aumento da sua velocidade, a aresta (v_0-PR_1) passa a ser viável, com um custo de 10,4 segundos. Desta forma, muda-se o caminho ótimo encontrado.

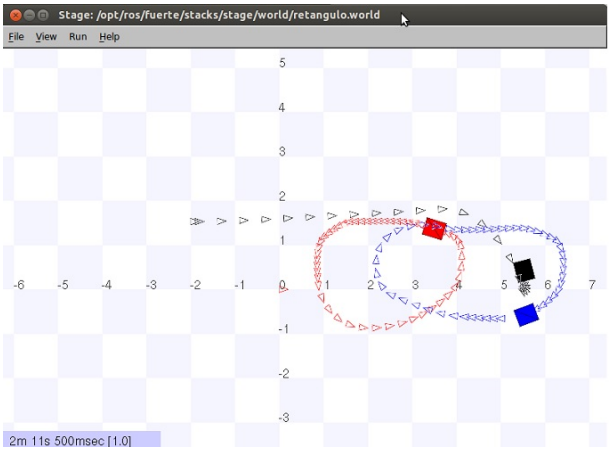


Figura 59 – Recarga de RS_1 em PR_1 (v = 1,0 m/s)

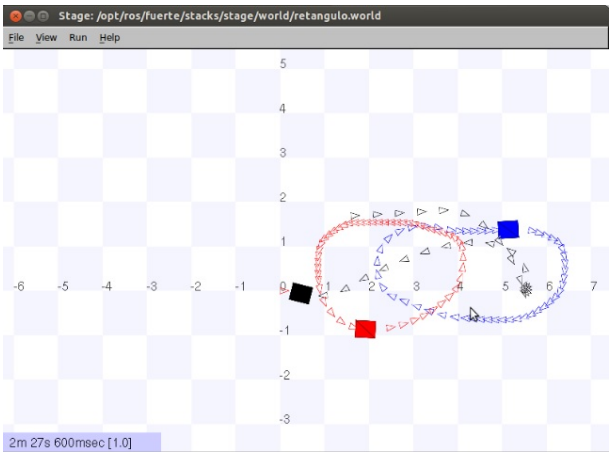


Figura 60 – Recarga de RS_2 em PR_3 (v = 1,0 m/s)

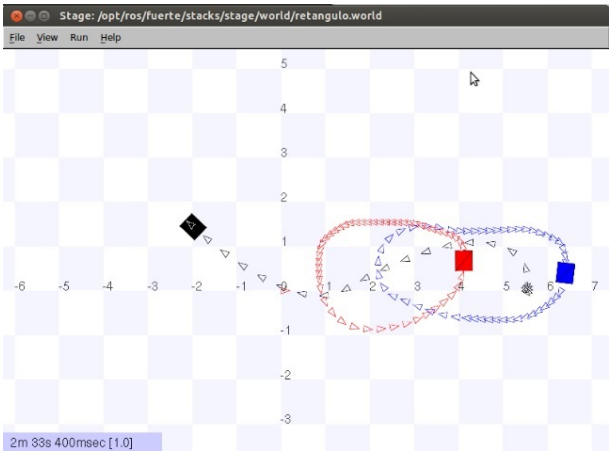


Figura 61 – Retorno de RR ao Vértice Inicial (v = 1,0 m/s)

Para fazer uma comparação com o resultado anterior, passam-se, agora, os mesmos *way points* considerados na subseção 5.3.1. Porém, RR desloca-se com a mesma velocidade

de 1 m/s. Neste caso, o tempo total para a tarefa de recarga foi de 34,33 segundos, ou seja, um tempo maior do que o gasto com RR percorrendo o caminho ótimo, de acordo com o esperado.

A Figura 62 mostra a representação em grafo, dos processos gerenciados pelo ROS durante as simulações.

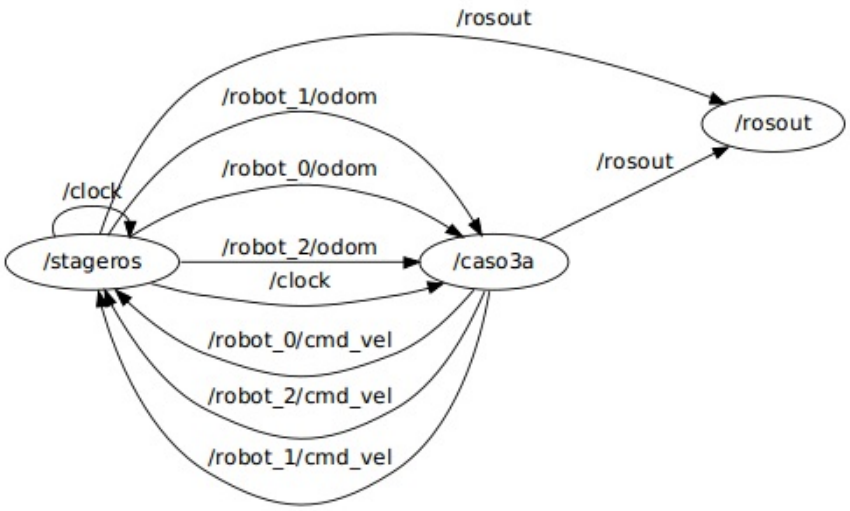


Figura 62 – Processos do ROS Gerenciados Durante as Simulações

Os vértices do grafo correspondem aos processos executados. O nó `/caso3a` contém o algoritmo de controle dos robôs. `/stageros` corresponde ao simulador utilizado e `/rosout` é responsável pelo registros dos *logs* das simulações.

Os tópicos com a formação `/robot_0/odom` contém os dados de odometria de RS_1. RS_2 e RR estão representtados por `robot_1` e `robot_2`, respectivamente.

De maneira análoga, os tópicos intitulados `/robot_0/cmd_vel` correspondem aos comandos de velocidade para os robôs.

- As animações referentes aos casos simulados podem ser acessadas em:
  https://www.dropbox.com/sh/np7lpyjsk2vft5x/AAAKoWd7A7mWb-1ljt8Gq_xja?n=93301146

  O arquivo `caso3a.mp4` refere-se ao robô de recarga com $v = 0{,}6$ m/s.
  O arquivo `caso3b.mp4` refere-se ao robô de recarga com $v = 1{,}0$ m/s.

6 CONCLUSÕES FINAIS

Este trabalho abordou uma questão de grande interesse na área robótica terrestre. Foi enfatizada a importância do crescimento das pesquisas, cujas motivações principais baseiam-se no tratamento da coordenação de múltiplos robôs autônomos. Neste escopo, discutiu-se o problema da autonomia de energia destas máquinas, que surge naturalmente quando as suas tarefas necessitam ser realizadas persistentemente. Buscou-se, durante todo o trabalho, frisar a ideia de cooperação entre robôs, utilizando como objeto de estudo os robôs autônomos terrestres.

Várias técnicas foram aplicadas no tratamento do problema da autonomia de energia dos veículos. Devido a isso, um capítulo inteiro foi dedicado aos aspectos teóricos que norteiam a aplicação abordada. Exemplos didáticos foram apresentados com o objetivo de melhorar a compreensão do trabalho. A integração do *solver* de otimização LINGO com o C++, a qual deve ser realizada cuidadosamente em aplicações *on-line*, também foi vista com detalhes.

Apresentou-se o problema tratado neste trabalho, procurando dividi-lo em duas partes principais. Deste modo, os problemas de otimização, responsáveis pela coordenação dos múltiplos robôs, foram tratados separadamente. Descreveram-se os objetivos e a importância de cada otimizador no problema global. Enfatizou-se a estrutura de otimização responsável pela tarefa de recarga, visto que esta técnica torna as tarefas realizadas pelos robôs de serviço, persistentes.

Aspectos teóricos sobre grafos foram abordados, dentre os quais discutiu-se o clássico problema do caixeiro viajante, fortemente ligado aos casos estudados no trabalho. Descreveu-se, de forma detalhada, o algoritmo que otimiza a tarefa de recarga. Neste sentido, reservou-se uma seção para a apresentação e discussão das funções do *solver* de otimização LINGO, utilizadas na implementação. Além disso, um exemplo didático foi resolvido, enfatizando o uso destas funções.

Finalmente, foram realizadas simulações, apresentando, primeiramente, os resultados dos algoritmos de otimização, apenas como um planejamento, alterando alguns parâmetros ligados aos mesmos. A Seção 5.3 abordou uma simulação completa do problema. Neste caso, os resultados dos algoritmos de otimização foram aplicados aos robôs controlados no simulador Stage/ROS.

A metodologia utilizada na tarefa de recarga não abordou a questão de possíveis colisões entre os múltiplos robôs, o que torna mais difícil coordenar os veículos em cenários maiores. Foi considerado também, que RR não precisa ser recarregado durante as operações. Mesmo em cenários menores, o fato de a técnica de recarga utilizada não garantir a ausência de colisões, aumentou o desafio durante as simulações com os robôs controlados. Na prática, deve-se posicionar o veículo de recarga em um ponto de partida

estratégico, conforme realizado na simulação, de forma que o risco de colisão durante a sua tarefa seja o menor possível. Foram obtidos resultados satisfatórios durante as simulações de coordenação global dos grupos robóticos.

O otimizador que atua no controle de alto nível dos robôs de serviço garantiu com sucesso a coordenação dos mesmos. O problema de possíveis colisões entre estes veículos foi solucionado. Em aplicações práticas, devido a erros de odometria, por exemplo, este otimizador deve ser executado diversas vezes durante as tarefas dos múltiplos robôs de serviço para que a ausência de colisões seja garantida.

6.1 SUGESTÕES PARA TRABALHOS FUTUROS

As principais sugestões para a continuação dos trabalhos nesta área de pesquisa, remetem-se ao aperfeiçoamento da tarefa de recarga. Implementar na prática o cenário considerado nas simulações realizadas neste trabalho, integrará todas as técnicas discutidas em um sistema funcionando *on-line*.

No campo teórico, pode-se planejar o caminho ótimo percorrido pelo robô de recarga, utilizando métodos para se evitar colisões. Isto poderia ser realizado, por exemplo, desenvolvendo-se um controlador de velocidade para o robô de recarga. Desta forma, simulações em grandes cenários ficariam interessantes. Além disso, pode-se desenvolver uma técnica na qual o ponto de partida do robô de recarga é escolhido de maneira ótima, minimizando os riscos de colisões.

O desenvolvimento de um único otimizador para a resolução do problema é uma ótima alternativa para futuros trabalhos. Desta maneira, o problema de possíveis colisões pode ser tratado de forma global. Pode-se considerar neste otimizador os tempos de recarga dos robôs de serviço, bem como o monitoramento da carga de suas baterias para que o robô de recarga parta à sua missão no momento exato.

Outra sugestão para trabalho futuro seria o desenvolvimento de uma técnica de recarga por indução. O trabalho teria como principal foco, o desenvolvimento de um *hardware* capaz de desempenhar esta função. Os algoritmos de otimização deverão ser modificados de forma que os robôs envolvidos no processo de recarga, tenham suas velocidades controladas próximo aos pontos de recarga.

Uma implementação prática deste trabalho também ficaria interessante, utilizando outros tipos de robôs manipuladores móveis, como o robô *KUKA*, ilustrado na Figura 14. O *framework* ROS, apresentado de forma detalhada no Capítulo 2, continua fortemente indicado como ferramenta para o gerenciamento dos processos envolvidos em todos os trabalhos aqui sugeridos.

APÊNDICE A – Software LINGO (funções básicas)

APÊNDICE A - Exemplo: modelagem de um problema de otimização em LINGO

Para exemplificar um completo problema de otimização, é apresentado um caso [42], junto com a sua modelagem e resolução.

Suponha que a companhia *Wireless Widget* (WW) possua seis armazéns que abastecem oito vendedores com seus *widgets*. Cada armazém tem uma oferta de *widgets* que não pode ser ultrapassada e cada vendedor tem uma demanda que deve ser suprida. As Tabelas 12 e 13 trazem os números exatos das capacidades dos armazéns e das demandas dos vendedores.

Tabela 12 – Capacidades dos armazéns

ARMAZÉM	CAPACIDADE
1	60
2	55
3	51
4	43
5	41
6	52

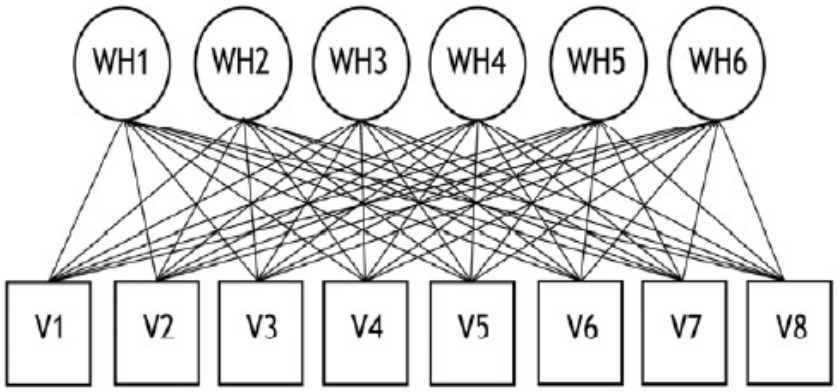
Tabela 13 – Demanda dos vendedores

VENDEDOR	DEMANDA
1	35
2	37
3	22
4	32
5	41
6	32
7	43
8	38

O objetivo deste problema é minimizar o custo total de transporte dos produtos dos armazéns a cada vendedor, de forma que todas as restrições de capacidade e de demanda sejam atendidas. Em outras palavras, deseja-se determinar quantos produtos partem de cada armazém direcionado a cada vendedor, de forma que o custo total de transporte seja mínimo.

A Figura 63 ilustra este cenário. Repare que cada armazém pode enviar a cada fornecedor, totalizando um número de 48 caminhos possíveis. Desta maneira, é necessária uma variável para cada caminho para representar a quantidade enviada.

Na Tabela 14 têm-se os custos por *widgets* ($) em cada um dos 48 caminhos.



Fonte: [42]

Figura 63 – Caminhos Possíveis para Transportes dos Armazéns aos Vendedores

Tabela 14 – Custos por *widgets* para cada caminho

	V1	V2	V3	V4	V5	V6	V7	V8
Wh1	6	2	6	7	4	2	5	9
Wh2	4	9	5	3	8	5	8	2
Wh3	5	2	1	9	7	4	3	3
Wh4	7	6	7	3	9	2	7	1
Wh5	2	3	9	5	7	2	6	5
Wh6	5	5	2	2	8	1	4	3

- **Definição dos conjuntos:**

```
SETS
    ARMAZENS: CAPACIDADE;
    VENDEDORES: DEMANDA;
    CAMINHOS( ARMAZENS, VENDEDORES): CUSTO, VOLUME;
ENDSETS
```

Temos três conjuntos para representar o problema, sendo dois primitivos e um derivado. Os atributos escritos a direita dos nomes dos conjuntos relacionam os mesmos com os seus respectivos valores, que são descritos na seção de dados.

- **Entrada de dados:**

Os dados de entrada podem ser digitados da seguinte forma:

```
DATA
    ARMAZENS = WH1 WH2 WH3 WH4 WH5 WH6;
```

```
VENDEDORES = V1 V2 V3 V4 V5 V6 V7 V8;
CAPACIDADE = 60 55 51 43 41 52;
DEMANDA = 35 37 22 32 41 32 43 38;
CUSTO = 6 2 6 7 4 2 5 9
        4 9 5 3 8 5 8 2
        5 2 1 9 7 4 3 3
        7 6 7 3 9 2 7 1
        2 3 9 5 7 2 6 5
        5 5 2 2 8 1 4 3;
ENDDATA
```

Uma boa alternativa para se inserir dados em um modelo no LINGO, é utilizar a função de interface @FILE. Esta função importa dados de arquivos texto e sua correta sintaxe é @FILE('nome do arquivo').

- **A função objetivo:**

Para escrever uma função que representa o problema descrito, faz-se o uso da variável $VOLUME_{ij}$, que denota o número de *widgets* que segue o caminho do armazém i ao vendedor j. Desta forma, seguindo a mesma linha de raciocínio para o atributo CUSTO, tem-se:

$$minimizar \sum_{ij} CUSTO_{ij} * VOLUME_{ij} \qquad (A.1)$$
$$\forall i \in ARMAZENS$$

A modelagem desta função em LINGO, cujos operadores já foram definidos anteriormente, pode ser realizada da seguinte forma:

```
MIN = @SUM(CAMINHOS(I,J): CUSTO(I,J)*VOLUME(I,J));
```

- **As restrições:**

Usando notação matemática, de acordo com as Tabelas 12 e 13, as restrições de capacidade e de demanda são escritas de acordo com as equações A.2 e A.3.

$$\sum_{j} VOLUME_{ij} \leq CAPACIDADE_i \qquad (A.2)$$
$$\forall i \in ARMAZENS$$

$$\sum_i VOLUME_{ij} = DEMANDA_j \qquad (A.3)$$
$$\forall j \in VENDEDORES$$

Suas respectivas modelagens em LINGO podem ser realizadas das seguintes maneiras:

```
@FOR(ARMAZENS(I):
 @SUM(VENDEDORES(J): VOLUME(I,J)) <=
  CAPACIDADE(I);
);

@FOR(VENDEDORES(J):
 @SUM(ARMAZENS(I): VOLUME(I,J)) =
  DEMANDA(J);
);
```

- **Resultados:**

Resolvendo o problema, são gerados relatórios para acompanhamento da solução. Neste caso, o problema é bem simples de resolver e como solução são gerados os valores computados na Tabela 15.

Tabela 15 – Resultados do problema de otimização

VARIÁVEL	VALOR
VOLUME(WH1,V2)	19
VOLUME(WH1,V5)	41
VOLUME(WH2,V1)	1
VOLUME(WH2,V4)	32
VOLUME(WH3,V2)	11
VOLUME(WH3,V7)	40
VOLUME(WH4,V6)	5
VOLUME(WH4,V8)	38
VOLUME(WH5,V1)	34
VOLUME(WH5,V2)	7
VOLUME(WH6,V3)	22
VOLUME(WH6,V6)	27
VOLUME(WH6,V7)	3

REFERÊNCIAS

[1] TOJO, A. *Robótica*. Disponível em: <http://www.ele.ita.cta.br/ tojo/arquivos-apres.ppt>. Acesso em: 21.11.2014.

[2] IFR. *Industrial robots*. Disponível em: <http://www.ifr.org/industrial-robots/>. Acesso em: 20.11.2014.

[3] SCIAVICCO, B.; SICILIANO, L. *Modelling and Control of Robot Manipulators*. [S.l.]: Hardcover, 1995.

[4] AUTOENTUSIASTAS. *As leis da robótica e o carro autônomo*. 2013. Disponível em: <http://autoentusiastas.com.br>. Acesso em: 17.09.2014.

[5] ANZEVE. *BROKK 330*. 2008. Disponível em: <http://www.anzeve.com>. Acesso em: 17.09.2014.

[6] SERL. *Completed quadrotor construction*. 2012. Disponível em: <http://systemsengineeringresearchlaboratory.org>. Acesso em: 18.09.2014.

[7] ROBOTS, C. *Mobilerobots Pioneer 3 (P3 DX)*. 2007. Disponível em: <http://www.conscious-robots.com>. Acesso em: 18.09.2014.

[8] CIENCIAHOJE. *Veículo submarino autônomo prepara-se para monitorizar barragem em ambiente real*. 2012. Disponível em: <http://www.cienciahoje.pt>. Acesso em: 18.09.2014.

[9] SUJIT, P.; KINGSTON, D.; BEARD, R. Cooperative forest fire monitoring using multiple uavs. In: IEEE. *Decision and Control, 2007 IEEE*. [S.l.], 2007. p. 4875–4880.

[10] MONTEIRO, D. C.; MADRID, M. K. Planning of robot trajectories with genetic algorithms. In: *Robot Motion and Control, 1999. RoMoCo '99. Proceedings of the First Workshop on*. [S.l.: s.n.], 1999. p. 223–228.

[11] MATHEW, N.; SMITH, S.; WASLANDER, S. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 3497–3502. ISSN 1050-4729.

[12] FUJII, K.; HIGUCHI, K.; REKIMOTO, J. Endless flyer: a continuous flying drone with automatic battery replacement. In: IEEE. *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. [S.l.], 2013. p. 216–223.

[13] SWIERINGA, A. K. et al. Autonomous battery swapping system for small-scale helicopters. In: IEEE. *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. [S.l.], 2010. p. 3335–3340.

[14] RIBEIRO, G. C. M. *Teleoperação Bilateral de Múltiplos Robôs Aplicada ao Transporte de Carga*. Dissertação (Mestrado) — UFRJ, 2013.

[15] PAULO, U. de S. *Sistemas para VANTs podem colaborar para o monitoramento da Amazônia*. 2013. Disponível em: <http://www.saocarlos.usp.br>. Acesso em: 20.11.2014.

[16] PEREIRA, L. A. M. *Robô Autômato para Monitoramento de Rebanho e Medição da Forragem do Pasto*. Dissertação (Mestrado) — USP, 2009.

[17] NIGAM, N.; KROO, I. Persistent surveillance using multiple unmanned air vehicles. In: IEEE. *Aerospace Conference, 2008 IEEE*. [S.l.], 2008. p. 1–14.

[18] GONCALVES, V. M. et al. Coordination of multiple fixed-wing uavs traversing intersecting periodic paths. In: IEEE. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.], 2013. p. 849–854.

[19] MIRZAEI, M. et al. Cooperative multi-vehicle search and coverage problem in uncertain environments. In: IEEE. *Decision and Control and European Control Conference (CDC-ECC), , 2011 IEEE*. [S.l.], 2011. p. 4140–4145.

[20] SNAPE, J. et al. The hybrid reciprocal velocity obstacle. *Robotics, IEEE Transactions on*, v. 27, n. 4, p. 696–706, Aug 2011. ISSN 1552-3098.

[21] SOLTERO, D. E.; SMITH, S.; RUS, D. Collision avoidance for persistent monitoring in multi-robot systems with intersecting trajectories. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2011. p. 3645–3652. ISSN 2153-0858.

[22] AKELLA, S.; HUTCHINSON, S. Coordinating the motions of multiple robots with specified trajectories. In: *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*. [S.l.: s.n.], 2002. v. 1, p. 624–631 vol.1.

[23] PENG, J.; AKELLA, S. Coordinating the motions of multiple robots with kinodynamic constraints. In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. [S.l.: s.n.], 2003. v. 3, p. 4066–4073 vol.3. ISSN 1050-4729.

[24] MESHREF, H. The rod-bearing problem: a cooperative autonomous robotics application based on artificial idiotypic immune networks. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. [S.l.], 2013. p. 1654–1659.

[25] BURGARD, W. et al. Collaborative multi-robot exploration. In: IEEE. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. [S.l.], 2000. p. 476–481.

[26] SMITH, L. S.; SCHWAGER, M.; RUS, D. Persistent robotic tasks: monitoring and sweeping in changing enviroments. *Robotics, IEEE Transactions on*, IEEE, v. 28, n. 2, p. 410–426, 2011.

[27] DERENICK, J.; MICHAEL, N.; KUMAR, V. Energy-aware coverage control with docking for robot teams. In: IEEE. *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. [S.l.], 2011. p. 3667–3672.

[28] LITUS, Y.; ZEBROWSKI, P.; VAUGHAN, T. R. A distributed heuristic for energy-efficient multirobot multiplace rendezvous. *Robotics, IEEE Transactions on*, IEEE, v. 25, n. 1, p. 1552–3098, 2009.

[29] COSTA, P. P. *Teoria de Grafos e suas Aplicações*. Dissertação (Mestrado) — UNESP, 2011.

[30] MELO, D. K. R. *Aplicações de busca em grafos*. Disponível em: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/david_krenkel_rodrigues_de_melo.pdf>. Acesso em: 03.11.2014.

[31] ARAUJO, A. *As pontes de Königsberg*. Disponível em: <http://www.mat.uc.pt/ alma/escolas/pontes/>. Acesso em: 21.08.2014.

[32] LIMA, A. *Teoria dos grafos usando C++*. 2012. Disponível em: <http://grafos-cplusplus.blogspot.com.br>. Acesso em: 02.10.2014.

[33] VITOR, U. R. C. *Transmissão de Dados Via Rede Elétrica: Função Transferência Utilizando Grafo de Fluxo*. Tese (Doutorado) — UFPE, 2013.

[34] MONTEZANO, A. F. M. *Modelo em Rede de Petri de um Sistema de Automação de Elevador de Passageiros*. 2009.

[35] BALL, W.; COXETER, H. *Mathematical Recreations and Essays*. [S.l.]: Dover, 1987.

[36] GREAT, G. the. *Un problema de cuidades y carreteras*. 2011. Disponível em: <http://spltmatematicas.blogspot.com.br>. Acesso em: 25.09.2014.

[37] SIQUEIRA, P. H. *Uma Nova Abordagem na Resolução do Problema do Caixeiro Viajante*. Tese (Doutorado) — UFPR, 2005.

[38] PEDRO, O. R. *Uma Abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios*. Dissertação (Mestrado) — UFMG, 2013.

[39] KALID, R. *Otimização de processos e sistemas*. Disponível em: <http://www.moodle.ufba.br>. Acesso em: 06.12.2014.

[40] CORKE, P. *Robotics, vision and control: fundamental algorithms in MATLAB*. [S.l.]: Springer, 2011.

[41] INC, L. S. 2014. Disponível em: <http://www.lindo.com>. Acesso em: 16.12.2014.

[42] INC, L. S. *LINGO user's guide*. [S.l.]: LINDO Systems Inc, 2008.

[43] CENTRE, U. vehicle. *Pioneer 3 DX*. 2010. Disponível em: <http://mecatron.rma.ac.be>. Acesso em: 10.10.2014.

[44] ROBOTWORK. *KUKA to use youBot in manipulation competition*. 2013. Disponível em: <http://www.robots.com>. Acesso em: 05.09.2014.

[45] OPEN SOURCE ROBOTICS FOUNDATION. *Ros Documentation*. Disponível em: <http://wiki.ros.org/>. Acesso em: 20.10.2014.

[46] MARTINEZ, A.; FERNANDEZ, E. *Learning ROS for Robotics Programming*. Packt Publishing, 2013. ISBN 1782161449. Disponível em: <`http://www.packtpub.com/learning-ros-for-robotics-programming/book`>.

[47] BOHREN, J. *ROS Crash-Course, Part I*. Disponível em: <http://courses.cs.washington.edu/courses/cse466/11au/calendar/ros_cc_1_intro-jrsedit.pdf>. Acesso em: 19.12.2014.

[48] OPEN SOURCE ROBOTICS FOUNDATION. *Ros Documentation*. Disponível em: <http://answers.ros.org/questions/>. Acesso em: 21.12.2014.

[49] TEIXEIRA, A. M. et al. Coordenação Ótima de múltiplos robôs de serviço e de recarga em tarefas persistentes. In: IEEE. *International Conference on Industry Applications Induscon, 2014 IEEE International Conference on*. [S.l.], 2014. p. 849–854.