ALGORITMOS DE CONTROLE DE TRÁFEGO PARA ENXAMES DE ROBÔS COM ALVOS EM COMUM

YURI TAVARES DOS PASSOS

ALGORITMOS DE CONTROLE DE TRÁFEGO PARA ENXAMES DE ROBÔS COM ALVOS EM COMUM

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

ORIENTADOR: LUIZ CHAIMOWICZ

Belo Horizonte

Março de 2012

Tavares dos Passos, Yuri
Algoritmos de Controle de Tráfego para Enxames de Robôs com Alvos em comum / Yuri Tavares dos Passos. — Belo Horizonte, 2012
xviii, 86 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Luiz Chaimowicz

1. Enxames de robôs. 2. Robótica. 3. Controle de tráfego. 4. Congestionamentos. I. Título.

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

FOLHA DE APROVAÇÃO

algoritmos de controle de tráfego para enxames de robôs com alvos em comum

YURI TAVARES DOS PASSOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

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

PROF. VALDIR GRASSI JÚNIOR
Departamento de Engenharia Elétrica - USP

Belo Horizonte, 15 de março de 2012.

Resumo

Enxames de robôs são sistemas compostos por uma grande quantidade de robôs, relativamente simples, dispostos no mesmo espaço e interagindo entre si para alcançar um objetivo comum. Dentre os problemas enfrentados em enxames de robôs, têm-se o controle de tráfego. Neste problema, os robôs devem se coordenar para evitar congestionamentos, especificamente quando os robôs devem se movimentar para o mesmo local simultaneamente. Neste trabalho, foi desenvolvido três algoritmos distribuídos para o controle de tráfego em enxames de robôs, permitindo evitar congestionamentos tanto na chegada ao alvo em comum quanto na saída. O primeiro algoritmo consiste em formar regiões de entrada e saída próximo ao alvo. O segundo consiste em criar uma fila em forma de espiral em torno do alvo. O terceiro utiliza troca de mensagens entre robôs visando expulsar alguns robôs, liberando passagem para os que estão próximos ao alvo. É feita uma análise de cada um deles e uma comparação de seus desempenhos.

Palavras-chave: Enxames de robôs, Robótica, Controle de tráfego, Congestionamentos.

Abstract

Robotic Swarms are systems formed by a large number of robots, relatively simple, placed in same space and interacting among themselves to fulfill a common goal. Among the problems encountered in swarms of robots, there exists traffic control. In this problem, robots must be coordinated to avoid congestion, specificly when robots must move to same local, simultaneously. In this work, we developed three distributed algorithms for traffic control in robotic swarms, allowing to avoid congestions both on arrival at common target and leaving it. The first algorithm consists in using entry and exiting regions next to the common target. The second one consists in making a spiral-shaped queue around the target. The third one uses message comunication to drive away some robots, releasing the passage for those are next to the target. We made an analysis of each one and compare the performance of these algorithms.

Keywords: Robotic swarm, Robotics, Traffic control, Congestions.

Lista de símbolos

a	Largura da espiral.
D	Distância miníma até o alvo para utilizar o algoritmo de coordenação.
$\mathbf{F}$	Vetor força resultante.
$\mathbf{F_R}$	Vetor de força repulsiva.
$\mathbf{F_A}$	Vetor de força atrativa.
$\mathbf{F_T}$	Vetor de força tangencial do algoritmo `Espiral`.
$\mathbf{F_N}$	Vetor de força normal do algoritmo `Espiral`.
$\mathbf{F_I}$	Vetor de força tangencial inversa do algoritmo `Espiral`.
$\mathbf{F_P}$	Vetor de força repulsiva projetada na tangencial do algoritmo `Espiral`.
$\mathbf{F_M}$	Vetor de força enviada na mensagem dos robôs em estado ENTRANDO no algoritmo `Sirene`.
$\mathbf{F_E}$	Vetor de força de expulsão replicada pelos robôs no algoritmo `Sirene`.
K	Constante multiplicativa para força resultante.
ρ	Probabilidade de transição para estado IMPACIENTE.
$\mathbf{q}$	Vetor posição do robô.
$Raio$	Raio das áreas de perigo e livre.
T_P	Iterações para entrar no estado IMPACIENTE.
T_M	Quantidade de iterações que o robô deve esperar até mandar a próxima mensagem.

T_E No algoritmo `Região`: Quantidade de iterações consecutivas que o robô deve esperar para perceber se não há um robô à sua frente.

T_E No algoritmo `Sirene`: Quantidade de iterações que o robô deve seguir a força de expulsão.

α_R Ângulo de percepção para réplica de mensagem no algoritmo `Sirene`.

α_A Ângulo de percepção para ignorar expulsão no algoritmo `Sirene`.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Enxames de robôs são sistemas compostos por uma grande quantidade de robôs simples que normalmente compartilham objetivos em comum. Nestes sistemas, a execução de uma dada tarefa é processada de modo descentralizado, ou seja, cada robô deve processar as informações obtidas localmente por seus sensores e eventualmente comunicar o resultado aos outros robôs no enxame. Em geral, os robôs possuem um limitado poder de processamento e de comunicação. Devido a esta limitação, os algoritmos desenvolvidos devem ser robustos e escaláveis, garantindo que a inclusão de mais robôs aumentem a eficiência destes sistemas [Sahin, 2004].

A inspiração dos enxames de robôs surge do comportamento de colônias de insetos, como formigas e abelhas. Nestas colônias, os indivíduos interagem entre si usando comunicação local e processamento individual, ou seja, não existe uma unidade central de controle administrando cada um dos indivíduos.

Este tipo de paradigma no controle de múltiplos robôs – descentralizado e por comunicação local – torna o sistema mais robusto e escalável. A robustez se torna evidente quando alguns robôs passam a não funcionar corretamente: a perda de robôs em enxames pode diminuir a eficiência do sistema, mas não impede que execute sua tarefa normalmente. A escalabilidade é definida como a não degradação do sistema à medida que se aumenta o número de indivíduos. O algoritmo de controle dos enxames de robôs deve ser projetado de tal forma que a inclusão de mais robôs para o grupo tenda a melhorar o desempenho do sistema. Se o enxame de robôs tivesse uma unidade de controle central, isto não seria possível porque a inclusão de novos robôs só aumentaria sua carga de processamento.

O ideal é que um enxame de robôs seja implementado visando a flexibilidade, ou seja, os robôs agirão de modo que situações diferentes gerem comportamentos diferentes, ainda assim sendo capazes de executar a tarefa principal. Na natureza, são

encontrados alguns exemplos de flexibilidade como numa colônia de formigas. Seu objetivo principal é coletar alimento para o formigueiro, mas subtarefas devem ser cumpridas quando situações inusitadas surgem durante a execução da tarefa principal: fuga de predadores, carregamento de alimentos pesados ou desvio de rotas bloqueadas por derramamento de líquidos ou objetos deixados no meio da rota.

Num ambiente compartilhado por vários robôs, a tendência de um robô interferir com o outro é maior. Em especial, um dos problemas existentes na navegação de enxames é o congestionamento, especificamente quando os robôs devem se movimentar para o mesmo local simultaneamente. É comum que robôs possuam alvos distintos que compartilham uma mesma região do espaço ou um mesmo alvo em comum, como ocorre em navegação por pontos de interesse (*waypoint navigation*). Isto também ocorre no problema de coleta (*foraging*) [Sahin et al., 2008], quando os robôs devem se deslocar para pontos do ambiente para coletar itens e transportá-los para um local específico. Pode-se citar aplicações para o problema da coleta como transporte de materiais tóxicos ou escombros provenientes de desabamentos.

O controle de tráfego é importante pois a presença de outros robôs no ambiente irá atrapalhar consideravelmente o planejamento do caminho de cada robô. Em um cenário onde uma grande quantidade de robôs compartilham o mesmo ambiente, o tempo de trajeto para cada robô poderá ser substancialmente aumentado devido ao congestionamento causado por outros robôs obstruindo sua passagem. O controle de tráfego é necessário, visto que a presença de outros robôs em sistemas cooperativos será constante na maioria das tarefas.

O problema do congestionamento poderia ser resolvido utilizando uma unidade central de processamento para computar as melhores trajetórias para cada robô. Porém, há a desvantagem do sistema ficar dependente desta unidade central, além de não ser uma solução escalável para um grande número de robôs. No contexto de enxames, um dos desafios no desenvolvimento destes sistemas é a garantia de que o todo irá funcionar corretamente sem que cada uma das partes tenha conhecimento global do ambiente.

1.1 Definição do Problema e Objetivos

O problema a ser tratado neste trabalho será o controle de tráfego em enxames de robôs quando existe um alvo em comum para eles. Em outras palavras, este problema pode ser enunciado como a seguir:

Dados N robôs no ambiente e um alvo em comum para todos eles, os

robôs devem cooperar entre si de forma descentralizada, usando apenas comunicação local, a fim de possibilitar que todos alcancem o alvo evitando congestionamentos e saiam dessa região no menor tempo possível.

Portanto, de forma a atacar esse problema, o objetivo deste trabalho é desenvolver algoritmos descentralizados para controle de tráfego em enxames de robôs quando existe um alvo em comum para muitos deles. Foram desenvolvidos três algoritmos diferentes e analisado o desempenho de cada um deles, comparando-os. O primeiro algoritmo consiste em estender um trabalho já realizado para resolver este problema [Marcolino & Chaimowicz, 2009], incluindo um controle de regiões de acesso, tornando mais eficiente a chegada e saída do alvo. O segundo algoritmo consiste em coordenar os robôs para que sigam uma fila em forma de espiral, cujo centro é onde se encontra o alvo. O terceiro algoritmo proposto consiste em repelir os outros robôs quando algum robô se aproxima do alvo através de mensagens emitidas pelos robôs que chegam próximo ao alvo e saem dele.

1.2 Organização deste documento

O texto está organizado como segue. O Capítulo 2 apresenta alguns trabalhos relacionados, apresentando de forma geral a área de enxames de robôs e citando alguns trabalhos que lidam com controle de tráfego para sistemas multirrobôs. Capítulo 3 descreve os três algoritmos propostos, apresentando as suposições que foram tomadas e detalhando o funcionamento de cada um deles. Capítulo 4 apresenta os cenários de teste e resultados obtidos da experimentação dos algoritmos propostos em simulação e em ambiente real. Os algoritmos são testados individualmente, para verificar a influência de seus parâmetros, e em conjunto, para comparar a eficiência de cada algoritmo. Capítulo 5 mostra as conclusões do trabalho realizado.

Capítulo 2

Revisão Bibliográfica

Inteligência distribuída se refere a sistemas de entidades que trabalham juntos para planejar, raciocinar, resolver problemas, pensar abstratamente, compreender ideias e linguagem, e aprender [Parker, 2008]. Neste contexto, uma entidade é qualquer tipo de processo inteligente, como agentes ou robôs. Segundo Parker [2008], as interações entre as entidades podem ser divididas em três eixos:

- ciente ou não ciente: se cada entidade tem conhecimento das demais envolvidas;
- objetivos individuais ou compartilhados: se as entidades dividem o mesmo objetivo ou cada uma possui um objetivo específico;
- influência das ações nos objetivos das outras entidades: quando uma ação tomada por uma entidade, pode ou não influenciar o objetivo de outra entidade.

Baseado nestes três eixos, quatro tipos de interação podem ser definidos. A interação é do tipo *Coletiva* quando as entidades não são cientes das demais, compartilham os mesmos objetivos e suas ações influenciam as das demais entidades. O segundo tipo de interação é a *Cooperativa*, na qual as entidades conhecem as demais, compartilham os mesmo objetivos e suas ações são benéficas para os outros. O terceiro tipo é o *Colaborativo*, que ocorre quando os objetivos são individuais, eles possuem conhecimento um dos outros e suas ações influenciam os demais. Por último, o *Coordenativo* é caracterizado pelo conhecimento dos demais, os objetivos são individuais, mas suas ações não influenciam os demais.

Considerando estes tipos de sistemas de inteligência distribuída e assumindo que as entidades são robôs, segundo Parker [2008], sistemas de robôs cooperativos são sistemas de inteligência distribuída, com robôs dispostos no mesmo ambiente que conhecem os demais, compartilham os mesmos objetivos e suas ações influenciam positivamente

as dos outros. Estes robôs podem ser iguais ou diferentes entre si, assim como podem estar em pequena ou grande quantidade. Sistemas de robôs cooperativos possuem aplicações industriais [Simmons et al., 2000], militares [Parker, 2003], em exploração espacial [Goldberg et al., 2002] e operações de resgate [Murphy et al., 2002], entre outras. Quando os robôs são relativamente simples, estão numa quantidade relativamente grande no mesmo espaço e precisam interagir entre si para alcançar um objetivo, de forma coletiva ou cooperativa, então este sistema pode ser considerado um enxame de robôs.

Enxames de robôs é uma área relativamente nova. Existem possíveis aplicações, mas devido ao caráter recente desta área de pesquisa, elas ainda não foram executadas em situações reais. Dentre elas têm-se: resgate em ambientes urbanos [Mondada et al., 2002; Rahmani, 2004] e minas [Songdong & Jianchao, 2008], detecção de gases tóxicos [Cui et al., 2004], [Hayes, 2002] e exploração espacial [Truszkowski et al., 2004]. O trabalho de Truszkowski et al. descreve um sistema de enxames de robôs a ser utilizado para exploração de asteróides, previsto para entre 2020 e 2030, visto que robôs mais leves podem ser atraídos pelo seu campo gravitacional. Em geral, na maioria dessas tarefas, é mais vantajoso o uso de enxames de robôs ao invés de um robô individual especializado devido a requisitos como custo, tempo de execução e risco de dano.

A ideia de enxames de robôs surgiu de uma tentativa de modelar sistemas robóticos celulares [Beni & Wang, 1989]. Neste contexto surge o termo Inteligência de Enxames (*Swarm Intelligence*) [Kennedy et al., 2001], iniciando os estudos de sistemas formados por agentes que processam os dados de forma descentralizada e auto-organizada. A inspiração desta área de pesquisa se deve ao comportamento de diversos seres vivos, desde grupos de células a insetos. Neste sistemas biológicos, cada indivíduo processa informações de forma descentralizada e por comunicação local. Não existe uma unidade central de processamento [Camazine et al., 2001], mas estes sistemas conseguem ser auto-organizados, executando tarefas para sua sobrevivência de forma coordenada.

Neste contexto, pode-se dizer que enxames de robôs são sistemas que possuem Inteligência Distribuída, ou seja, são compostos por entidades que trabalham juntos para planejar, raciocinar, resolver problemas, pensar abstratamente, compreender ideias e linguagem, e aprender [Parker, 2008].

Existem várias tarefas comumente realizadas por enxames. Na literatura desta área, estas tarefas podem ser divididas nos seguintes grupos [Sahin et al., 2008] [Cao et al., 1997]:

- Agregação: os robôs, inicialmente espalhados no ambiente, devem se agregar em

uma região do espaço;

- Dispersão: os robôs, inicialmente reunidos numa região do espaço, devem se afastar o máximo possível um do outro;
- Coleta (do inglês *Foraging*): o objetivo de cada um deles é coletar objetos específicos que se encontram no ambiente de forma coordenada e levar de volta para uma área;
- Automontagem: eles devem se coordenar para se encaixar uns nos outros, tornando grupos de robôs conectados fisicamente;
- Movimento Conectado: quando os robôs estão conectados fisicamente, é necessário que o grupo esteja coordenado para se mover como um todo;
- Controle de Tráfego: a coordenação tem o intuito de estabelecer um padrão de movimentação de tal forma que não haja problemas de tráfego, isto é, um robô não atrapalha na navegação de outro;
- Transporte Coordenado: objetos maiores devem ser transportados por vários robôs usando coordenação;
- Formação de padrões: os robôs devem formar padrões geométricos entre eles como formação em corrente, formação em bando ou se movimentar em forma de figuras geométricas;
- Construção auto-organizada: os robôs devem levar objetos espalhados no ambiente a lugares comuns, com intuito de montar alguma estrutura.

Como mencionado, o problema específico que será tratado nesta dissertação é o controle de tráfego para evitar congestionamentos em situações onde os robôs possuem o mesmo alvo. Na literatura, existem alguns trabalhos que lidam com o controle de tráfego para agentes e robôs cooperativos. Em [Cao et al., 1997] é denominado conflito de recursos quando um recurso único e indivisível é requisitado por vários robôs. No contexto deste trabalho, o recurso em questão é o alvo, representado por uma região pequena do espaço. Há trabalhos que lidam com o controle de tráfego de robôs cooperativos, mas utilizam marcas para delimitar regiões de tráfego [Caloud et al., 1990; Asama et al., 1991]. Em [Grossman, 1988] é apresentado um algoritmo para controle de tráfego de veículos guiados automaticamente, mas o controle é feito de maneira centralizada. Também existem trabalhos que tratam este problema de forma descentralizada utilizando regras de tráfego que cada robô obedece, mas não considera

o problema específico onde todos desejam chegar no mesmo ponto [Kato et al., 1992; Wang, 1991].

Existem trabalhos recentes que lidam com controle de tráfego em grupos de robôs. Em Olmi et al. [2009], um algoritmo para coordenar caminhos predefinidos de um grupo de robôs é desenvolvido, mas em sua abordagem existe uma unidade central de processamento definindo a alteração das rotas de cada robô em caso de possível colisão. Em Guo & Parker [2002], um algoritmo distribuído é desenvolvido para planejamento de movimento para múltiplos robôs, contudo seus experimentos não envolvem muitos robôs. O trabalho de Peasgood et al. [2008] trata o problema de colisão de trajetórias para vários robôs, mas num contexto onde os alvos de cada um podem ser diferentes. Em [Hoshino, 2011], é tratado o congestionamento na forma de engarrafamento de vias, ou seja, em vias nas quais os robôs não podem ultrapassar uns aos outros criando agrupamentos desordenados numa região do espaço. O trabalho de Krontiris & Bekris [2011] trata o problema da resolução de choques entre robôs de modo descentralizado, usando uma extensão de uma política de prevenção de obstáculos denominada *Generalized Roundabout Policy* [Pallottino et al., 2007]. Apesar de ser um trabalho que lida com o problema de resolução de conflitos de modo descentralizado e usando poucas trocas de mensagens, os robôs não possuem um alvo em comum.

Nos trabalhos de Sahin [2004] e Sahin et al. [2008], que fazem uma revisão da área de enxames de robôs, não foram encontradas menções sobre o problema exposto neste trabalho, nem, de forma mais geral, sobre controle de tráfego em enxames de robôs. Especificamente, não foi encontrado na literatura algum trabalho que trate para enxames de robôs o problema do conflito de alvo usando controle de tráfego descentralizado, a não ser o de Marcolino & Chaimowicz [2009], cuja extensão aqui está sendo proposta.

Capítulo 3

Algoritmos Propostos

O problema do congestionamento no acesso à alvo comum ocorre quando uma grande quantidade de robôs deve acessar o mesmo alvo ou a mesma região de espaço (Figura 3.1). O controle de tráfego para este problema consiste em estabelecer regras ou comportamentos para os robôs de tal forma que minimize o congestionamento próximo ao alvo. Reduzindo o congestionamento, espera-se diminuir o tempo de chegada e saída no alvo e a quantidade de choques entre estes robôs.

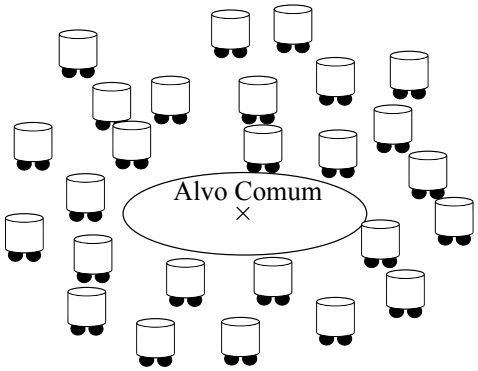


Figura 3.1. Problema do alvo em comum.

Para solucionar o problema do congestionamento para controle de tráfego em enxames de robôs foram desenvolvidos três algoritmos distintos. O primeiro algoritmo desenvolvido consiste em dividir a área próxima ao alvo em regiões de entrada e saída. O segundo algoritmo tem por objetivo formar uma fila em forma de espiral em torno do alvo. O terceiro algoritmo visa simular o efeito das sirenes usadas no trânsito por ambulâncias e viaturas com o propósito de expulsar os robôs que estão no caminho de entrada e saída do alvo.

Todos os três algoritmos desenvolvidos utilizam campos potenciais artificias [La-

tombe, 1991] para navegação dos robôs. Seu uso foi motivado devido à sua facilidade de implementação para robôs que utilizam somente informações locais para locomoção e tomada de decisões. Estas informações locais são obtidas por sensores de alcance local e limitado. Para os algoritmos propostos, supõe-se que o sensor utilizado não possui falhas. Tipicamente, para o planejamento de caminhos usando campos potenciais são utilizados dois tipos de campo: um campo repulsivo, que o repele de outros robôs e um campo atrativo, que gera uma força que atrai o robô ao alvo.

Os robôs utilizam um campo potencial repulsivo para evitar se chocarem com seus vizinhos. Este campo está sempre presente, não importa qual algoritmo de coordenação está sendo utilizado. As forças resultantes deste campo são geradas localmente a partir dos sensores do robô, sem necessidade de conhecimento *a priori* sobre o ambiente e a quantidade total de robôs presentes. Para gerar forças repulsivas é usada a seguinte função [Siegwart & Nourbakhsh, 2004]:

$$\mathbf{F_R} = \begin{cases} -K * \left(\frac{1}{d} - \frac{1}{I}\right)\left(\frac{\mathbf{o}-\mathbf{q}}{d^3}\right) & \text{se } d < I \\ 0 & \text{caso contrário} \end{cases}$$

onde $\mathbf{q} = [x, y]^T$ é a posição atual do robô, $\mathbf{o} = [x_o, y_o]^T$, a posição de outro robô, $d = \|\mathbf{o} - \mathbf{q}\|$, a distância Euclidiana entre $\mathbf{o}$ e $\mathbf{q}$ e I a influência do outro robô, isto é, a distância máxima para se considerar algo como obstáculo.

De um modo geral, os algoritmos propostos visam alterar as forças do campo potencial atrativo a depender das informações obtidas localmente pelos robôs. A partir destas informações, os robôs podem: (i) escolher um novo alvo, de caráter temporário, tratado como uma subtarefa antes de alcançar o alvo principal, ou (ii) seguir um sentido diferente do sentido que vai de encontro ao alvo principal, visando evitar regiões muito congestionadas ou abrir espaço para outros robôs.

Os algoritmos foram projetados seguindo o paradigma reativo [Murphy, 2000]. Neste paradigma, para cada comportamento preestabelecido, o robô executa uma ação que consiste em *sentir* e imediatamente *agir*. Neste trabalho, a etapa *sentir* consiste na obtenção de informações advindas dos sensores e de mensagens recebidas – para algoritmos que exigem troca de mensagens – e a etapa *agir* processa essas informações transformando em atuação no motor. A Figura 3.2 apresenta um esquema que representa o sistema com os dados de entrada e saída. A cada iteração do ciclo sentir-agir, os sensores são lidos e mensagens recebidas são usadas como entrada, quando o algoritmo envolve troca de mensagens. Estes dados são processados e geram como saída um vetor $\mathbf{F} = [F_x, F_y]^T$, que será usado como entrada para um controlador responsável

pela atuação do robô.

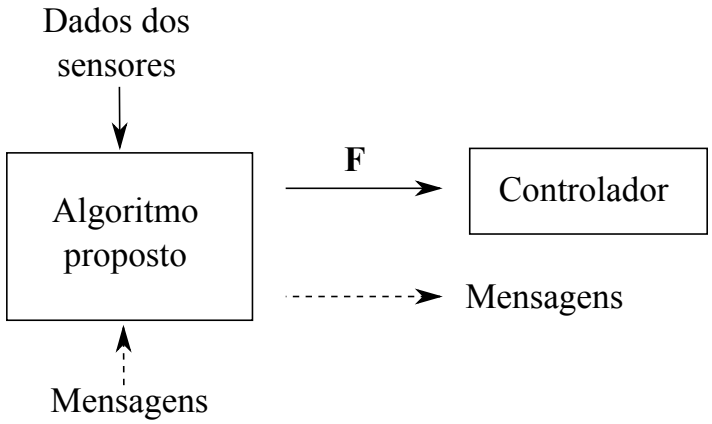


Figura 3.2. Esquema geral dos algoritmos desenvolvidos.

Caso o algoritmo envolva troca de mensagens, mensagens são enviadas e recebidas pelos robôs. Considera-se que as mensagens entre robôs possuem um raio de alcance igual ao valor de I. Assim, mensagens só serão enviadas quando um robô perceber outro em sua área de influência. Para os algoritmos propostos, supõe-se que apenas robôs são percebidos pelos sensores. Esta condição pode ser relaxada com o uso de algum protocolo de troca de mensagens que permita a identificação entre robôs.

Considerando um robô completamente atuado, o modelo dinâmico utilizado é dado por:

$$\mathbf{u} = \mathbf{F} - C\mathbf{v},$$

$$\dot{\mathbf{q}} = \mathbf{v},$$

$$\dot{\mathbf{v}} = \mathbf{u}.$$

onde: $\mathbf{u}$ é a entrada de controle, $\mathbf{v}$ a velocidade do robô e o termo $C\mathbf{v}$ é um fator de amortecimento.

Visto que os algoritmos propostos foram projetados para enxames de robôs, os robôs não possuem informação *a priori* de quantos robôs existem no ambiente. Assim, os algoritmos propostos só entram em funcionamento quando o robô alcança uma distância mínima de D metros do alvo.

Neste capítulo serão apresentados os algoritmos originais, bem como as melhorias implementadas após a realização de alguns experimentos. Além disso, os algoritmos são explicados considerando a localização dos robôs num referencial global. Contudo, se os robôs estiverem equipados com um dispositivo que indique um sentido comum a todos (por exemplo uma bússola), os algoritmos podem ser implementados usando referencial local e, posteriormente, transformando-os para o referencial global obtido

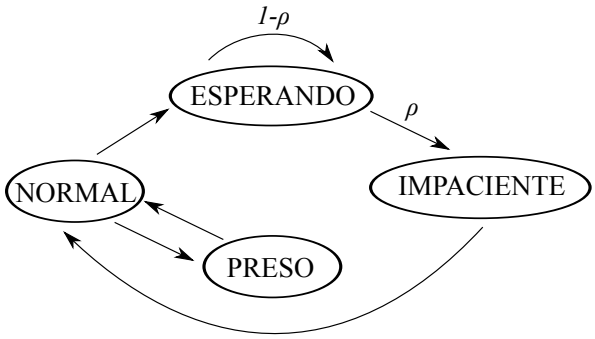


Figura 3.3. Máquina de estados do algoritmo.

pelo dispositivo. Este capítulo está organizado como segue: a Seção 3.1 explica o funcionamento do algoritmo que usa regiões de entrada e saída, a Seção 3.2 explica o algoritmo que cria uma fila em espiral e a Seção 3.3, o algoritmo que simula sirenes.

3.1 Algoritmo 1: Região de entrada e saída

A idéia principal deste algoritmo é evitar que muitos robôs cheguem simultaneamente ao mesmo alvo. Trata-se de uma extensão do trabalho proposto por Marcolino & Chaimowicz [2009], onde é descrito um algoritmo que consiste em cada robô usar uma máquina de estados probabilística para se coordenar dentro de uma região em volta do alvo. O algoritmo proposto consiste em estender este incluindo um controle de regiões de acesso, tornando mais eficiente a chegada e saída do alvo. O algoritmo é executado de forma completamente distribuída e os robôs utilizam apenas comunicação e percepção locais.

3.1.1 Algoritmo distribuído para controle de congestionamento

A idéia principal do algoritmo proposto por Marcolino & Chaimowicz [2009] é fazer com que os robôs esperem sua vez para alcançar o alvo, se posicionando numa região próxima a ele, de forma a diminuir a chance de um robô atrapalhar o outro.

A máquina de estados, apresentada na Figura 3.3 [Marcolino & Chaimowicz, 2009], ilustra o comportamento de cada robô. Um robô em estado $NORMAL$ irá se mover para o alvo enquanto desvia de obstáculos. Se ele alcançar a região de perigo, definida como uma região de raio s ao redor do alvo, e encontrar um robô próximo com o mesmo alvo que ele, então ele muda para o estado $ESPERANDO$. Existe também uma área circular menor, com raio $r < s$, chamada de região livre. A Figura 3.4 [Marcolino & Chaimowicz, 2009] mostra as regiões de perigo e livre. Robôs

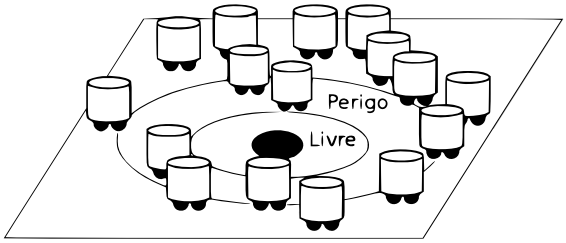


Figura 3.4. Área de perigo e livre.

no estado $ESPERANDO$ ou $PRESO$ não podem entrar nesta região. Cada robô em $ESPERANDO$ tem uma probabilidade ρ de mudar para o estado $IMPACIENTE$, na qual o robô irá para o alvo não se importando se existem outros robôs com o mesmo alvo. Robôs no estado $ESPERANDO$ podem continuar neste estado com probabilidade $1 - \rho$.

Para isso, é usado um gerador de números aleatórios no intervalo $[0, 1]$, obedecendo uma distribuição de probabilidade uniforme. O robô só entra no estado IMPACIENTE se o número obtido é menor que o valor de ρ. Este teste é realizado somente a cada valor fixo de iterações. Este valor é determinado pelo parâmetro *Ciclos para impaciência* (T_P). Esta limitação foi feita para evitar que a cada iteração seja feita um teste, tornando muito rápida a transição para o estado IMPACIENTE.

Se um robô está no estado $NORMAL$ e encontra um robô no estado $ESPERANDO$ na sua frente, ele muda para o estado $PRESO$. No estado $PRESO$, o robô possui o mesmo comportamento do estado $ESPERANDO$, contudo, ele não depende de transições probabilísticas para mudar de estado, basta não haver mais robôs $ESPERANDO$ próximos e ele voltará ao estado $NORMAL$. É importante mencionar que este algoritmo não evita que os robôs se movam ao mesmo tempo para o alvo, mas diminui a chance disso acontecer devido a máquina de estados probabilística utilizada.

Para os robôs perceberem quando existem alvos em comum, eles trocam mensagens com robôs detectados em sua proximidade, informando qual seu alvo e o tipo da mensagem. Os robôs só obedecem o algoritmo acima se o alvo é comum. Para limitar a quantidade de mensagens enviadas, os robôs só podem enviar as mensagens a cada T_M ciclos. O valor T_M será denominado *ciclos para mensagens*.

Existem dois tipos de mensagem: de parada e de atenção. A mensagem de parada é enviada para aqueles que estão no estado $NORMAL$ e se encontram atrás dos que estão no estado $ESPERANDO$, fazendo que mudem para o estado $PRESO$. A mensagem de atenção é enviada para aqueles que estão na região perigo, fazendo com que os que estão no estado $NORMAL$, mudem para o estado $ESPERANDO$.

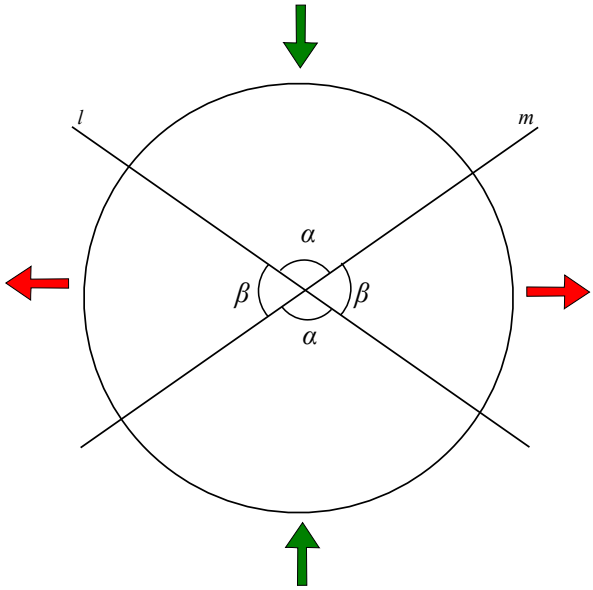


Figura 3.5. Divisão da região de perigo em áreas de entrada e saída.

Para evitar que os robôs fiquem em estado PRESO por pouco tempo, os robôs são forçados a ficarem no estado $PRESO$ por T_E ciclos. O valor T_E representa *ciclos de espera*. Sem o uso de T_E, os robôs ficam trocando do estado NORMAL para PRESO a cada ciclo. Isto ocorre porque, na prática, quando um robô entra no estado $PRESO$, ele permanecerá neste estado enquanto não receber outra mensagem de parada ou não perceber algum robô a sua volta. Se houver erro de sensor, erroneamente será considerado que não houve robô na sua área de influência, fazendo com que haja transição do estado PRESO para NORMAL. Como os outros robôs levam T_M ciclos para enviar uma próxima mensagem, durante o tempo entre o envio de uma mensagem e outra, os robôs no estado PRESO podem fazer a transição para o estado NORMAL e seguir até o alvo, como se não existissem robôs em espera.

Como mostrado em [Marcolino & Chaimowicz, 2009], este algoritmo diminui os congestionamentos até a chegada no alvo. No entanto, a saída dos robôs da região do alvo fica congestionada devido à presença de outros robôs esperando para entrar. O algoritmo proposto neste trabalho pretende contornar este problema.

3.1.2 Algoritmo proposto

O algoritmo proposto neste trabalho visa dividir a área em torno do alvo em regiões para entrada e saída de robôs. Estabelecendo estas regiões, pretende-se diminuir os congestionamentos entre robôs que desejam chegar ao alvo e os que desejam sair. Para isso, a região de perigo será dividida em quatro regiões por ângulos α e β, como ilustra

a Figura 3.5. A fatia dividida pelo ângulo α será usada para entrada dos robôs e a do ângulo β, para saída. Com isso espera-se que os robôs possam entrar e sair da região de alvo sem ocorrência de congestionamentos.

O algoritmo proposto neste trabalho estabelece um novo comportamento aos robôs nos estados $ESPERANDO$ e $PRESO$, quando estão próximos do alvo. Quando um robô se encontra a uma distância D do alvo, ele verifica se está dentro da região de entrada. Caso esteja fora, ele se move para dentro da região. A região de entrada é verificada considerando a posição do alvo, a posição atual do robô, usando um referencial global. Considere $\gamma = 90° - \alpha/2$, o ângulo da reta m, e $\delta = 90° + \alpha/2$, da reta l, conforme mostra a Figura 3.6 (as retas l e m também podem ser vistas na Figura 3.5). Dados (x_G, y_G), as coordenadas do alvo, e (x, y), a posição atual do robô, a seguinte condição determina se o robô está dentro da região:

$$\begin{aligned}
&(y - y_G - \tan\gamma(x - x_G) \geq 0) \wedge (y - y_G - \tan\delta(x - x_G) \geq 0) \quad \text{se } y > y_G;\\
&(y - y_G - \tan\gamma(x - x_G) \leq 0) \wedge (y - y_G - \tan\delta(x - x_G) \leq 0) \quad \text{se } y \leq y_G.
\end{aligned}$$

Se o robô não está na região, ele é impelido a se locomover para o ponto da reta que divide a região de entrada mais próximo de sua posição atual. A distância d entre sua posição atual e o ponto mais próximo da região de entrada é dado por:

$$d = \frac{y - \tan\theta(x - x_G) - y_G}{\sqrt{(\tan^2\theta + 1)}}, \tag{3.1}$$

onde

$$\theta = \begin{cases} \gamma, & \text{se } ((y > y_G) \wedge (x > x_G)) \vee \\ & \quad ((y \leq y_G) \wedge (x \leq x_G)); \\ \delta, & \text{caso contrário.} \end{cases}$$

A Equação 3.1 pode retornar valores positivos ou negativos, a depender da posição do robô em relação a reta mais próxima.

Como ilustrado na Figura 3.6, o ponto (x_W, y_W) para o qual ele deve se locomover é dado por:

$$x_W = x + d\cos\phi; \tag{3.2}$$

$$y_W = y_G + \tan\theta(x_W - x_G). \tag{3.3}$$

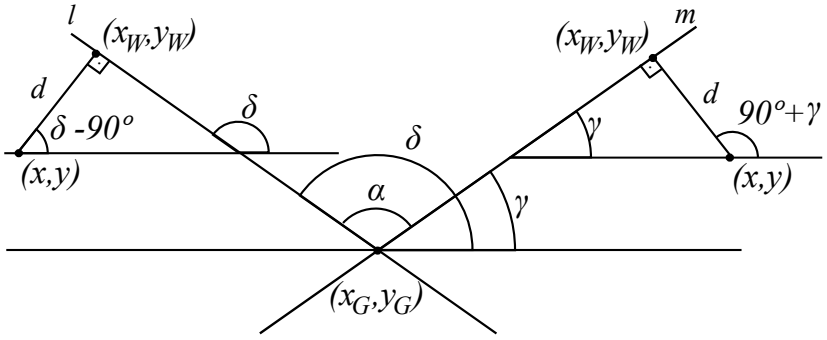


Figura 3.6. Distância mais próxima entre um ponto fora da região e a fronteira da região de entrada.

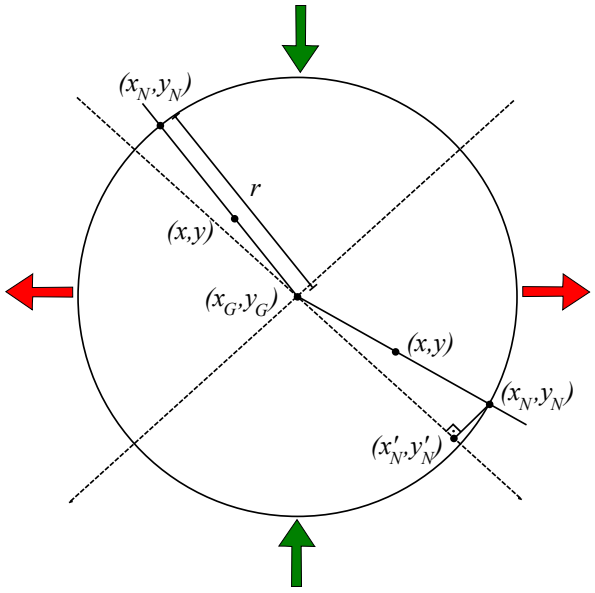


Figura 3.7. Nova posição do robô caso esteja na área livre.

onde

$$\phi = \begin{cases} 90° + \gamma, & \text{se } ((y > y_G) \wedge (x > x_G)) \vee \\ & \quad ((y \leq y_G) \wedge (x \leq x_G)); \\ \delta - 90°, & \text{caso contrário.} \end{cases}$$

Além das regiões delimitadas pelas retas, os robôs, ao entrarem nos estados *ESPERANDO* ou *PRESO*, verificam se não estão na região livre próxima ao alvo (Figura 3.4). Um robô em qualquer um destes dois estados, deve estar a no mínimo uma distância r do alvo. Um robô no estado *ESPERANDO* deve estar a uma distância do alvo que varia de r a s. Se um robô está em *ESPERANDO* e se afasta do alvo, ficando a mais de s metros, ele passa para o estado *PRESO*. Se um robô *ESPERANDO* encontra-se a menos de r metros, ele deve ir para a posição (x_N, y_N) (Figura 3.7), pois fazendo isso o robô não ficará dentro da região livre.

Sabe-se que:

$$\frac{y - y_G}{x - x_G} = \frac{y_N - y_G}{x_N - x_G}, \tag{3.4}$$

$$(x_N - x_G)^2 + (y_N - y_G)^2 = r^2. \tag{3.5}$$

Elevando ao quadrado ambos os lados da Equação 3.4, tem-se:

$$\frac{(y - y_G)^2}{(x - x_G)^2} = \frac{(y_N - y_G)^2}{(x_N - x_G)^2}. \tag{3.6}$$

Usando Equação 3.5 em 3.6, tem-se:

$$\begin{aligned} \frac{(y - y_G)^2}{(x - x_G)^2} &= \frac{r^2 - (x_N - x_G)^2}{(x_N - x_G)^2} \\ \frac{(y - y_G)^2}{(x - x_G)^2} + 1 &= \frac{r^2}{(x_N - x_G)^2} \end{aligned} \tag{3.7}$$

Isolando x_N na Equação 3.7 fica:

$$(x_N - x_G)^2 = \frac{r^2}{\frac{(y-y_G)^2}{(x-x_G)^2} + 1}$$

$$(x_N - x_G) = \frac{r}{\sqrt{\frac{(y-y_G)^2}{(x-x_G)^2} + 1}}$$

$$(x_N - x_G) = \frac{r}{\sqrt{\frac{(y-y_G)^2+(x-x_G)^2}{(x-x_G)^2}}}$$

$$(x_N - x_G) = \frac{r(x - x_G)}{\sqrt{(y - y_G)^2 + (x - x_G)^2}} \tag{3.8}$$

$$x_N = x_G + \frac{r(x - x_G)}{\sqrt{(y - y_G)^2 + (x - x_G)^2}} \tag{3.9}$$

Usando a Equação 3.8 na Equação 3.6, tem-se:

$$\frac{(y - y_G)^2}{(x - x_G)^2} = \frac{(y_N - y_G)^2}{\left(\frac{r(x-x_G)}{\sqrt{(y-y_G)^2+(x-x_G)^2}}\right)^2}$$

$$\frac{(y - y_G)^2}{(x - x_G)^2} \left(\frac{r(x - x_G)}{\sqrt{(y - y_G)^2 + (x - x_G)^2}}\right)^2 = (y_N - y_G)^2$$

$$\left(\frac{r(y-y_G)}{\sqrt{(y-y_G)^2+(x-x_G)^2}}\right)^2 = (y_N-y_G)^2$$

$$\frac{r(y-y_G)}{\sqrt{(y-y_G)^2+(x-x_G)^2}} = y_N-y_G$$

$$y_N = y_G + \frac{r(y-y_G)}{\sqrt{(y-y_G)^2+(x-x_G)^2}} \tag{3.10}$$

Caso o resultado (x_N, y_N) esteja fora da região de entrada, x_N e y_N são usados como entrada (no lugar de x e y) para as Equações 3.1-3.3. As Equações 3.9 e 3.10 são inválidas apenas quando o robô se encontra exatamente no alvo. Porém, em aplicações reais, considera-se que o robô alcançou o alvo antes de alcançar a posição exata do alvo, evitando que ocorra divisão por zero ao usar estas duas equações.

A força de atração $\mathbf{F_A}$ é aplicada nos robôs a depender da sua posição em relação as regiões de entrada e seu estado. Duas equações podem ser usadas para calcular o valor de $\mathbf{F_A}$:

$$\mathbf{F_A} = K_a\mathbf{d} \tag{3.11}$$

ou:

$$\mathbf{F_A} = K_a\frac{\mathbf{d}}{\|\mathbf{d}\|} \tag{3.12}$$

onde $\mathbf{d} = ([x', y']^T - [x, y]^T)$ e $[x', y']^T$ é o destino desejado.

A Equação 3.11 é usada quando se deseja diminuir a velocidade do robô à medida que se alcança o alvo. O módulo de $\mathbf{F_A}$ diminui a medida que se aproxima de $[x', y']^T$. Já, a Equação 3.12 é usada quando se deseja usar um vetor de atração com módulo fixo. Dessa forma, garante-se que a velocidade do robô a chegar no alvo será maior que zero. Quando o destino desejado for o alvo $[x_G, y_G]^T$, qualquer uma dessas equações podem ser utilizadas.

Contudo, quando os robôs forem atraídos para algum ponto delimitador da região (isto é, $[x', y']^T = [x_N, y_N]^T$ ou $[x', y']^T = [x_W, y_W]^T$), a Equação 3.12 é mais apropriada, pois deseja-se que os robôs passem do destino desejado. Se os robôs utilizarem a Equação 3.11, suas velocidades tenderão a zero próximo do destino, devido ao amortecimento do modelo dinâmico. Dessa forma, eles não alcançarão o outro lado da região de entrada, para $[x', y']^T = [x_W, y_W]^T$, ou a região livre, para $[x', y']^T = [x_N, y_N]^T$. A força resultante $\mathbf{F}$ que atua no robô é igual a soma de $\mathbf{F_A}$ com a força repulsiva $\mathbf{F_R}$.

3.1.3 Otimização do algoritmo

Numa versão preliminar do algoritmo proposto, os robôs só iam para a região de entrada se estivessem nos estados ESPERANDO ou PRESO. Nos demais estados, eles iam de encontro ao alvo. Durante o trajeto até o alvo, se um robô percebesse outro robô com o mesmo alvo, então ele entraria no estado ESPERANDO ou PRESO, a depender do estado do robô detectado. Como a percepção é local e susceptível a erros, ocasionalmente um robô em ESPERANDO ou PRESO poderia deixar de perceber o robô próximo e entrar no estado NORMAL, e seguir em direção ao alvo. Como robôs em estado NORMAL não são obrigados a seguir para região de entrada, robôs que erroneamente não percebiam outros em sua proximidade causavam congestionamentos até voltarem a perceber e retornarem a região de entrada. Para solucionar este comportamento, optou-se por obrigar todos os robôs, não importa qual estado, a irem para região de entrada ao perceber robôs próximos. Assim, a região de saída fica livre para os robôs que deixam o alvo.

Percebeu-se também que os congestionamentos se concentravam na região próxima ao alvo, principalmente devido às forças repulsivas advindas dos outros robôs que se aproximam. Para contornar este problema, as forças repulsivas são diminuídas pela metade ao chegar na região livre e durante a saída do alvo. Assim, os robôs que estão bem próximos do alvo, diminuem a força resultante da repulsão e tendem a não se afastar muito da região de saída. Com pouca repulsão, a área ocupada pelo robô durante o trajeto de saída é menor, permitindo o proveito da área próxima ao alvo para tráfego durante a saída. Esta diminuição da força repulsiva é aplicada somente: (i) enquanto estiver na região livre, (ii) depois de alcançar o alvo e (iii) até se alcançar a distância D deste alvo. Depois de alcançada esta distância, a força repulsiva é aplicada normalmente. Com esta modificação, um novo estado é inserido na máquina de estados. A Figura 3.8 apresenta o estado $SAINDO$, que é alcançado após um robô se aproximar do alvo, isto é, alcançar a região livre. Apenas neste estado, a força repulsiva é diminuída pela metade. O robô passa do estado $SAINDO$ para o estado $NORMAL$ quando se alcança D metros do alvo anterior.

Além destas modificações, as componentes de força repulsiva perpendiculares às linhas que demarcam as regiões são diminuídas pela metade. Assim, os robôs que já estão na região de entrada têm menos chances de saírem desta região e atrapalharem os que estão saindo. Se um robô estiver dentro da região de entrada, as forças repulsivas aplicadas a ele são reduzidas pela metade caso tentem expulsar o robô da região de entrada. Esta redução é feita verificando o ângulo do vetor de força repulsiva em relação as retas l e m (Figuras 3.5 e 3.6). Considere $\mathbf{v}$ um vetor que indica um obstáculo, obtido

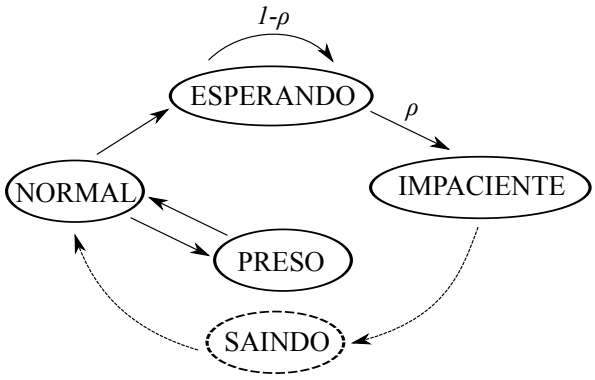


Figura 3.8. Nova máquina de estados do algoritmo.

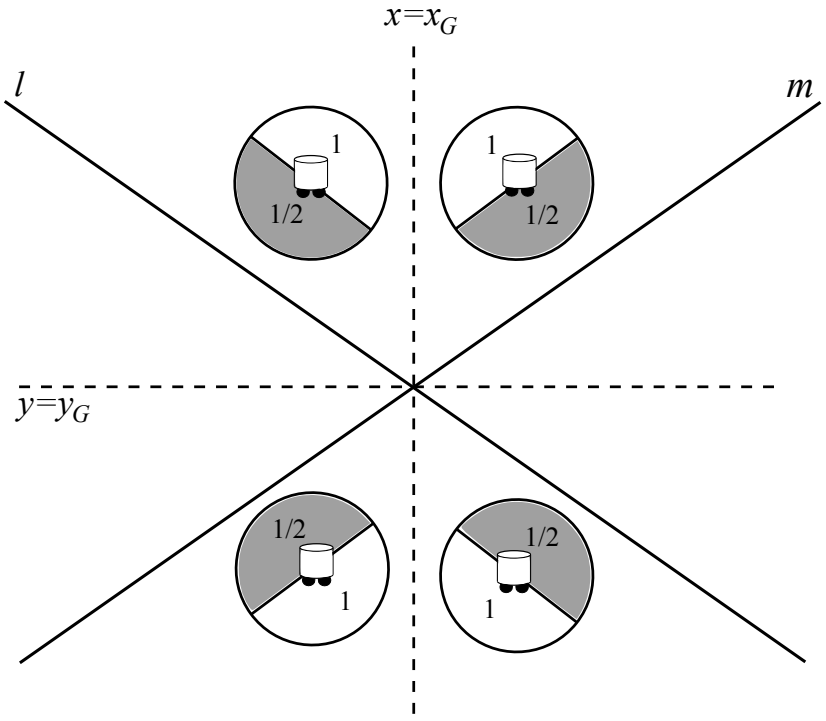


Figura 3.9. Redução das forças repulsivas para evitar que os robôs saiam da região de entrada.

do sensor do robô. A direção da força repulsiva $\mathbf{F_R}$ influenciada por $\mathbf{v}$ terá sentido inverso. A Figura 3.9 apresenta as direções de $\mathbf{F_R}$ que sofrerão redução se um robô estiver dentro da região de entrada (em cinza). Nota-se que existem quatro casos distintos a depender da sua posição em relação ao alvo.

Assuma o robô no referencial global com pose $[x, y, \theta]^T$ e ϕ o ângulo formado por $\mathbf{v}$ no referencial global. Considere as seguintes condições:

$$(90^\circ - \alpha/2 \leq \theta + \phi \leq 180^\circ) \vee (-180^\circ \leq \theta + \phi \leq -90^\circ - \alpha/2), \qquad (3.13)$$

$$(0^\circ \geq \theta + \phi \geq -90^\circ + \alpha/2) \vee (0^\circ \leq \theta + \phi \leq 90^\circ + \alpha/2). \quad (3.14)$$

onde $-180^\circ < \theta + \phi \geq 180^\circ$.

A Condição 3.13 corresponde ao ângulo estar dentro da região cinza do círculo inferior esquerdo da Figura 3.9 e a Condição 3.14, para a região cinza do círculo inferior direito.

A força repulsiva $\mathbf{F_R}$ será reduzida pela metade a depender das seguintes condições:

- se $x > x_G$ e $y > y_G$ e Condição 3.13 é verdadeira;
- se $x \leq x_G$ e $y > y_G$ e Condição 3.14 é verdadeira;
- se $x \leq x_G$ e $y \leq y_G$ e Condição 3.13 é falsa;
- se $x > x_G$ e $y \leq y_G$ e Condição 3.14 é falsa.

Esta redução não impede que os robôs saiam da região de entrada, mas reduz a quantidade de robôs na região de saída.

3.2 Algoritmo 2: Fila em espiral

O segundo algoritmo proposto consiste em coordenar os robôs para que sigam uma fila em forma de espiral de Fermat [Pickover, 2005] (Figura 3.10), cujo centro é onde se encontra o alvo. Basicamente, foi implementado um algoritmo de formação de padrão para enxames de robôs, cujo padrão geométrico é uma espiral de Fermat. A idéia é fazer uma fila que ocupe uma área compacta no ambiente, permitindo a entrada e saída de robôs na região de alvo, sem a necessidade de espera, de forma distribuída e usando apenas sensoriamento local.

3.2.1 Fundamentos sobre espirais

A equação polar que descreve esta espiral é dada por[Pickover, 2005]:

$$r^2 = a^2\theta, \text{ para } \theta \geq 0. \quad (3.15)$$

Esta equação gera dois valores para r, um positivo e outro negativo, para cada θ. Como a fila em espiral deve possuir um caminho para entrada e outro para saída, esta equação deve ser reescrita de tal forma que permita que uma única posição seja

retornada para cada ângulo θ fornecido. Além disso, é necessário que sejam descritas duas equações distintas: uma para a entrada ao alvo e outra para a saída.

Sabe-se que um ponto descrito em coordenadas polares (r, θ) pode ser reescrito para um ponto em coordenadas cartesianas usando:

$$(x, y) = (r\cos\theta, r\,\mathrm{sen}\,\theta). \tag{3.16}$$

Substituindo o valor de r da Equação 3.15 na Equação 3.16 obtém-se:

$$\begin{aligned}(x, y) &= (\pm\sqrt{a^2\theta}\cos\theta, \pm\sqrt{a^2\theta}\,\mathrm{sen}\,\theta) \\ &= (\pm a\sqrt{\theta}\cos\theta, \pm a\sqrt{\theta}\,\mathrm{sen}\,\theta).\end{aligned} \tag{3.17}$$

A equação acima pode ser modificada para gerar uma equação que seja função de θ. Duas modificações são propostas:

- O termo $\sqrt{\theta}$, é substituído por θ. Isto simplifica alguns cálculos que serão apresentados mais a frente e elimina o fato da raiz quadrada produzir valores positivos e negativos para um mesmo θ;

- Ao invés de tomar o valor positivo e negativo, resultantes de $\pm a$, considera-se apenas um deles. Assim, o sinal do fator a de escala da espiral pode ser usado para definir o sentido da fila espiral ao redor de seu centro. Em outras palavras, considerando apenas os valores $\theta \geq 0$, com a positivo define uma espiral de entrada e a negativo define uma espiral de saída.

Utilizando estas modificações, uma equação paramétrica pode ser usada para

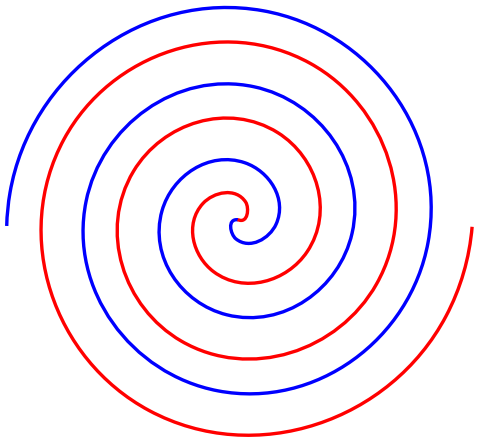


Figura 3.10. Espiral de Fermat.

descrever uma espiral com forma geométrica semelhante ao da espiral de Fermat. Tendo a variável t como parâmetro, tal que $t \geq 0$, a equação da espiral pode ser reescrita como:

$$\begin{cases} x(t) = at\cos(t); \\ y(t) = at\,\mathrm{sen}(t). \end{cases} \tag{3.18}$$

Valores de a positivos e negativos, formam as espirais vermelha e azul da Figura 3.10, respectivamente. O parâmetro t, pode ser visto como o ângulo em radianos que um ponto da espiral se encontra, mas este ângulo deve ser maior ou igual a 0 e pode assumir valores maiores que 2π. Quando $t > 2\pi$, significa que o ponto já deu no mínimo uma volta em torno do centro. A quantidade de voltas v que um ponto da espiral deu em volta do centro é dado por:

$$v = \lfloor t/2\pi \rfloor. \tag{3.19}$$

Supondo que os robôs conhecem a posição do alvo em um referencial global, considere (x_G, y_G), as coordenadas do alvo, e (x, y), a posição atual do robô.

A equação que define a fila de entrada e de saída para o ponto (x_G, y_G) é dada , respectivamente, por:

$$\begin{aligned} x(t) &= at\cos(t) + x_G; \\ y(t) &= at\,\mathrm{sen}(t) + y_G. \end{aligned} \tag{3.20}$$

$$\begin{aligned} x(t) &= -at\cos(t) + x_G; \\ y(t) &= -at\,\mathrm{sen}(t) + y_G. \end{aligned} \tag{3.21}$$

3.2.2 Obtenção de um ponto da espiral mais próximo de um ponto qualquer

Dada uma posição atual (x, y) do robô, fora da espiral, deve-se escolher um ponto dentro da espiral, (x_S, y_S), que seja o mais próximo possível. Para este algoritmo, foi estabelecido que (x_S, y_S) será o ponto da espiral mais próximo de (x, y) que intersecta a reta que cruza (x, y) e (x_G, y_G). A Figura 3.11 ilustra este conceito.

Considere $(x_S, y_S) = (at\cos(t) + x_G, at\,\mathrm{sen}(t) + y_G)$ e $d(t) = \|[x_S - x, y_S - y]\|$ a distância entre (x_S, y_S) e (x, y). Pretende-se escolher um t tal que:

$$\underset{t}{\operatorname{argmin}}\ d(t) = \underset{t}{\operatorname{argmin}}\ \|[at\cos(t) + x_G - x, at\,\mathrm{sen}(t) + y_G - y]^T\|. \tag{3.22}$$

Para computar o valor de t que satisfaz a Equação 3.22, assuma que (p, γ) é a

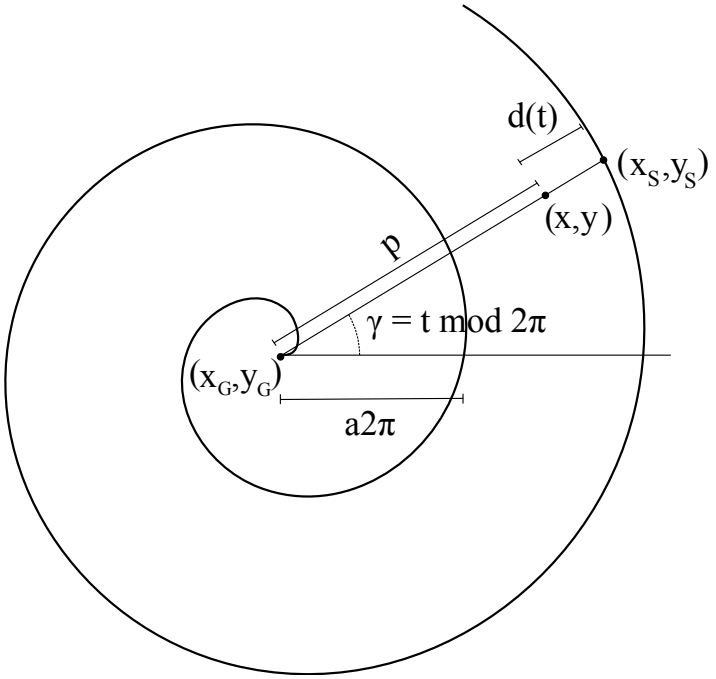


Figura 3.11. (x_S, y_S), quando $t > 2\pi$.

representação polar de $(x - x_G, y - y_G)$, isto é

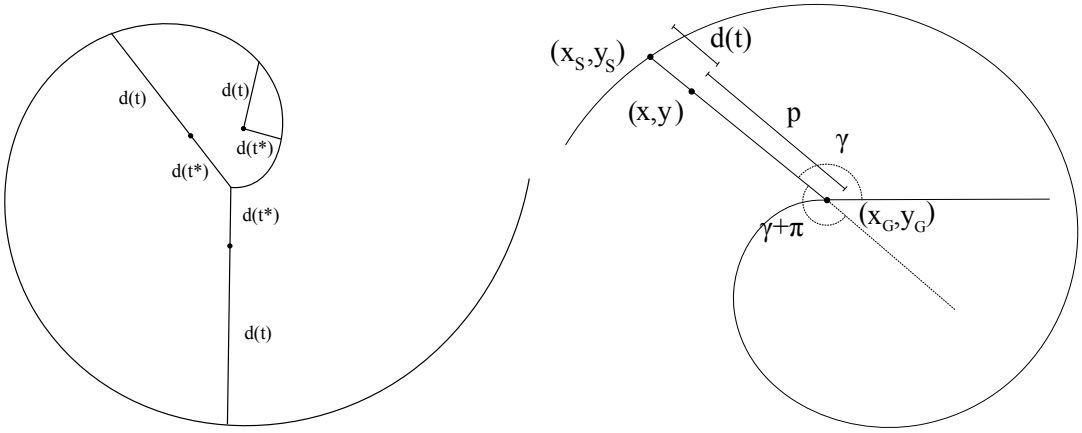
$$p = \sqrt{(x - x_G)^2 + (y - y_G)^2}; \tag{3.23}$$

$$\gamma = \arctan\left(\frac{y - y_G}{x - x_G}\right). \tag{3.24}$$

Pela Figura 3.11, sabe-se que:

$$\begin{aligned}
d(t) + p &= \|[x_S, y_S]^T - [x_G, y_G]^T\| \\
d(t) + p &= \|[x_S - x_G, y_S - y_G]^T\| \\
d(t) + p &= \|[at\cos t + x_G - x_G, at\,\mathrm{sen}\, t + y_G - y_G]^T\| \\
d(t) + p &= \|[at\cos t, at\,\mathrm{sen}\, t]^T\| \\
d(t) + p &= \sqrt{(at\cos t)^2 + (at\,\mathrm{sen}\, t)^2} \\
d(t) + p &= \sqrt{(at)^2} \\
d(t) + p &= at \\
d(t) &= at - p
\end{aligned}$$

Sabe-se que $t \equiv \gamma \mod 2\pi$. Se o robô se encontra na volta v, $a(\gamma + 2\pi(v-1)) < p \leq a(\gamma + 2\pi v)$. Desta expressão pode-se isolar v de duas maneiras:



(a) (x_S, y_S), quando $p < a2\pi$. (b) (x_S, y_S), quando $t > 2\pi$ e $a < 0$.

Figura 3.12. Alguns casos para (x_S, y_S).

$$\begin{aligned} p &\leq a(\gamma + 2\pi v) \\ \frac{p}{a} &\leq \gamma + 2\pi v \\ \frac{p}{a} - \gamma &\leq 2\pi v \\ \frac{p - a\gamma}{a} &\leq 2\pi v \\ \frac{p - a\gamma}{2\pi a} &\leq v \end{aligned} \qquad (3.25)$$

$$\begin{aligned} a(\gamma + 2\pi(v-1)) &< p \\ \gamma + 2\pi(v-1) &< \frac{p}{a} \\ 2\pi(v-1) &< \frac{p}{a} - \gamma \\ v - 1 &< \frac{p - a\gamma}{2\pi a} \\ v &< \frac{p - a\gamma}{2\pi a} + 1 \end{aligned} \qquad (3.26)$$

Sabe-se que $\lceil x \rceil = n$ se e somente se $x \leq n < x + 1$. Portanto, tomando as Equações 3.25 e 3.26, obtém-se:

$$v = \left\lceil \frac{p - a\gamma}{2\pi a} \right\rceil . \qquad (3.27)$$

Com isto, é possível obter o valor de t da seguinte maneira:

$$t = \begin{cases} \gamma + 2\pi(v-1), & \text{se } |p - a(\gamma + 2\pi(v-1))| < |p - a(\gamma + 2\pi v)|; \\ \gamma + 2\pi v, & \text{caso contrário.} \end{cases} \qquad (3.28)$$

A Equação 3.28 é uma aproximação do valor de t que satisfaz a Equação 3.22. Existem casos em que o valor de t retornado pela Equação 3.28 não minimiza $d(t)$. De fato, isto só ocorre com quando $p < a2\pi$, isto é, $v = 0$. A Figura 3.12(a) ilustra exemplos de pontos em que isto não ocorre, onde $d(t)$ significa o valor calculado com t obtido da Equação 3.28 e $d(t^*)$ indica a verdadeira menor distância.

Para encontrar o valor de t para a espiral de saída, o processo é semelhante ao

descrito até agora. Como o valor de a é negativo, o sentido da espiral é invertido. O ponto $(x_S, y_S) = (-at\cos(t) + x_G, -at\,\mathrm{sen}(t) + x_G)$ irá apontar para a direção oposta ao ângulo fornecido pelo t. Figura 3.12(b) ilustra esta idéia.

Similar ao apresentado anteriormente, quando $a < 0$, basta adicionar π a γ, obtendo:

$$v = \left\lceil \frac{p - a(\gamma + \pi)}{2\pi a} \right\rceil . \tag{3.29}$$

$$t = \begin{cases} \gamma + \pi + 2\pi(v-1), & \text{se } |p - a(\gamma + \pi + 2\pi(v-1))| < |p - a(\gamma + \pi + 2\pi v)|; \\ \gamma + \pi + 2\pi v, & \text{caso contrário.} \end{cases} \tag{3.30}$$

3.2.3 Forças que compõem a formação da fila em espiral

Depois de obtida uma forma de descobrir t, e consequentemente (x_S, y_S), a partir de um (x, y) qualquer no espaço, deve-se desenvolver uma forma de criar forças que impilam um robô numa posição (x, y) a seguir a fila espiral. Foram desenvolvidas quatro forças para cumprir este propósito:

- Força tangencial ($\mathbf{F_T}$): Força de módulo variável que tangencia a espiral. Para a espiral de entrada, ela atrai em direção ao centro e para espiral de saída, ela afasta do centro;
- Força normal a superfície da espiral ($\mathbf{F_N}$): Força de módulo variável normal a superfície da espiral. Seu objetivo é atrair um robô a superfície da espiral;
- Força tangencial inversa ($\mathbf{F_I}$): Força de módulo constante que tangencia a espiral no sentido inverso. Para a espiral de entrada, ela afasta do centro e para espiral de saída, ela atrai ao centro. Visa empurrar os robôs fora da espiral quando muitos se acumulam em um ponto da espiral;
- Força repulsiva projetada na tangencial ($\mathbf{F_P}$): Força de módulo variável que é resultante da: (i) projeção da força repulsiva entre robôs na superfície da espiral e (ii) diminuição da metade da força repulsiva que deveria ser aplicada. Seu propósito é evitar que robôs dentro da fila sejam expulsos por outro robô externo.

A força $\mathbf{F_N}$ é sempre computada enquanto o robô está próximo da espiral. Forças $\mathbf{F_T}$ e $\mathbf{F_I}$ são mutuamente excludentes. A força tangencial ocorre somente quando o robô se encontra a uma distância menor ou igual a $\epsilon = a\pi/4$ da curva. Define-se *região-ϵ* a região onde $\mathbf{F_T}$ é maior que 0. A força tangencial inversa ocorre quando a distância

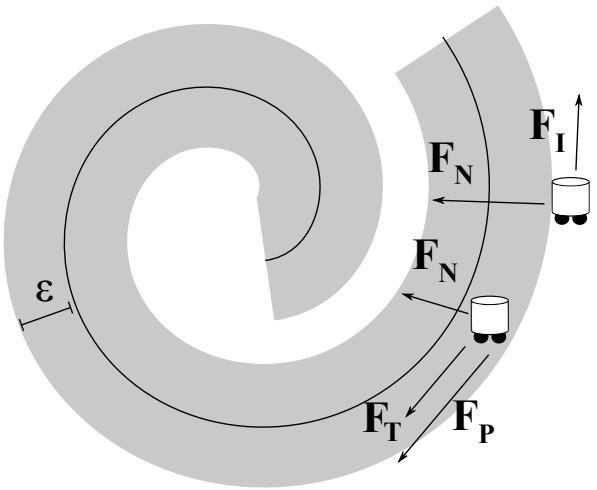


Figura 3.13. Forças aplicadas para formar a fila em espiral

entre o robô e a espiral é maior que ϵ. Figura 3.13 apresenta estas forças. A força repulsiva projetada na tangencial é induzida quando o robô se encontra na região-ϵ e existe outro robô próximo.

Considere:

- $\mathbf{p}$: vetor que vai do alvo até a posição atual do robô, isto é, $\mathbf{p} = [x, y]^T - [x_G, y_G]^T$;

- $\mathbf{S}(t)$: o vetor que vai do alvo até o ponto da espiral mais próximo do robô, para o parâmetro t computado com a Equação 3.28 ou 3.30, ou seja, $\mathbf{S}(t) = [x_S, y_S]^T - [x_G, y_G]^T = [at\cos t + x_G, at\,\mathrm{sen}\,t + y_G]^T - [x_G, y_G]^T = [at\cos t, at\,\mathrm{sen}\,t]^T$;

- $\mathbf{d}$: vetor que vai do ponto da espiral mais próximo do robô e sua posição, isto é, $\mathbf{d} = [\mathbf{d}_x, \mathbf{d}_y] = [x, y]^T - [x_S, y_S]^T = \mathbf{p} - \mathbf{S}(t)$. Quando o robô está na espiral, $\mathbf{d} = 0$.

O vetor $\mathbf{S}'(t) = \frac{d\mathbf{S}(t)}{dt}$ corresponde ao vetor tangente à espiral em relação a t. O vetor $\mathbf{S}'(t)$ é útil para indicar a direção que o crescimento da espiral aponta para um dado t. As coordenadas xy deste vetor podem ser computadas da seguinte maneira:

$$\begin{aligned} \mathbf{S}'(t) &= \frac{d\mathbf{S}(t)}{dt} \\ \mathbf{S}'(t) &= \left[\frac{d(at\cos t)}{dt}, \frac{d(at\,\mathrm{sen}\,t)}{dt}\right]^T \\ \mathbf{S}'(t) &= [a(\cos t - tsint), a(\mathrm{sen}\,t + t\cos t)]^T \end{aligned} \tag{3.31}$$

O vetor da força tangencial é computado com:

$$\mathbf{F_T} = \begin{cases} -K_T \begin{bmatrix} e^{-\mathbf{d}_x{}^2/0.01} & 0 \\ 0 & e^{-\mathbf{d}_y{}^2/0.01} \end{bmatrix} \dfrac{\mathbf{S}'(t)}{\|\mathbf{S}'(t)\|} & \text{se } \|\mathbf{d}\| \leq \epsilon \\ 0 & \text{caso contrário} \end{cases} \tag{3.32}$$

onde $K_T > 0$ e representa o maior valor para o módulo de $\mathbf{F_T}$.

A Equação 3.32 possui um sinal negativo para seguir o sentido inverso do vetor $S'(t)$. Como apenas a direção é necessária, $S'(t)$ é dividida pelo seu módulo para obter um vetor de tamanho unitário. O módulo do vetor $\mathbf{F_T}$ é afetado pela distância do robô à espiral, dada pelo vetor $\mathbf{d}$. A matriz na Equação 3.32 é igual a identidade somente quando $\mathbf{d} = 0$. Fora da linha, essa matriz assume valores menores que 1. Esta matriz tem o objetivo de fazer com que os robôs andem na direção do alvo somente se estiverem na linha espiral, forçando-os a seguir a fila espiral para alcançá-lo e deixá-lo.

A força normal é computada da seguinte forma:

$$\mathbf{F_N} = -K_N \mathbf{d} \tag{3.33}$$

onde $K_N > 0$. A força normal atrai o robô para a espiral. Quando o robô já está na espiral, seu valor é igual a zero.

A força inversa é obtida por:

$$\mathbf{F_I} = \begin{cases} K_T \dfrac{\mathbf{S}'(t)}{\|\mathbf{S}'(t)\|} & \text{se } \|\mathbf{d}\| > \epsilon \\ 0 & \text{caso contrário} \end{cases} \tag{3.34}$$

O vetor $\mathbf{F_I}$ é aplicado quando o robô não está na região-ϵ. Ele possui sentido inverso ao de $\mathbf{F_N}$ e seu módulo é sempre igual a K_T. Na Equação 3.34, o valor do módulo de $S'(t)$ cresce quadraticamente em relação ao valor de t. Por isso, optou-se por dividí-lo pelo módulo, forçando o valor de $\mathbf{F_I}$ a ser constante. Assim, à medida que o robô está longe do centro (isto é, t possui um valor grande), $\mathbf{F_I}$ não se altera, o que prejudicaria a ação das outras forças no robô.

A força repulsiva projetada na tangencial é obtida por:

$$\mathbf{F_P} = \begin{cases} proj_{\mathbf{F_I}}\mathbf{F_R} - 0.5\mathbf{F_R} & \text{se } \|\mathbf{d}\| < \epsilon \\ 0 & \text{caso contrário} \end{cases} \tag{3.35}$$

onde, $proj_{\mathbf{u}}\mathbf{v}$ é a projeção do vetor $\mathbf{v}$ em $\mathbf{u}$. O valor $0.5\mathbf{F_R}$[1] é diminuído para reduzir pela metade o efeito da força repulsiva quando o robô se encontra na região-ϵ. Sem esta redução, as linhas da espiral tendem a ser ocupadas por dois robôs lado a lado. Isto ocorre devido ao resultado de $\mathbf{F_R}$ para ambos ser alto a ponto do valor de $proj_{\mathbf{F_I}}\mathbf{F_R}$ não ser capaz de espalhá-los na linha.

O algoritmo proposto induz no robô uma força resultante $\mathbf{F} = \mathbf{F_T}+\mathbf{F_N}+\mathbf{F_I}+\mathbf{F_P}$. A implementação deste algoritmo forma uma fila em espiral sem a necessidade de troca de mensagens entre os robôs.

3.2.4 Otimização do algoritmo

Após alguns experimentos iniciais, foram feitas as seguintes modificações no algoritmo original:

- Força atrativa próximo do alvo, ao invés de seguir a tangente da espiral; e
- Os robôs ignoram a fila espiral na ausência de robôs na proximidade durante a entrada ou saída do alvo.

No algoritmo original, os robôs seguem a espiral durante todo o trajeto. No entanto, quando estão próximos do alvo, o sentido do vetor tangente à espiral varia muito em um curto espaço.

Considere $[x, y, \theta]^T$ a pose do robô num plano bidimensional. Quando ele está indo em direção ao alvo, sua orientação θ deve seguir o mesmo valor da orientação do vetor $\mathbf{F_T}$. A variação de sua orientação ($\dot{\theta}$) dentro da espiral é igual a variação da orientação de $\mathbf{F_T}$. A partir da Equação 3.31, tem-se o vetor $S'(t) = [x'(t), y'(t)]^T$, que possui orientação $\theta(t)$, dado por:

$$\theta(t) = \arctan\frac{y'(t)}{x'(t)} \tag{3.36}$$

$$\theta(t) = \arctan\frac{\operatorname{sen} t + t\cos t}{\cos t - t\operatorname{sen} t} \tag{3.37}$$

[1]Foram feitos testes com fatores iguais a $0,25$, $0,5$ e $0,75$ e os melhores resultados foram encontrados com $0,5$. Os testes realizados foram semelhantes aos apresentados no Capítulo 4, mas com menos quantidade de experimentos e comparados quanto ao tempo de execução.

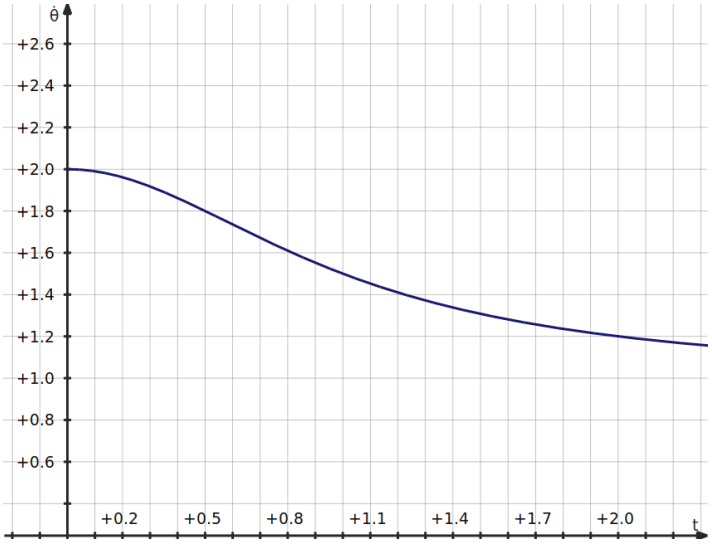


Figura 3.14. Gráfico de $t \times \dot{\theta}$.

A derivada de $\theta(t)$ em relação a t fornece o valor de $\dot{\theta}$:

$$\begin{aligned} \dot{\theta} &= \frac{d\left(\arctan\dfrac{\operatorname{sen} t + t\cos t}{\cos t - t\operatorname{sen} t}\right)}{dt} \\ \dot{\theta} &= \frac{t^2+2}{t^2+1} \end{aligned} \tag{3.38}$$

A Figura 3.14 apresenta os valores de $\dot{\theta}$ para $t \geq 0$. Nota-se que dentro de $[0, \pi/2]$ encontram-se valores altos para esta função.

Essa variação é alta apenas quando o robô está próximo do alvo. Nesta região, os robôs ficam mais lentos visto que tendem a corrigir sua orientação de acordo com a orientação imposta pelo vetor tangente a espiral. Além disso, enquanto corrigem sua orientação, os robôs que estão se aproximando do alvo induzem um campo potencial repulsivo que desvia os robôs para fora da espiral. Isto ocorre devido ao espaço próximo ao alvo ser pequeno, onde fica a curva espiral próxima do centro da espiral. O resultado disso é o surgimento de congestionamentos próximos ao alvo, atrasando a entrada e saída na região do alvo.

Visando solucionar isto, a primeira modificação imposta no algoritmo foi atrair o robô para o alvo quando $t \leq \pi/2$. Essa atração é feita usando um campo potencial atrativo, ao invés de atrair apenas pela tangente da espiral. Com esta modificação, além de diminuir o congestionamento próximo ao alvo, os robôs não perdem tempo e energia contornando a espiral, visto que a aceleração angular nesta região é maior que nas demais áreas da espiral. Para este algoritmo, a região de livre é a região em torno

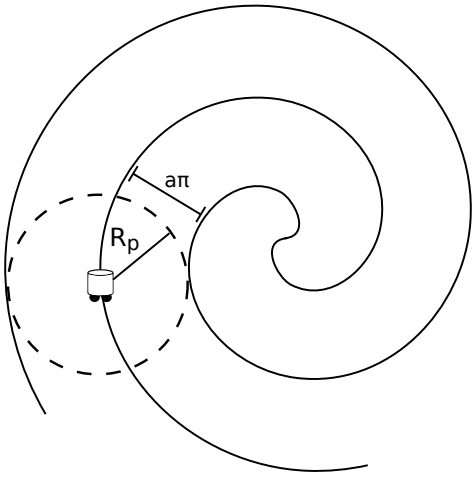


Figura 3.15. Espaçamento da espiral, dado o raio de percepção do robô.

do alvo com raio igual a $a\pi/2$.

A outra modificação consiste em permitir que os robôs possam avançar para o alvo ou para fora da espiral se não for detectado algum robô em sua proximidade. Como foi discutido, no algoritmo original todos os robôs seguem pela espiral antes de se aproximar do alvo e durante a saída da região próxima do alvo. Contudo, sabe-se que o comprimento da espiral a partir do alvo até um ponto qualquer da espiral é maior que a distância deste ponto do espaço até o alvo. Portanto, para diminuir o tempo que um robô leva para chegar ou sair ao alvo, quando um robô não perceber a presença de outro ele pode avançar em direção ao alvo ou sair da região do alvo sem necessitar contornar a espiral.

Se o espaço entre duas voltas da espiral for maior que o raio de percepção do robô, essa modificação pode atrasar o trajeto do robô. Suponha que um robô se encontre num ponto da espiral e deixe de perceber outro em sua proximidade. Com a modificação proposta, o robô deixará de seguir o contorno da espiral e irá de encontro ao alvo. No momento em que perceber outro robô na linha seguinte da espiral, ele terá que voltar a seguir pela linha da espiral. Portanto, se a largura entre duas voltas for maior que o raio de percepção, este movimento de avançar em direção ao alvo e retornar ao ponto mais próximo da espiral atrasará mais o seu percurso em direção ao alvo. Analogamente, isto pode acontecer durante a saída do alvo. A Figura 3.15 ilustra qual deve ser o espaçamento da espiral que impede este movimento.

Como explicado anteriormente, a equação da espiral fornece o parâmetro a que determina a largura da espiral. A equação polar da espiral é $r = at$, onde r é o comprimento do vetor posição e t o valor do parâmetro, em radianos. Quando $t > 2\pi$,

a distância entre cada volta i da espiral é dada por:

$$d_{i,i+1}(t) = a(2\pi(i+1)+t) - a(2\pi i + t) = 2\pi a$$

Como existem duas espirais, deseja-se a metade desta distância, pois a espiral de saída passa pelo meio. Sendo assim, o valor de a pode ser obtido de tal forma que $d_{i,i+1}(t)/2 = R_p$, onde R_p é o raio de percepção do robô. Portanto, $a = R_p/\pi$. Assim, um robô pode perceber se existe outros robôs mais a frente na fila espiral.

3.3 Algoritmo 3: "Sirene"

O terceiro algoritmo proposto consiste em repelir os outros robôs quando algum robô se aproxima do alvo. Para isso, todo robô ao chegar perto do alvo envia uma mensagem aos robôs próximos. Os robôs que não estiverem próximos o suficiente do alvo, ao receberem a mensagem, serão repelidos do alvo por um curto tempo. Os robôs que são repelidos, propagam uma mensagem para seus vizinhos, para que também sejam repelidos, assim é liberado espaço para se distanciarem do alvo.

O robô que alcança o alvo continua enviando mensagens em um período fixo de tempo. As mensagens tem o propósito de expulsar qualquer outro robô que se aproxime, ampliando a área de movimentação. Tais mensagens são enviadas pelo robô quando ele está entrando no alvo, isto é, quando está próximo do alvo antes de alcançá-lo, e durante a saída deste alvo. Esta mensagem funciona analogamente a sirene das ambulâncias, pedindo a liberação da via para passagem.

3.3.1 Troca de mensagens e força de expulsão

Os robôs enviam dois tipos de mensagens: expulsão e réplica. As mensagens de expulsão são enviadas pelos robôs que estão entrando na região próxima do alvo ou saindo do alvo. Qualquer robô que escutar essa mensagem deve seguir na direção perpendicular ao vetor do movimento do emissor da mensagem. Para isso, a mensagem possui as componentes do vetor que o robô emissor está seguindo no momento em que envia a mensagem. Considerando referencial global, o vetor da direção seguido pelo robô emissor é denotado por $\mathbf{F_M} = [F_{Mx}, F_{My}]^T$. Este vetor é enviado na mensagem de expulsão e os robôs próximos ao emissor que escutarem esta mensagem deverão seguir o vetor de expulsão $\mathbf{F_E}$ que é perpendicular a $\mathbf{F_M}$. A Figura 3.16 ilustra os vetores $\mathbf{F_E}$ e $\mathbf{F_M}$.

O sentido de $\mathbf{F_E}$ depende da posição relativa entre o robô emissor e o robô re-

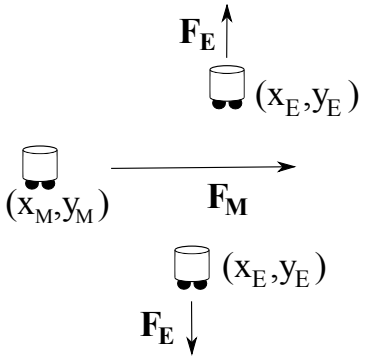


Figura 3.16. Vetores $\mathbf{F_E}$ e $\mathbf{F_M}$.

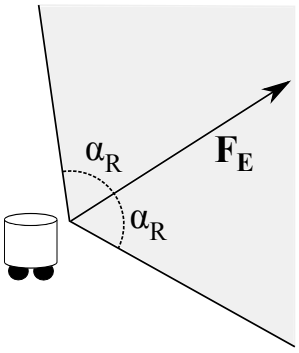


Figura 3.17. Região-α_R.

ceptor. Para isso, a mensagem enviada possui também sua posição, denotada por (x_M, y_M). Se o robô receptor estiver na posição (x_E, y_E), o valor de $\mathbf{F_E}$ é calculado por:

$$\mathbf{F_E} = \begin{cases} K_E \dfrac{[-F_{My}, F_{Mx}]^T}{\|\mathbf{F_M}\|} & \text{se } -F_{My}(x_E - x_M) + F_{Mx}(y_E - y_M) > 0, \\ -K_E \dfrac{[-F_{My}, F_{Mx}]^T}{\|\mathbf{F_M}\|} & \text{caso contrário.} \end{cases} \tag{3.39}$$

onde K_E é o módulo do vetor $\mathbf{F_E}$.

A mensagem de réplica é enviada pelos robôs que escutaram a mensagem de expulsão. O propósito da mensagem de réplica é fazer com que os robôs próximos do emissor também sigam no mesmo sentido de $\mathbf{F_E}$. Se $\mathbf{F_E}$ possui sentido para cima os robôs acima se moverão no mesmo sentido, liberando espaço para o emissor seguir $\mathbf{F_E}$.

Para limitar os robôs que deverão seguir o sentido de $\mathbf{F_E}$, define-se *região-α_R* a região, com origem na posição do robô emissor, formada pelos vetores que formam um ângulo entre $-\alpha_R$ e $+\alpha_R$ com o vetor $\mathbf{F_E}$ (Figura 3.17). Assim, apenas os robôs que lerem a mensagem de réplica que se encontram na região-α_R irão seguir $\mathbf{F_E}$. Para isso,

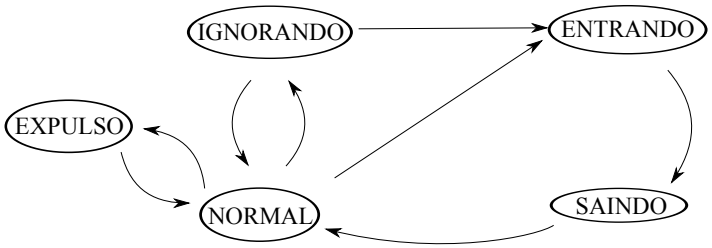


Figura 3.18. Máquina de estados para o algoritmo `Sirene`.

o robô emissor envia na mensagem sua posição (x_E, y_E) e $\mathbf{F_E} = [F_{Ex}, F_{Ey}]^T$. O robô receptor na posição (x_R, y_R) seguirá a força $\mathbf{F_E}$ apenas se:

$$\frac{\mathbf{d_{ER}} \cdot \mathbf{F_E}}{\|\mathbf{d_{ER}}\|\|\mathbf{F_E}\|} \geq \cos \alpha_R \tag{3.40}$$

onde $\mathbf{d_{ER}}$ o vetor que vai do robô emissor ao receptor, isto é, $\mathbf{d_{ER}} = [x_R - x_E, y_R - y_E]^T$. Em outras palavras, a Equação 3.40 expressa que o robô receptor apenas seguirá o vetor $\mathbf{F_E}$ se o vetor $\mathbf{d_{ER}}$ forma um ângulo menor ou igual a α_R com o vetor $\mathbf{F_E}$.

Os robôs que receberam a mensagem de réplica e estão dentro da região-α_R do emissor, também replicam a mensagem. Assim, eles se tornam emissores e apenas os robôs que estão em sua região-α_R receberão sua mensagem, repetindo o envio.

Para limitar o envio de mensagens, os robôs podem enviar mensagens a cada T_M iterações. O tempo de atuação da força de expulsão $\mathbf{F_E}$ também é limitado. Todo robô segue este vetor durante T_E ciclos. Após este tempo, o robô volta a ser atraído pelo alvo.

3.3.2 Máquina de estados

Este algoritmo foi modelado com a máquina de estados da Figura 3.18. Um robô no estado NORMAL segue em direção ao alvo. Contudo, se ele escutar uma mensagem de expulsão ou de réplica, ele passa para o estado EXPULSO, no qual ele segue a força de expulsão e reenvia a mensagem de acordo com o descrito acima. Após T_E ciclos, um robôs no estado EXPULSO, retorna ao estado NORMAL.

Se, entre ele e o alvo, não for notado outro robô, haverá uma transição do estado NORMAL para IGNORANDO. Neste estado o robô segue em direção ao alvo, ignorado mensagens, enquanto não perceber outro no caminho até o alvo. O estado IGNORANDO serve para evitar que o robô seja expulso caso não haja robôs entre ele e o alvo. Se durante o caminho até o alvo não surgir algum outro robô entre ele e o alvo, retorna-se para o estado NORMAL.

A verificação da existência de um robô no caminho até o alvo é feita de modo semelhante ao descrito sobre a réplica de mensagens. Considere o ângulo de amplitude α_A e o alvo na posição (x_G, y_G). A *região-α_A* é a região, com origem na posição (x, y) do robô no estado IGNORANDO, formada pelos vetores que formam um ângulo entre $-\alpha_A$ e α_A com o vetor $\mathbf{r} = [x_G, y_G]^T - [x, y]^T$. Considere que um sensor do robô retorna um vetor $\mathbf{p}$ toda vez que detecta algum robô em sua proximidade. Por exemplo, se for utilizado um sensor de varredura laser, $\mathbf{p}$ pode ser estimado a partir do feixe que retornou um obstáculo. Semelhante a região-α_R, um robô só é considerado no caminho até o alvo se:

$$\frac{\mathbf{p} \cdot \mathbf{r}}{\|\mathbf{p}\|\|\mathbf{r}\|} \geq \cos \alpha_A. \tag{3.41}$$

Considere que o raio de alcance para mensagem seja R_M. A região circular com centro no alvo e raio igual a $R_M/2$ é denotada região de perigo. Se um robô entrar na região de perigo, uma mensagem enviada por ele será alcançada por qualquer outro que esteja na região de perigo. Isto ocorre porque o círculo da região de perigo caberá dentro do círculo de alcance de envio de mensagens, pois seu raio é igual ao diâmetro da região de perigo.

Se um robô estiver no estado NORMAL ou IGNORANDO e alcançar a região de perigo, ele entrará no estado ENTRANDO. Se durante esta transição for percebido outro robô, uma mensagem de expulsão é enviada. Como o raio de alcance de mensagens cobre a região de perigo, a maioria dos robôs que se encontram nela serão expulsos. Apenas os que já estão muito próximos do alvo poderão não escutar a tempo, devido a um possível atraso na leitura e processamento da mensagem. Este algoritmo facilita a entrada de robôs na região de perigo, pois apenas robôs que não mudaram para o estado ENTRANDO dentro da região serão expulsos.

Após alcançado o alvo, um robô no estado ENTRANDO muda para o estado SAINDO. Neste estado, toda vez que for percebido um robô, será enviado uma mensagem de expulsão. Tanto no estado SAINDO quanto no ENTRANDO, os robôs ignoram a leitura de mensagens e seguem seu caminho.

Tabela 3.1 apresenta um resumo das funções do robô para cada estado.

Estados	**Escuta Mens. ?**	**Envia Mens. de**	**Pode ser expulso?**
NORMAL	sim	réplica	sim
EXPULSO	sim	réplica	sim
IGNORANDO	sim	réplica	não
ENTRANDO	não	expulsão	não
SAINDO	não	expulsão	não

Tabela 3.1. Estados do robô para o algoritmo `Sirene`.

Capítulo 4

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados e as dificuldades enfrentadas para alcançar a versão mais eficiente para cada algoritmo. Também serão apresentadas e justificadas as falhas, sucessos e descobertas em cada um destes experimentos, visando mostrar os avanços no desenvolvimento do algoritmo durante a pesquisa.

De um modo geral, os experimentos foram feitos de duas formas:

- Testes em simulação: o algoritmo especificado foi testado em uma plataforma de simulação de robôs de forma a avaliar diversos parâmetros. A plataforma utilizada foi o Player/Stage [Gerkey et al., 2003];
- Testes com robôs reais: o algoritmo foi implementado em robôs reais de forma a verificar sua aplicabilidade em ambiente real. Foi utilizada a infraestrutura de experimentos do VeRLab, composta de um conjunto de robôs *e-puck* (Figura 4.1) e um sistema de localização [Garcia & Chaimowicz, 2009].

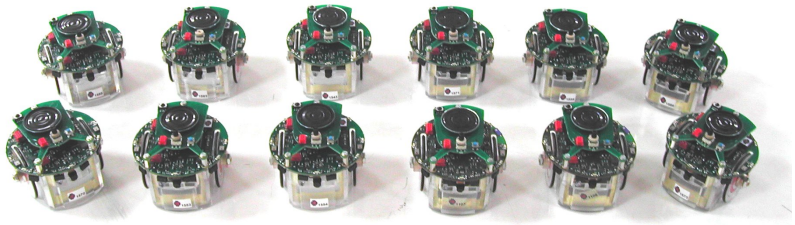


Figura 4.1. Robôs e-puck utilizados para testes reais.

Os robôs são considerados homogêneos durante as simulações, isto é, usam mesmo hardware e software. Para percepção, os robôs utilizam 360 feixes de laser com alcance de 3 m (com exceção da Seção 4.1.7) e campo de visão de 360°. O raio de alcance para envio e recepção de mensagens é igual ao usado no laser.

Em todos os algoritmos, foi utilizado um cenário inicial comum. Este cenário consiste em dispor os robôs no espaço em posições aleatórias dentro de uma coroa circular com o menor raio igual a 4,5 m e o maior raio igual a 12,5 m (com exceção da Seção 4.1.3). Como descrito no Capítulo 3, os robôs obedecem o algoritmo somente quando alcançam D metros do alvo. Em todas as simulações foi utilizado $D = 10$m. Em seguida, os robôs devem se movimentar para um alvo em comum e, após alcançá-lo, sair da região livre se deslocando para um novo alvo. Considera-se que o novo alvo estará distante e alinhado com as direções de saída do alvo original, isto é, é escolhido um ponto distante à esquerda ou à direita do alvo original.

Nas simulações, as seguintes métricas foram avaliadas:

1. *Tempo*: quantidade de iterações que o último robô levou para sair da sua posição inicial, alcançar o objetivo (a origem dos eixos) e se distanciar do objetivo à 10 m de distância;

2. *Mensagens*: quantidade total de mensagens enviadas durante o experimento;

3. *Colisões*: quantidade de colisões entre robôs no experimento.

A quantidade de iterações para medida de tempo corresponde à quantidade de vezes que o robô passa pelo laço $sense - act$ do modelo reativo utilizado [Murphy, 2000] (como discutido no início do Capítulo 3). Os algoritmos propostos podem ser implementados de tal forma que uma iteração do laço principal utilize apenas comandos que podem ser executados em tempo constante. Usando a quantidade de iterações para representação de tempo desvincula a medida de desempenho da arquitetura de máquina usada nos experimentos. Dessa forma, as simulações podem ser realizadas em computadores com diferentes frequências de processamento.

Essas três métricas foram avaliadas variando-se o número de robôs, de forma a analisar a escalabilidade dos algoritmos. Portanto, todos os gráficos (exceto na Seção 4.1.2) apresentam no eixo horizontal o número de robôs e no eixo vertical a métrica sendo avaliada. Os valores apresentados são resultantes de 40 execuções do mesmo algoritmo com as mesmas condições para cada quantidade de robô. Em vista da variação obtida nestas execuções e a natureza aleatória dos algoritmos e condições iniciais dos cenários de experimentação, os valores de tempo gasto e quantidade de mensagens apresentados correspondem à média da métrica obtida dessas experimentações. Também é apresentada uma barra que corresponde a dois desvios padrões acima e abaixo da média, pois este intervalo corresponde a aproximadamente 95% dos valores. Quando o gráfico apresentar o número de colisões, ao invés de desvio padrão, a barra representará os valores máximo e mínimo obtidos durante o experimento, visto que estes valores

costumam possuir a média próxima de zero, tornando incoerente a representação de valores negativos para número de colisões. Em alguns gráficos, o número de colisões foi igual a zero, e portanto não são apresentadas as linhas que correspondem a este resultado.

Se na apresentação dos números de colisões indicar em sua legenda um gráfico e ele não aparecer, significa que todos os seus valores são iguais a zero.

Os resultados que possuem valores próximos foram submetidos a testes de significância estatística. A maioria dos experimentos realizados compara mais de dois grupos de dados. Por este motivo, optou-se por utilizar o método *ANOVA* (*Analysis of variance*) [Barnes, 1994], para verificar se os grupos representam o mesmo resultado. O método ANOVA é uma generalização do teste t de Student para mais de duas amostras. Quando se deseja verificar se as médias de $n > 2$ grupos são iguais entre si, se for usado um teste t para cada par formado pelos n grupos, a chance de se cometer um erro do tipo I aumenta. O erro de tipo I, geralmente representado pela letra α, é o erro de rejeitar a hipótese nula quando na verdade ela era verdadeira. Nos testes apresentados, a hipótese nula é que as médias dos dados são iguais. Rejeitá-la significa dizer que essas médias não são todas iguais entre si.

Os testes estatísticos usando ANOVA são apresentados somente aos resultados que não são estatisticamente diferentes. Para entender como serão apresentados os resultados do método ANOVA para os testes realizados, considere a Figura 4.2.

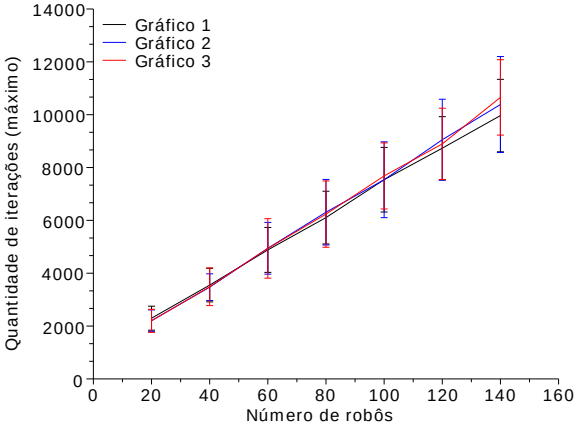


Figura 4.2. Exemplo de gráfico.

Neste trabalho, para representar o resultado do método ANOVA, uma matriz com 2 linhas e n colunas é apresentada. Para a Figura 4.2, o resultado é:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0,075 & 0,498 & 0,825 & 0,293 & 0,525 & 0,116 & 0,001 \end{pmatrix}$$

No geral, n é igual a quantidade de pontos representados no eixo x dos gráficos, que costuma ser igual ao número de robôs testados, exceto na Seção 4.1.2, onde é usado o valor de ρ. Quando o eixo horizontal representa o número de robôs, $n = 7$, como no exemplo acima, quando representa ρ, $n = 10$. Assim, a coluna i, representa o resultado do método ANOVA para os dados apresentados para a abscissa i do experimento analisado. A primeira linha é formada por valores iguais a 0 ou 1, representando respectivamente, não rejeição e rejeição da hipótese nula. Nos experimentos realizados, a hipótese nula é que as médias dos valores na abscissa i são todas iguais. A segunda linha representa o valor-p associado ao teste estatístico. O valor-p é um indicador da rejeição da hipótese nula para as amostras apresentadas. Nos testes realizados, o erro de tipo I utilizado é $\alpha = 0.05$. Quanto menor o valor-p é em relação a α, quando a hipótese nula é rejeitada, sua rejeição é mais confiável.

Os algoritmos foram primeiramente avaliados individualmente de forma a analisar a influência dos diversos parâmetros. Nesta análise individual, foram considerados robôs não holonômicos usando equações de controle baseados em [Luca & Oriolo, 1994]. Em seguida, os três algoritmos são comparados, cada um deles usando os valores dos parâmetros que melhoram seu desempenho. Nesta comparação, foram realizados testes para robôs holonômicos e não holonômicos.

As seguintes siglas serão usadas nas próximas seções: algoritmo sem coordenação (`SemCoord`), algoritmo de coordenação original (`SemRegião`), algoritmo usando coordenação e divisão de regiões (`Região`), algoritmo para fila em espiral (`Espiral`) e algoritmo sirene (`Sirene`).

Este capítulo será dividido da seguinte forma: a Seção 4.1 descreve os experimentos feitos em simulação com o Algoritmo 1, a Seção 4.2, com o Algoritmo 2 e a Seção 4.3, com o Algoritmo 3. A Seção 4.4 compara os três algoritmos e a Seção 4.5 descreve os experimentos feitos em ambientes reais com um dos algoritmos como prova de conceito.

4.1 Algoritmo 1: Regiões de entrada e saída

A análise de cada parâmetro foi feita variando-o, enquanto os outros permanecem fixos. Os seguintes parâmetros foram analisados:

- Ke e Ki: Uso de constantes diferentes para força repulsiva a depender do estado do robô;

- ρ: Probabilidade de transição para estado IMPACIENTE;

- *Raio*: Raio das áreas de perigo e livre;
- *Ciclos para impaciência* (T_P): Uso de iterações para entrar no estado IMPACIENTE.
- *Ciclos para mensagens* (T_M): Quantidade de iterações que o robô deve esperar até mandar a próxima mensagem;
- *Ciclos de espera* (T_E): Quantidade de iterações consecutivas que o robô deve esperar para perceber se não há um robô à sua frente; e
- *Alcance do sensor*: Distância máxima para percepção entre o robô e os seus vizinhos, quando estão próximo do alvo;

Alguns testes foram feitos com uma versão do algoritmo `Região` antes de aplicar as otimizações propostas na Seção 3.1.3. Isto ocorre porque tais experimentos foram realizados antes de descobrir os efeitos das modificações propostas na Seção 3.1.3. Os experimentos feitos com esta versão serão mencionados como `Região Versão 0`.

Nesta seção, os valores padrões utilizados para as variáveis estão apresentados na Tabela 4.1.

Ke	Ki	ρ	$Raio$	T_P	T_M	T_E	Alc. do sensor
0,16	0,16	0,3	$(3,5; 1,5)$	40	25	60	3,0

Tabela 4.1. Valores padrões para os experimentos com o Algoritmo `Região`.

4.1.1 Forças repulsivas

Foram realizados testes com uso de constantes diferentes para força repulsiva a depender do estado do robô. Como descrito no Capítulo 3, é utilizada uma função de campo potencial repulsivo que contém o parâmetro K que multiplica a força repulsiva entre os vizinhos. Se um robô está no estado ESPERANDO ou $PRESO$, o valor de $K = K_e$. Caso esteja no estado $NORMAL$ ou $IMPACIENTE$, $K = K_i$, onde $K_e \geq K_i$. O propósito desta mudança era garantir que os robôs que estão esperando se afastem mais entre si, dando passagem para aqueles que estão entrando nas regiões de entrada ou tentando ir para a região de saída, ou seja, robôs em estado NORMAL ou IMPACIENTE. Contudo, será mostrado mais adiante, que esta mudança não influencia no desempenho do algoritmo.

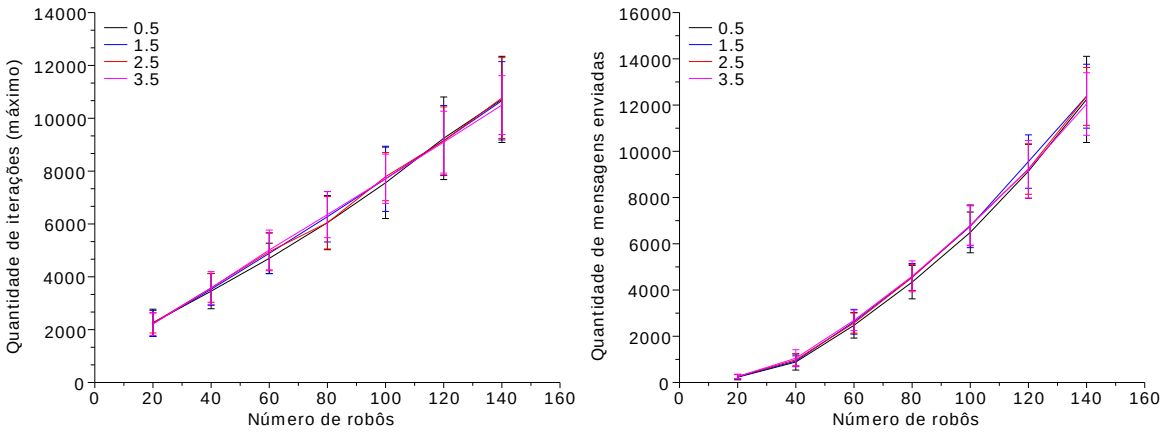
A Figura 4.3 mostra o tempo e quantidade de mensagens para o algoritmo `Região Versão 0` para diferentes valores de K_e e $K_i = 0,16$. Observa-se que não há ganho nem

de tempo nem de quantidade de mensagens. Isto se dá pelo fato dos robôs que estão fora da região de perigo não atrapalharem no tráfego do ponto em comum, que é, no geral, prejudicado pela presença de vários robôs IMPACIENTES na sua proximidade. Portanto, conclui-se que a mudança desses valores não altera no resultado final, ou seja, se os robôs se afastam mais nas regiões de espera, não há melhoria considerável no controle de tráfego.

Usando ANOVA para o tempo de execução, obtém-se o seguinte resultado:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0,784 & 0,23 & 0,001 & 0,006 & 0,307 & 0,801 & 0,383 \end{pmatrix}$$

indicando que a maioria dos valores possui médias iguais, com exceção da quantidade de robôs iguais a 60 e 80.

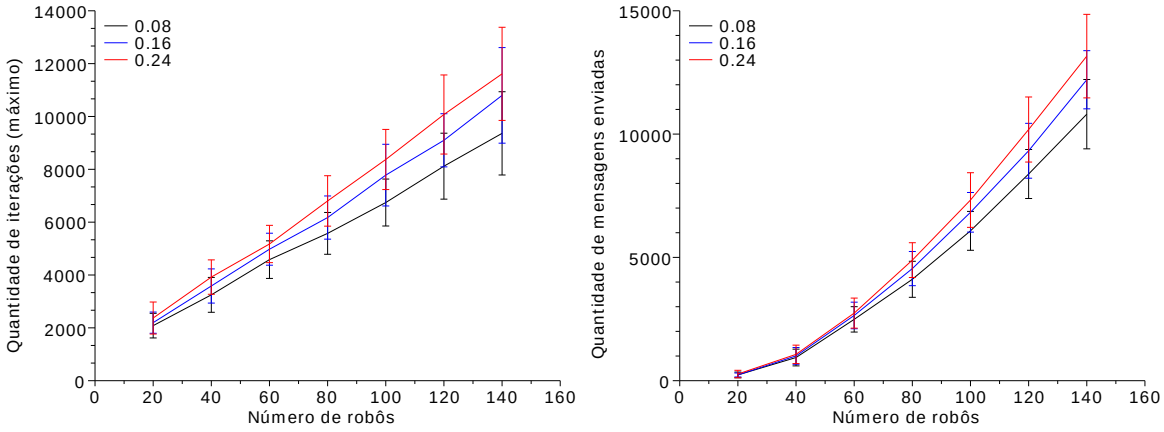


(a) Tempo de execução (b) Quantidade de mensagens

Figura 4.3. Análise de Ke para algoritmo `Região Versão 0`.

A Figura 4.4 mostra o tempo e quantidade de mensagens para o algoritmo `Região Versão 0` para diferentes valores de Ki, com $Ke = 3,0$. Observa-se que há uma relação entre o desempenho do algoritmo e o valor da constante Ki. Observa-se claramente que quanto menor este valor, mais eficiente é o algoritmo, tanto em tempo como em quantidade de mensagens. Isto ocorre porque os robôs quando estão na região de perigo devem se afastar o mínimo possível para não impedir os que estão indo em direção ao alvo se dispersarem demais, ocasionando congestionamentos na região em torno do alvo.

Contudo, existe uma desvantagem ao usar valores muito baixos de Ki. Como observa-se na Figura 4.5 que contém a quantidade de vezes que robôs se colidem durante a execução dos testes, para diferentes Ki. Nesta figura, as linhas tocam as médias



(a) Tempo de execução (b) Quantidade de mensagens

Figura 4.4. Análise de Ki para algoritmo `Região Versão 0`.

dos intervalos apresentados, que vão do menor valor ao maior valor obtidos neste experimento. Não foram usados intervalos de dois desvios padrões porque não faz sentido apresentar quantidade de colisões com valores negativos. Nota-se que à medida que o valor de Ki diminui, a quantidade de robôs que se colidem é maior. Quando Ki é muito baixo, também existe a chance de robôs se chocarem entre si e eles não se moverem mais visto que a força resultante entre eles acaba sendo igual a 0[1]. O ideal é obter um valor de Ki que seja baixo, mas não resulte em colisões.

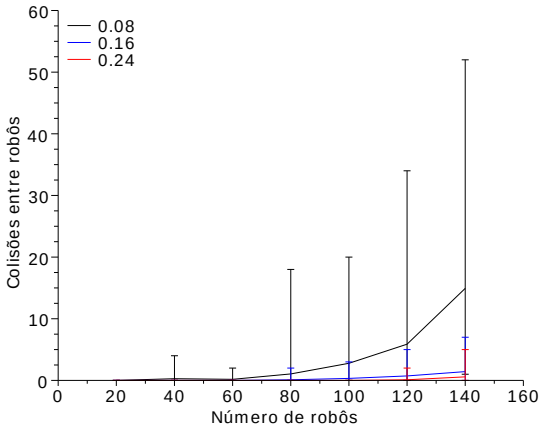
Conforme foi notado, apenas Ki influencia no desempenho, mas possui a desvantagem de valores muito baixos atrasarem o tempo de execução devido a colisões entre robôs ou ainda travarem a execução do experimento devido a mínimos locais criados entre os robôs.

4.1.2 Probabilidade de transição para IMPACIENTE

Como explicado no Capítulo 3, para o robô fazer uma transição do estado ESPERANDO para IMPACIENTE é usado um gerador de números aleatórios. O número obedece uma distribuição de probabilidade uniforme e o robô só entra no estado IMPACIENTE se o número obtido é menor que o valor de ρ. Este teste é realizado somente a cada valor fixo de iterações. Este valor é determinado pelo parâmetro *Ciclos para impaciência*. Esta limitação foi feita para evitar que a cada iteração seja feita um teste, tornando muito rápida a transição para o estado IMPACIENTE.

Foram feitos testes para ρ com o algoritmo `Região` com $T_P = 40$ e variando os

[1]Nos experimentos realizados, foram excluídos os resultados em que isso ocorre. Os valores apresentados aqui correspondem às execuções onde todos os robôs conseguiram realizar o percurso.



(a) Colisões

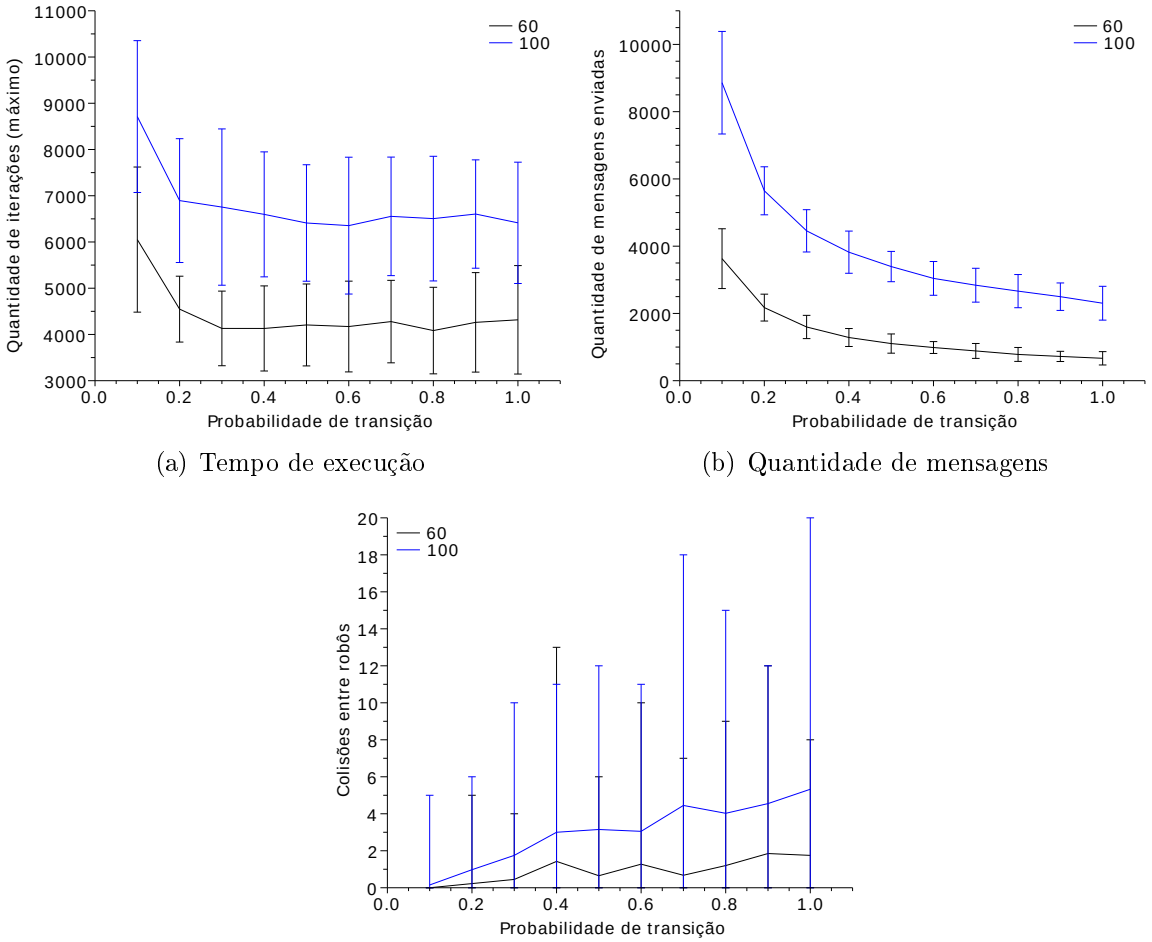
Figura 4.5. Análise de Ki para os algoritmos `Região Versão 0` - quantidade de colisões.

valores de ρ de 0,1 a 1,0, com saltos de 0,1. Foram rodados com 60 e 100 robôs. A Figura 4.6 apresenta os resultados para o algoritmo `Região`. A quantidade de tempo de execução, quantidade de mensagens são maiores usando 0,1. Isto ocorre porque eles esperam menos a medida que aumenta-se o valor de ρ, já que eles entram no estado IMPACIENTE com maior probabilidade. Contudo, as chances dos robôs chocarem entre si aumenta à medida que aumenta-se a quantidade de robôs e o valor de ρ, visto que à medida que o valor de ρ aumenta, mais robôs ocupam a região de livre.

4.1.3 Raio das regiões de perigo e livre

O teste de *Raio* serve para determinar a influência do tamanho dos raios da região de perigo e livre para o algoritmo proposto. Nos testes realizados, este parâmetro será representado por um par ordenado, onde o primeiro valor corresponde ao raio da área de perigo e o segundo, da área livre. Isto foi adotado visto que os dois devem ser aumentados, pois a área livre deve ser maior que a de perigo.

Neste experimento, foram testados os seguintes valores para o raio da região de perigo e livre: (3,5;1,5), (5,5;3,5) e (7,5;5,5). Em todos os testes realizados, os robôs iniciam em posições aleatórias fora da região de perigo. Dessa forma, no início do experimento, a coroa circular onde os robôs são dispostos aleatoriamente possui tamanho proporcional ao valor de *Raio*. Respectivamente, a coroa inicial possui menor raio igual a 4,5 m, 5,5 m e 6,5 m e maior raio igual a 12,5 m, 13,5 m e 14,5 m. Foi escolhido apenas o algoritmo `Região` para verificar este parâmetro. A Figura 4.7 apresenta os resultados deste experimento. A quantidade de tempo de execução é maior apenas para (7,5;5,5). Até 120 robôs, usando-se (3,5;1,5) e (5,5;3,5) resultam em faixas



(a) Tempo de execução

(b) Quantidade de mensagens

(c) Quantidade de colisões entre robôs

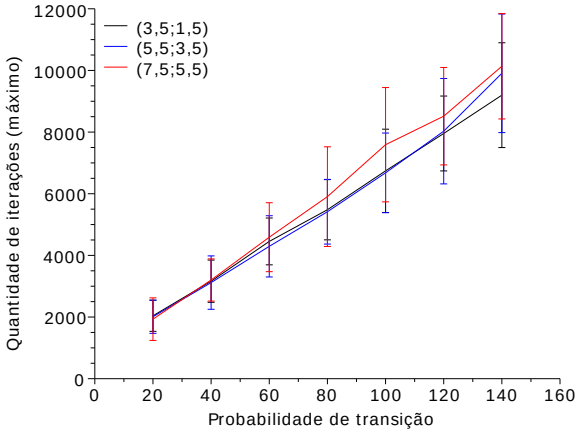
Figura 4.6. Resultados da variação de ρ para 60 e 100 robôs usando o algoritmo `Região`.

de valores semelhantes, como mostra o resultado de ANOVA para estes dois casos:

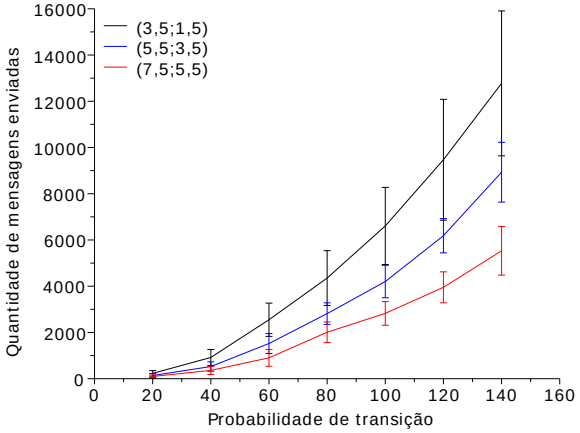
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0,602 & 0,606 & 0,111 & 0,566 & 0,664 & 0,655 & 0,001 \end{pmatrix}$$

Percebe-se que ao aumentar a área, aumenta-se a concentração de robôs nas áreas próximas da região livre, aumentando as chances de congestionamentos e quantidade de colisões. A quantidade de mensagens diminui a medida que aumenta-se a área visto que mais robôs vão para a área livre e as mensagens só são enviadas, em sua maioria, quando os robôs estão no estado PRESO.

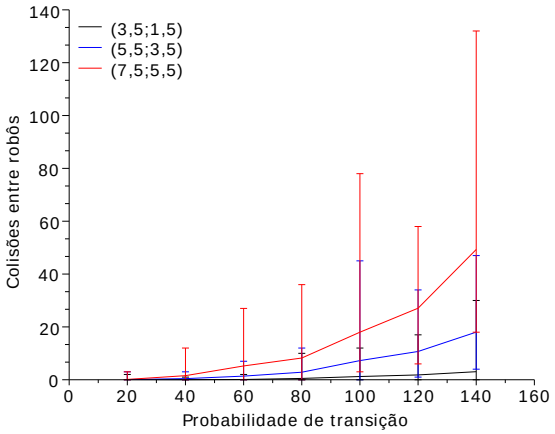
Também foi feito um teste com os valores de ρ (como descrito na Subseção 4.1.2) com os valores de *Raio* iguais a (3,5;1,5) e (6,5;4,5), para verificar se havia alguma relação do tamanho das áreas de região livre e perigo e os valores de ρ. A Figura 4.8



(a) Tempo de execução



(b) Quantidade de mensagens



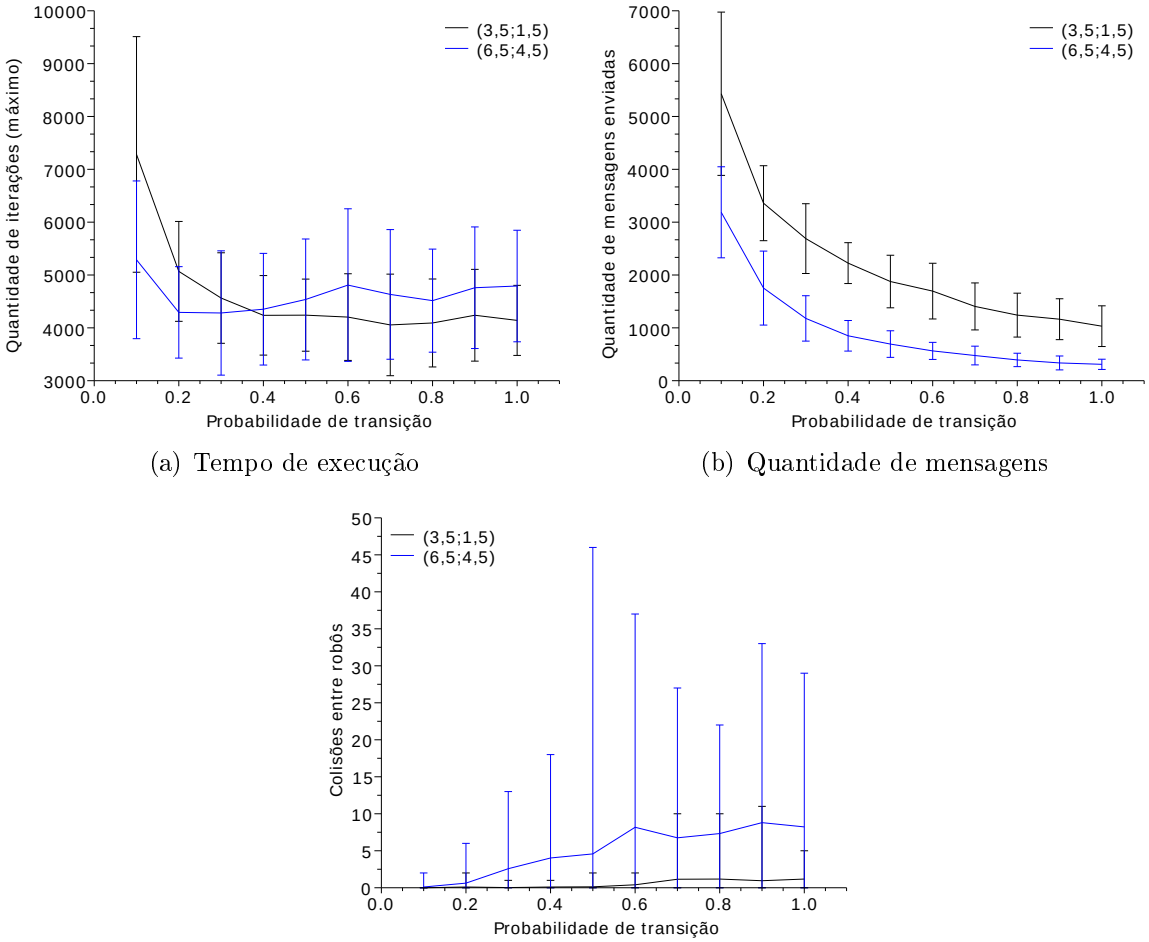
(c) Quantidade de colisões entre robôs

Figura 4.7. Resultados da variação de *Raio* no algoritmo `Região`.

mostra um comparativo usando (3,5;1,5) e (6,5;4,5) para 60 robôs e a Figura 4.9, para 100 robôs. Exceto para o tempo de execução, observa-se nessas figuras que o comportamento de ρ é o mesmo ao descrito na Subseção 4.1.2, assim como é mantida a relação de ordem entre essas métricas para os dois valores de *Raio* analisados, independente do valor de ρ. Em outras palavras, independente de ρ: para a quantidade de mensagens, o uso de (6,5;4,5) continua menor que usando (3,5;1,5) e a quantidade de colisões para (6,5;4,5) ainda é maior que usando (3,5;1,5).

Contudo, para o tempo de execução, quando usado um valor maior para *Raio*, os mínimos locais para os valores médios do tempo de execução tendem a aparecer em diferentes valores de ρ. Na Figura 4.8(a), o mínimo local encontra-se após 0,5 para (3,5;1,5), enquanto que para (6,5;4,5) o mínimo está próximo de 0,2. Observa-se o mesmo na Figura 4.9(a).

Quanto ao tempo de execução, apesar das médias nos experimentos realizados



(a) Tempo de execução

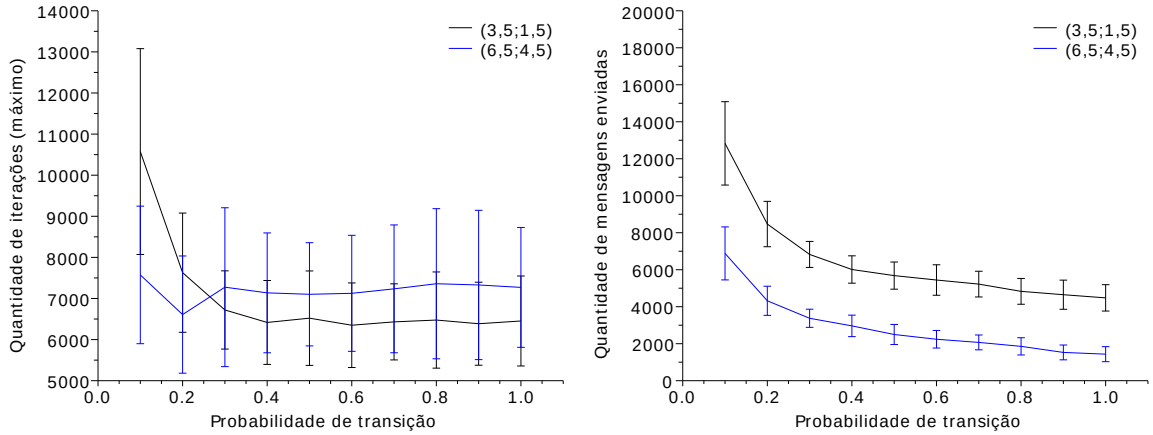
(b) Quantidade de mensagens

(c) Quantidade de colisões entre robôs

Figura 4.8. Resultados da variação de *Raio* no algoritmo `Região` com 60 robôs.

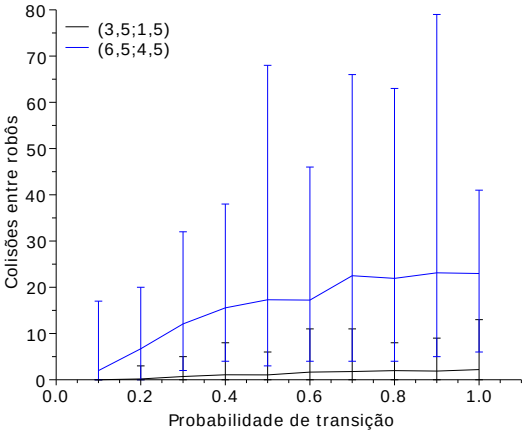
estarem em pontos que indiquem mínimos, é importante ressaltar que o desvio padrão para os experimentos sugerem que a partir de um ρ^* todas elas pertencem a um mesmo intervalo de possíveis valores. Isto pode ser verificado usando-se ANOVA considerando cada abscissa como sendo um grupo de teste. A depender do número de robôs e o raio utilizado, existe um valor de ρ^* tal que todo $\rho \geq \rho^*$ o tempo de execução médio é igual. Abaixo, para os experimentos realizados, é apresentado o menor ρ^* que rejeita a hipótese nula e o valor-p para o teste ANOVA feito com os grupos cujo valor de $\rho \geq \rho^*$:

- 60 robôs e *Raio* = (3,5;1,5): $\rho^* = 0,4$, valor-p = 0,1940245;
- 60 robôs e *Raio* = (6,5;4,5): $\rho^* = 0,5$, valor-p = 0,0836206 ;
- 100 robôs e *Raio* = (3,5;1,5): $\rho^* = 0,3$, valor-p = 0,0668617; e
- 100 robôs e *Raio* = (6,5;4,5): $\rho^* = 0,3$, valor-p = 0,7677154.



(a) Tempo de execução

(b) Quantidade de mensagens



(c) Quantidade de colisões entre robôs

Figura 4.9. Resultados da variação de *Raio* no algoritmo `Região` com 100 robôs.

Uma análise semelhante com ANOVA indica a existência de um ρ^{**}, similar a ρ^{*}, mas para a quantidade de colisões. Analisando os dados para a quantidade de colisões obtém-se:

- 60 robôs e *Raio* = (3,5;1,5): $\rho^{**} = 0,6$, valor-p = 0,2590189;
- 60 robôs e *Raio* = (6,5;4,5): $\rho^{**} = 0,5$, valor-p = 0,0785089;
- 100 robôs e *Raio* = (3,5;1,5): $\rho^{**} = 0,4$, valor-p = 0,1765221; e
- 100 robôs e *Raio* = (6,5;4,5): $\rho^{**} = 0,5$, valor-p = 0,0568585.

Foi observado nos experimentos que, quando a área de região perigo é grande, aumenta-se as chances de mais robôs no estado IMPACIENTE ocuparem-na. Com muitos robôs nessa região, mais tempo será usado para sair do alvo, devido ao congestionamento nesta região. Se os robôs ficam mais tempo no estado ESPERANDO

– quando ρ é baixo – mais rápido um robô IMPACIENTE sairá do alvo. Todavia, irá demorar mais para alcançá-lo devido ao tempo usado enquanto estava no estado ESPERANDO. Conclui-se que à medida que a região aumenta, o tempo de chegada baixo é compensado com o tempo de saída alto e vice-versa.

4.1.4 Período de espera em ciclos

Como comentado na Subseção 4.1.2, antes dos robôs entrarem no estado IMPACIENTE, eles realizam um teste com probabilidade ρ de sucesso. Este teste é realizado somente a cada valor fixo de iterações, determinado pelo parâmetro *Ciclos para impaciência* (T_P).

Como observado na Subseção 4.1.2, o valor de ρ não parece influenciar muito no tempo de execução do algoritmo. O objetivo do uso de ρ é possibilitar que os robôs esperem até a chegada do alvo, permitindo que eles possam acessar o alvo com menos congestionamentos. Neste experimento, foi usado um valor de tempo (em iterações) ao invés de fornecer uma probabilidade de transição do estado ESPERANDO para $IMPACIENTE$. Em outras palavras, após I iterações, o robô fará esta transição com $\rho = 1,0$. Nos experimentos desta seção foram usados valores de I variando de 0 a 320 com saltos de 80 iterações.

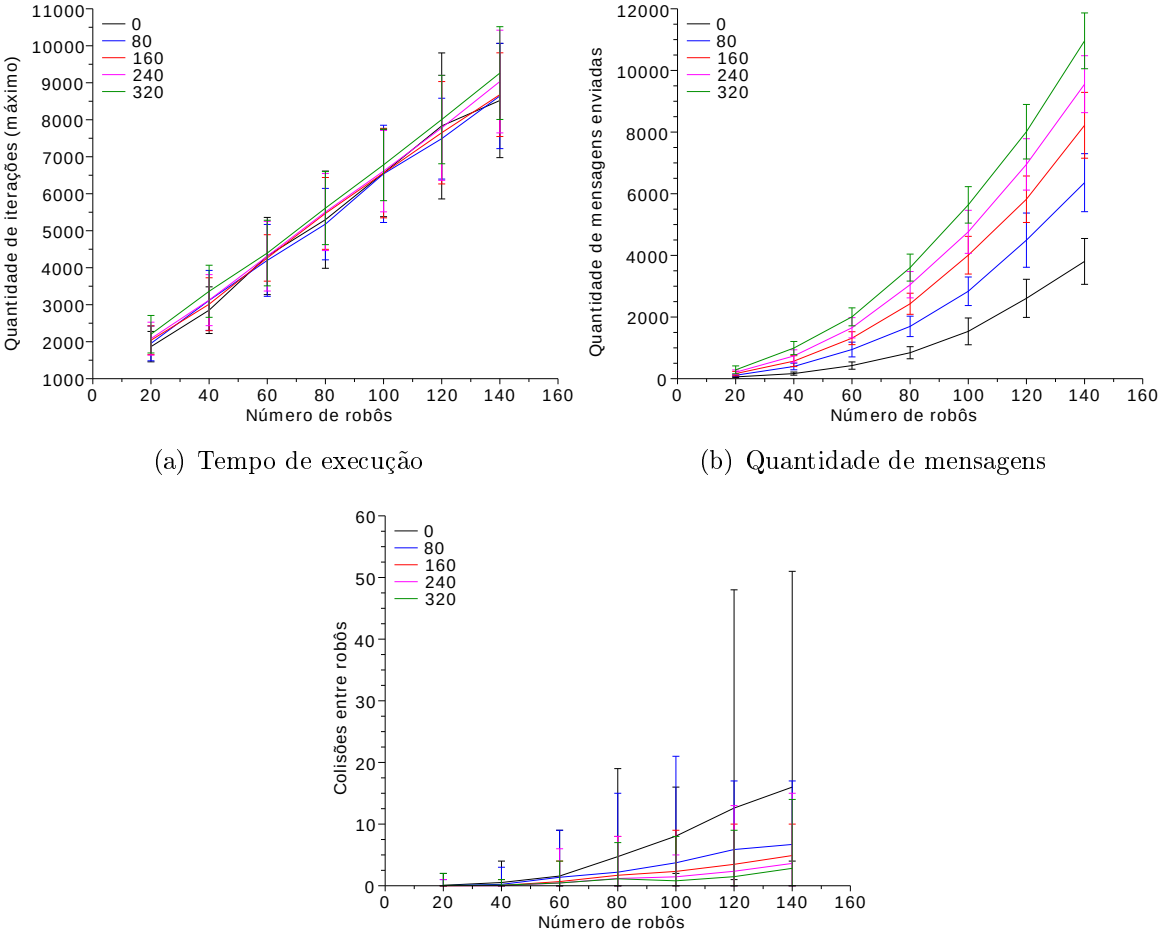
Neste experimento foram feitos testes com os algoritmos `Região`. Foram executados experimentos com *Raio* igual a (3,5; 1,5) e (5,5; 2,5). Foram escolhidos diferentes valores para os raios das regiões de perigo e livre para verificar se a quantidade de tempo a ser usada deve ou não ser proporcional à distância livre até o alvo. Isto é justificado pelo tempo de chegada até o alvo ser proporcional a esta distância, variando de acordo com o congestionamento nesta região.

A Figura 4.10 apresenta os resultados para o algoritmo `Região` com *Raio* = (3,5; 1,5). Observa-se que o tempo de execução quase não se altera entre 0 e 160. Com $0 \leq I \leq 160$, o teste ANOVA, para o tempo de execução, resulta em:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0,002 & 0,006 & 0,501 & 0,064 & 0,983 & 0,135 & 0,560 \end{pmatrix}$$

mostrando que, com exceção de 20 e 40 robôs, os resultados são semelhantes.

Dentro deste intervalo, os robôs com *Ciclos para impaciência* menor chegam mais rápido ao alvo. Contudo, quanto mais rápidos eles vão para o alvo, maior os congestionamentos nesta região, ocasionando num maior tempo de saída. O inverso ocorre quando *Ciclos para impaciência* é igual a 160. Porém, a partir de 240, o tempo de execução é menor, pois os robôs esperam mais do que deveriam atrasando o tempo de



(a) Tempo de execução

(b) Quantidade de mensagens

(c) Quantidade de colisões entre robôs

Figura 4.10. Resultados da variação de *Ciclos para impaciência* no algoritmo `Região` com *Raio* = (3,5; 1,5).

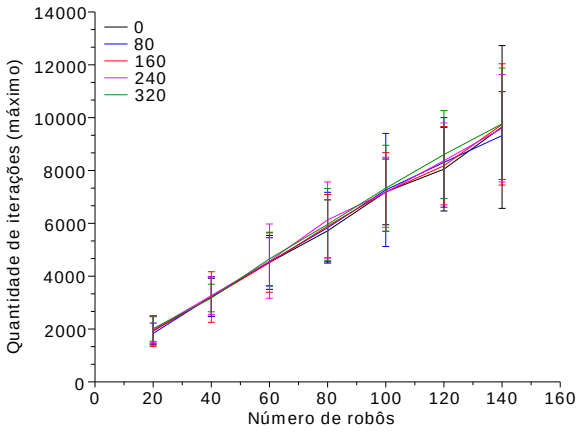
execução. Quanto à quantidade de mensagens, observa-se que os robôs enviam menos mensagens à medida que se usa um valor menor de *Ciclos para impaciência*, pois os robôs ficam menos tempo no estado ESPERANDO e consequentemente os robôs atrás ficam menos tempo no estado PRESO. Observa-se também que a quantidade de colisões é maior quando se usa apenas *Ciclos para impaciência*, considerando $\rho = 1,0$, pois mais robôs se encontram na mesma região do espaço próximo ao alvo. As forças repulsivas entre eles afetam mais seus movimentos criando colisões entre eles. Como discutido na Subseção 4.1.2, o uso de $\rho > 0,5$ implica em mais chances deles se chocarem.

A Figura 4.11 apresenta os resultados para o algoritmo `Região` com *Raio* = (5,5; 3,5). Observa-se que os resultados são semelhantes ao gerado com *Raio* = (3,5; 1,5). A única diferença é que o tempo de execução torna-se semelhante num intervalo maior: de 0 a 320. Usando este intervalo, o teste ANOVA para o tempo de execução resulta

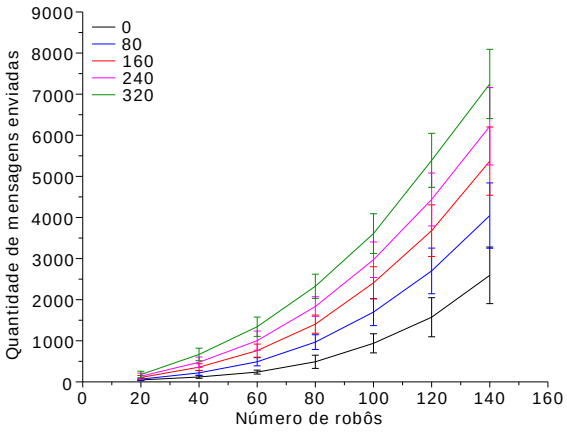
em:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0,019 & 0,871 & 0,787 & 0,069 & 0,893 & 0,028 & 0,426 \end{pmatrix}$$

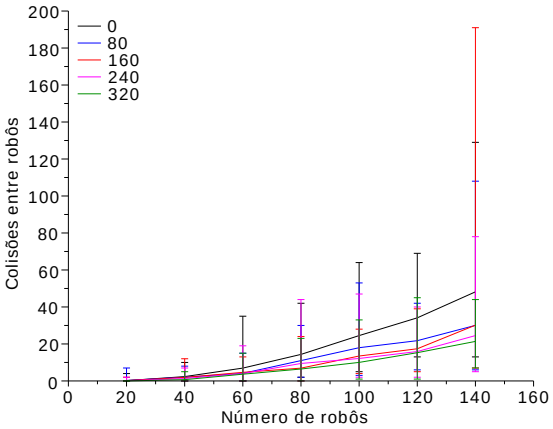
mostrando que, com exceção de 20 e 120 robôs, os resultados são semelhantes. Note que, se fosse usado um nível de confiança de 99%, também para 20 e 120 robôs os resultados seriam semelhantes. Isto sugere que quando *Raio* é maior, o valor de *Ciclos para impaciência* a partir do qual o tempo de execução começa a atrasar também é maior.



(a) Tempo de execução



(b) Quantidade de mensagens



(c) Quantidade de colisões entre robôs

Figura 4.11. Resultados da variação de TEMPO no algoritmo `Região` com RAIO = (5,5; 3,5).

4.1.5 Ciclos para mensagens

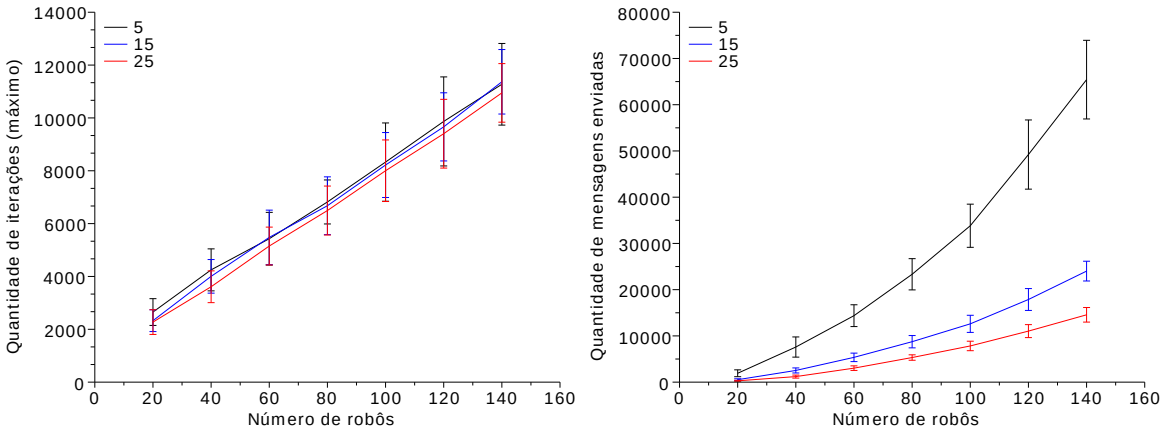
Como mencionado no Capítulo 3, *Ciclos para mensagens* (T_M) é um parâmetro útil para diminuir a quantidade de mensagens enviadas entre os robôs. O propósito de

experimentar esse parâmetro é confirmar se ele influencia o desempenho geral do algoritmo.

Foram feitos testes para T_M com valores 5, 15 e 25. A Figura 4.12 mostra a influência no tempo de execução e quantidade de mensagens, quando se usa *Ciclos de espera* igual a 90. Observa-se que, quanto ao tempo de execução, apesar de não haver uma diferença considerável para os valores apresentados, usando-se 25 *Ciclos para mensagens*, a média fica abaixo dos demais. Quanto ao número de mensagens, observa-se que usando *Ciclos para mensagens* igual a 25 há uma economia considerável.

Usando ANOVA para o tempo de execução confirma-se que as médias são diferentes (com exceção de 100 robôs) como mostra o resultado abaixo:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0,000 & 0,000 & 0,004 & 0,011 & 0,077 & 0,017 & 0,012 \end{pmatrix}$$



(a) Tempo de execução

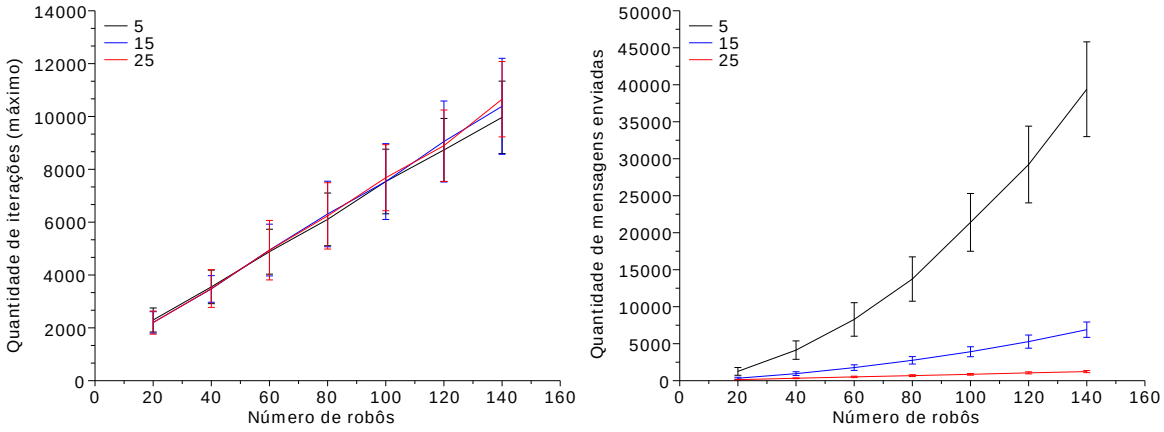
(b) Quantidade de mensagens

Figura 4.12. Análise *Ciclos para mensagens* para algoritmo `Região Versão 0`(*Ciclos de espera* = 90).

Quando se utiliza *Ciclos de espera* igual a 15, o resultado é diferente, como mostra a Figura 4.13. Nota-se agora que 25 *Ciclos para mensagens* não resulta num tempo de execução menor, na média. O resultado para o teste ANOVA neste caso é:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0,075 & 0,498 & 0,825 & 0,293 & 0,525 & 0,116 & 0,001 \end{pmatrix}$$

indicando que os resultados são iguais, com exceção do caso em que se usa 120 robôs. Isto sugere que estes dois parâmetros estão relacionados, como é discutido na Subseção 4.1.6.



(a) Tempo de execução

(b) Quantidade de mensagens

Figura 4.13. Análise *Ciclos para mensagens* para algoritmo `Região Versão 0` (*Ciclos de espera* = 15).

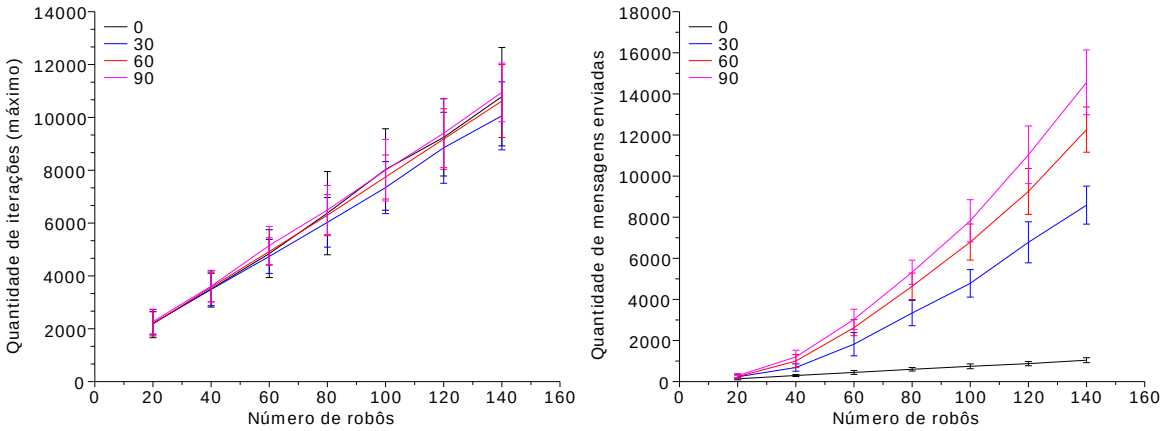
4.1.6 Ciclos de espera

Ciclos de espera (T_E) é usado para evitar que os robôs fiquem em estado PRESO por pouco tempo. Como foi explicado no Capítulo 3, os robôs entram para o estado PRESO quando percebem outro robô e este envia uma mensagem contendo seu estado. Como os outros robôs levam T_M ciclos para enviar uma próxima mensagem, durante o tempo entre o envio de uma mensagem e outra, os robôs no estado PRESO podem fazer a transição para o estado NORMAL e seguir até o alvo, como se não existisse robôs em espera. Os robôs só voltarão ao estado PRESO, quando no próximo ciclo perceberem o mesmo robô a sua frente. Sem *Ciclos de espera*, os robôs ficam trocando do estado NORMAL para PRESO a cada ciclo.

Outro motivo por usar *Ciclos de espera* está no fato do controlador utilizado [Luca & Oriolo, 1994] permitir que o robô ande em círculos ao ser aplicadas forças cujo sentido forma um ângulo diferente de zero com a orientação do robô. Quando o robô anda em círculos, seu campo de visão muda, fazendo com que o robô que estava na sua frente não seja percebido e, consequentemente, mude para o estado NORMAL. Com o uso de *Ciclos de espera*, o robô permanece no estado PRESO, mesmo que o robô mude de orientação enquanto gira.

Os gráficos da Figura 4.14 apresentam os resultados da variação de *Ciclos de espera*, usando *Ciclos para mensagens* igual a 25. Nestes gráficos percebe-se que o melhor resultado para o tempo de execução é usando um valor igual a 30. Isto era esperado, visto que este valor é próximo da quantidade de ciclos entre o envio de mensagens consecutivas. Como foi comentado anteriormente, durante o tempo de envio

entre uma mensagem e outra, os robôs podem seguir até o alvo, como se não existisse robôs em espera. Isto faz com que os robôs avancem, atrapalhando o trajeto de robôs que já estão na região de perigo ou estão saindo pela região de saída.



(a) Tempo de execução (b) Quantidade de mensagens

Figura 4.14. Análise *Ciclos de espera* para algoritmo `Região Versão 0`.

Quanto ao envio de mensagens, percebe-se que a quantidade de mensagens cresce à medida que *Ciclos de espera* cresce. Isto ocorre devido ao robô permanecer mandando mensagens enquanto está no estado PRESO. Como descrito anteriormente, o robô permanece neste estado enquanto perceber outro robô em ESPERANDO próximo a ele. Como T_E determina o tempo que o robô permanece no estado PRESO, a quantidade de mensagens enviadas será proporcional a este valor.

Quando o valor de *Ciclos de espera* é alto, os robôs ficam mais tempo esperando para avançarem na posição onde estava o robô no estado ESPERANDO à sua frente. Isto libera um pouco mais de espaço para o robô que está saindo do alvo poder passar, mas em compensação, o robô em PRESO demora para avançar. Valores muito altos de *Ciclos de espera* tendem a atrasar ainda mais o algoritmo. [2]

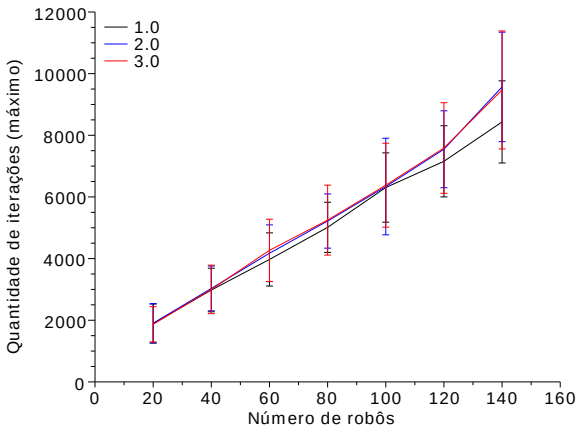
4.1.7 Área de alcance do sensor

O teste para *Alcance do sensor* foi realizado para ajustar um valor que impeça os robôs de se afastarem muito quando sentem a presença de outro robô quando estão próximos do alvo. Isto visa diminuir sua área de ação possibilitando que outros robôs possam usar mais espaço livre em áreas críticas e evitar que outros robôs se afastem mais, diminuindo o tempo para os robôs alcançarem seu objetivo. A idéia deste teste

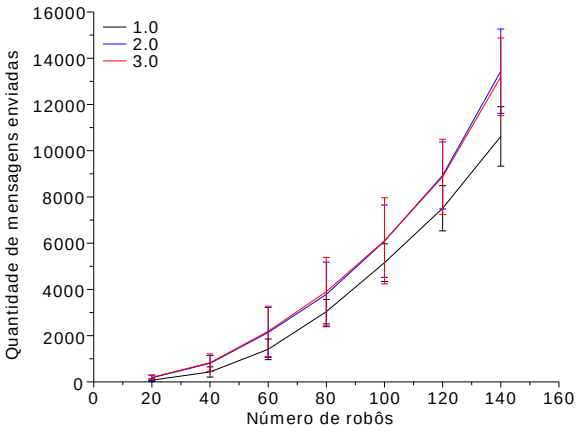
[2] Ao se utilizar um valor igual a 200, por exemplo, é maior a chance de alguns robôs ficarem parados uns na frente dos outros indefinidamente.

era verificar uma distância de percepção ideal que fizesse com que os robôs não se afastassem muito devido às forças repulsivas e não se aproximassem o bastante para se chocarem. Quando os robôs recuam pouco na presença de outros robôs, uma área menor é utilizada, dando mais espaço para que outros robôs possam alcançar ou sair da região em torno do alvo.

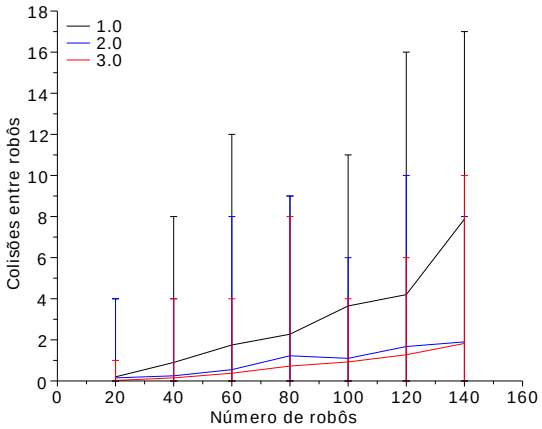
A Figura 4.15 apresenta os resultados para variações do *Alcance do sensor* para o algoritmo `Região`. Observa-se que o tempo de execução e quantidade de mensagens melhoram à medida que se diminui o valor da área de alcance de sensor, mas a quantidade de colisões entre robôs aumenta. Como o raio de percepção é diminuído, eles percebem uns aos outros quando já estão muito próximos. Isto é semelhante ao verificado na Subseção 4.1.1, quanto a influência de Ki no algoritmo.



(a) Tempo de execução

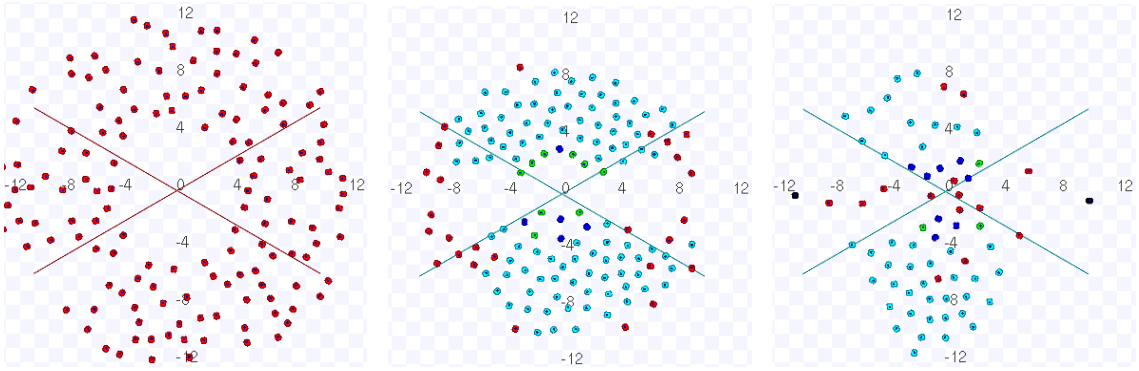


(b) Quantidade de mensagens



(c) Quantidade de colisões entre robôs

Figura 4.15. Análise do parâmetro *Percepção*.



(a) Início da execução (b) Formação das regiões de entrada e saída (c) Execução mais à frente

Figura 4.16. Imagens da execução (vídeo disponível em http://youtu.be/v3odCzK_hh4).

4.1.8 Execução do algoritmo

A Figura 4.16 apresenta algumas imagens da execução do algoritmo `Região`. Robôs em estado *NORMAL*, *ESPERANDO*, *PRESO* e *IMPACIENTE* são representados com as cores vermelha, verde, ciano e azul, respectivamente. Robôs que terminaram a execução estão representados com a cor preta. Na Figura 4.16(a), tem-se o início da execução. Algumas iterações após o início, a Figura 4.16(b) mostra quando as regiões estão começando a serem formadas. Observa-se que todos os robôs, independente de qual estado esteja, vão para a região de entrada mais próxima. Nota-se, na região próxima ao alvo, que alguns robôs estavam no estado *ESPERANDO* e passaram para o estado *IMPACIENTE*. Percebe-se também que os robôs que ainda estão na região de saída estão apontados para as retas que dividem estas regiões, indicando que eles estão se movendo para a área de entrada. A Figura 4.16(c) ilustra uma iteração mais a frente quando alguns poucos robôs ainda não terminaram de entrar na região do alvo. Nota-se, tanto nas regiões de entrada superior quanto inferior, que há dois robôs no estado NORMAL. Como não havia robôs no estado *ESPERANDO*, estes robôs, antes no estado *PRESO*, mudaram para o estado *NORMAL* e seguiram em frente. Contudo, como discutido na Seção 3.1.1, ao chegarem próximo do alvo e se detectarem, eles deverão entrar no estado *ESPERANDO*, visto que haverá outros robôs tentando alcançá-los.

4.1.9 Conclusões

A partir dos resultados apresentados, nota-se que alguns parâmetros influenciam de diferentes formas no desempenho do algoritmo. O valor de K para a força repulsiva

influencia diretamente no tempo de execução. Contudo, existe a desvantagem de aumentar a chance dos robôs se chocarem. Isto ocorre porque os robôs passam a ocupar uma menor área de espaço para se movimentar quando o valor de K é pequeno. Com isto, eles alcançam o alvo mais rapidamente. No entanto, se esta área é pequena, há uma maior chance dos robôs se chocarem durante os congestionamentos. Apesar da área ocupada pelo robô influenciar o desempenho geral do algoritmo, percebeu-se que diminuir o alcance do sensor próximo ao alvo não melhora o desempenho. Isto ocorre porque ao diminuir o alcance do sensor próximo do alvo, os robôs tendem a se colidirem mais e também mais robôs passam a ocupar a região próxima do alvo, aumentando os congestionamentos.

A probabilidade de transição para o estado IMPACIENTE ρ deve ser escolhida levando em conta o tamanho da área da região livre e de perigo. Sabe-se que: (i) a distância entre a posição onde um robô ESPERANDO se encontra na região de perigo e o alvo é aproximadamente igual ao raio da região livre e (ii) a quantidade de robôs que podem esperar na região de perigo é proporcional à área desta região. Quanto menor o valor de ρ mais rápido o robô vai em direção ao alvo. Quando um robô espera mais tempo, isto libera passagem nas regiões livre e de perigo para outros robôs alcançarem o alvo. Se um robô leva pouco tempo no estado ESPERANDO, ele chega mais rápido no alvo. Contudo, se todos os robôs fizerem isto, mais congestionamentos são gerados. Dessa forma, o valor de ρ deve se adequar ao raio da região livre, visto que este valor não deve fazer com que os robôs esperem muito, atrasando o desempenho global nem deve fazer com que os robôs avancem rápido para alcançar o alvo, causando mais congestionamentos. Quando o raio da região livre é pequeno, uma menor área é usada para tráfico dos robôs próximo ao alvo. Neste caso, o valor de ρ pode ser maior em relação ao ρ de um caso onde a região livre é maior, pois existirá uma maior área para o tráfego e, consequentemente, os robôs podem avançar mais rápido sem muitos congestionamentos.

Os experimentos realizados levantam a hipótese de que a partir de um ρ^*, o desempenho do algoritmo não se altera. Devido ao fato de existir este ρ^* nos experimentos realizados, concluiu-se que usar um $\rho = 1,0$ teria-se o mesmo desempenho de usar ρ^*. Sendo assim, levantou-se a possibilidade de não utilizar o ρ, mas sim uma quantidade de ciclos fixa. Isto seria o mesmo que utilizar $\rho = 1,0$ e o robô passaria a mudar para o estado IMPACIENTE depois de T_P ciclos, isto é, o tempo entre testes probabilísticos de mudança de estado. Assim como ocorre com ρ, o valor de T_P também depende do tamanho da região livre, pelo mesmo motivo explicado sobre a relação entre ρ e o tamanho das regiões livres e de perigo.

Os experimentos também mostram que existe uma relação entre T_M e T_E. Caso

eles sejam muito diferentes o tempo de execução é aumentado. Como foi comentado anteriormente, durante o tempo de envio entre uma mensagem e outra, os robôs podem seguir até o alvo, como se não existisse robôs em espera. Isto faz com que os robôs avancem, atrapalhando o trajeto de robôs que já estão na região de perigo ou estão saindo pela região de saída. Quando o valor de T_E é alto, os robôs ficam mais tempo esperando para avançarem na posição onde estava o robô no estado ESPERANDO à sua frente. Isto libera um pouco mais de espaço para o robô que está saindo do alvo poder passar, mas em compensação, o robô em PRESO demora para avançar.

O algoritmo apresentado necessita que estes parâmetros sejam estabelecidos antes de uma implementação real. Estes parâmetros dependem de medidas do ambiente (tamanho do alvo, tamanho do espaço livre em torno do alvo) e do robô (raio de alcance do sensor, raio de alcance do envio e recepção de mensagens, área ocupada pelo robô, fragilidade do robô - para estabelecer um K que evite colisões entre robôs). Uma melhoria a ser sugerida é a descoberta automática destes parâmetros, dadas estas medidas.

Na Seção 4.4, a versão otimizada deste algoritmo com parâmetros adequadamente escolhidos é comparada com os outros algoritmos.

4.2 Algoritmo 2: Fila em espiral

Assim como no Algoritmo `Região`, o Algoritmo `Espiral` também possui parâmetros que determinam sua eficiência, mas em menor número. Portanto, os seguintes experimentos foram realizados para verificar a influência dos parâmetros:

- Uso de valores diferentes para a constante K que multiplica a força repulsiva;
- Variação da largura da espiral a;

Nesta seção, os valores padrões utilizados para as variáveis estão apresentados na Tabela 4.2.

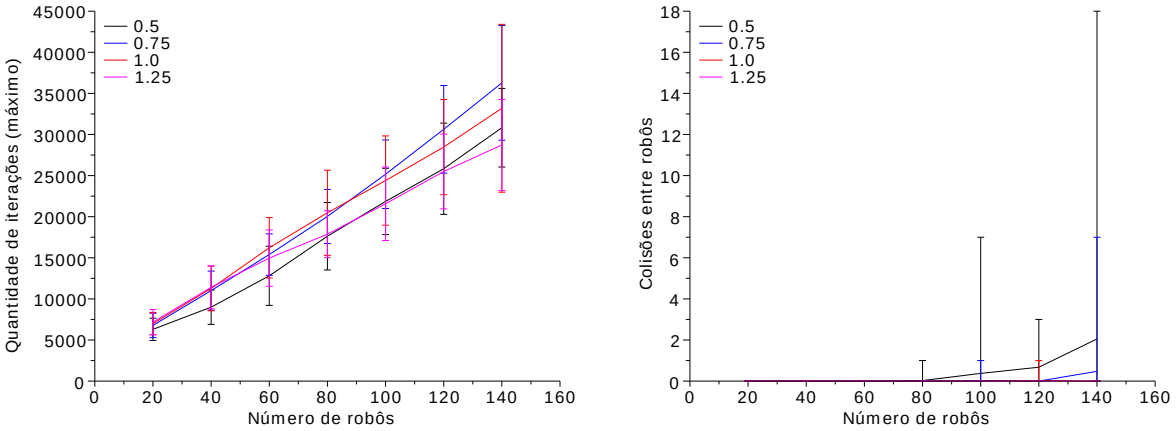
K	a
0,16	1,0

Tabela 4.2. Valores padrões para os experimentos com o Algoritmo `Espiral`.

4.2.1 Forças repulsivas

Este experimento verifica a relação entre o efeito da força repulsiva aplicada aos robôs e o algoritmo de formação da fila.

A Figura 4.17 apresenta os resultados do uso de diferentes valores para a constante K da força repulsiva e $a = 1,0$. Nos experimentos realizados, apenas foram utilizados valores para K entre 0,5 e 1,25, com saltos de 0,25.



(a) Tempo de execução (b) Colisões entre robôs

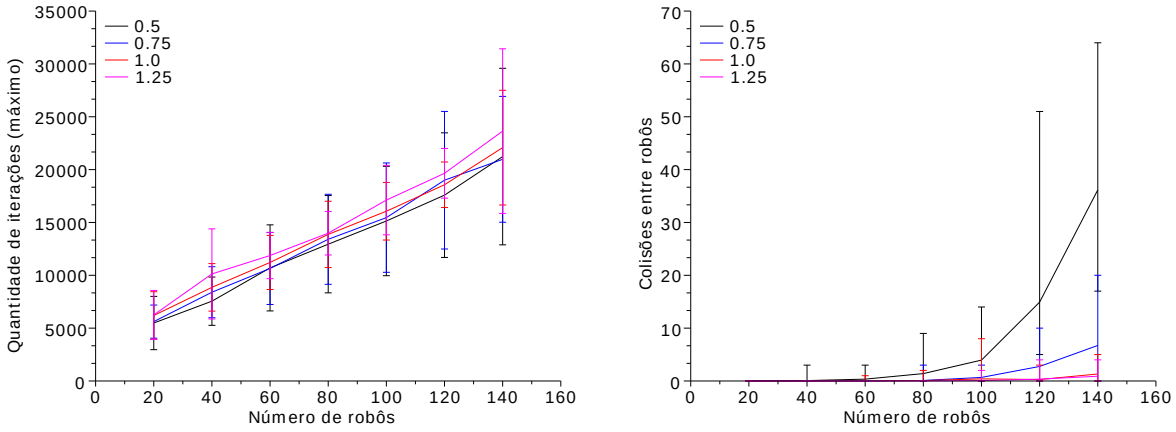
Figura 4.17. Análise de K para tempo de execução ($a = 1,0$).

Nota-se que, quanto ao tempo de execução, usando $K = 1,25$ e $K = 0,5$ obteve-se os menores valores. Diferente do que ocorre com o algoritmo `Região` (Seção 4.1.1), percebe-se que a variação de K não produz uma ordem nem direta nem inversamente proporcional na variação do tempo de execução.

Quanto a quantidade de colisões entre robôs, valores pequenos de K geram mais colisões, devido a área entre os robôs se tornar pequena. Mesmo assim, eles não foram tão grandes, comparado ao algoritmo `Região`. Pela Figura 4.17(b), nota-se que o maior valor de colisões obtido nos experimentos foi igual a 18, contudo a média para este experimento foi próxima de 2 (usando 140 robôs e $K = 0,5$).

Em outros experimentos, com valores de K diferentes dos supracitados, observou-se que robôs com valores de $K > 3,0$ tendem a se afastar mais. Durante a ordenação da fila em forma de espiral, quando se utiliza valores muito acima do ideal, os robôs tendem a ser empurrados para o final, devido a força repulsiva expulsar os robôs da região-ϵ. Com menos robôs nesta região, o tráfego dos robôs que saem do alvo é liberado. Contudo, a mesma força repulsiva que libera este tráfego, também é responsável por fazer com que os robôs sofram desvios no trajeto em forma da espiral. Um robô que se encontra no caminho de entrada para o alvo induz uma força repulsiva num robô que

se encontra no caminho de saída. Os dois se repelem, fazendo com que ambos desviem seus trajetos, atrasando o progresso global dos robôs. Outra consequência que surge com o uso de valores para $K > 3,0$ é o atraso da chegada ao alvo na região próxima ao alvo. Próximo ao alvo, os robôs que estão dentro da região livre são repelidos pelos robôs que se encontram na fila curva próxima a região livre. A quantidade de robôs nessa região em conjunto com uma maior magnitude da força repulsiva (devido ao valor alto de K) produzem uma força contrária a atração exercida pelo alvo. Isto também é um fator que atrasa o desempenho global do algoritmo. Quando os valores de K são menores que o ideal, a região-ϵ fica com mais robôs. Estes robôs avançam em direção ao alvo gerando congestionamentos próximo do alvo, visto que mais robôs continuam avançando dentro da região-ϵ.



(a) Tempo de execução (b) Colisões entre robôs

Figura 4.18. Análise de K para tempo de execução ($a = 2,0$).

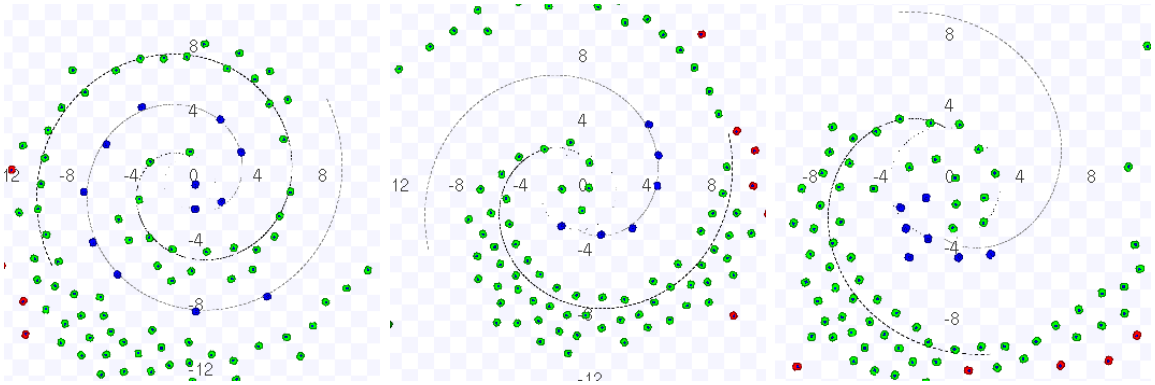
A Figura 4.18 apresenta o resultado para $a = 2,0$. Observa-se que a relação de ordem mudou, comparado ao experimento usando $a = 1,0$. Agora o resultado para $K = 1,25$ é maior que os demais. Com o ANOVA aplicado nos resultados resulta em:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0,002 & 0,000 & 0,002 & 0,046 & 0,000 & 0,001 & 0,003 \end{pmatrix}$$

Contudo, os três resultados abaixo de $K = 1,25$ são estatisticamente semelhantes, como mostra o resultado do teste ANOVA para $K \in \{0,5; 0,75; 1,0\}$:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0,009 & 0,000 & 0,260 & 0,126 & 0,182 & 0,050 & 0,307 \end{pmatrix}$$

Quanto ao número de colisões, apesar da relação entre os valores de K ser a



(a) $a = 1{,}0$ (b) $a = 1{,}5$ (c) $a = 2{,}0$

Figura 4.19. Forma da espiral para diferentes valores de a.

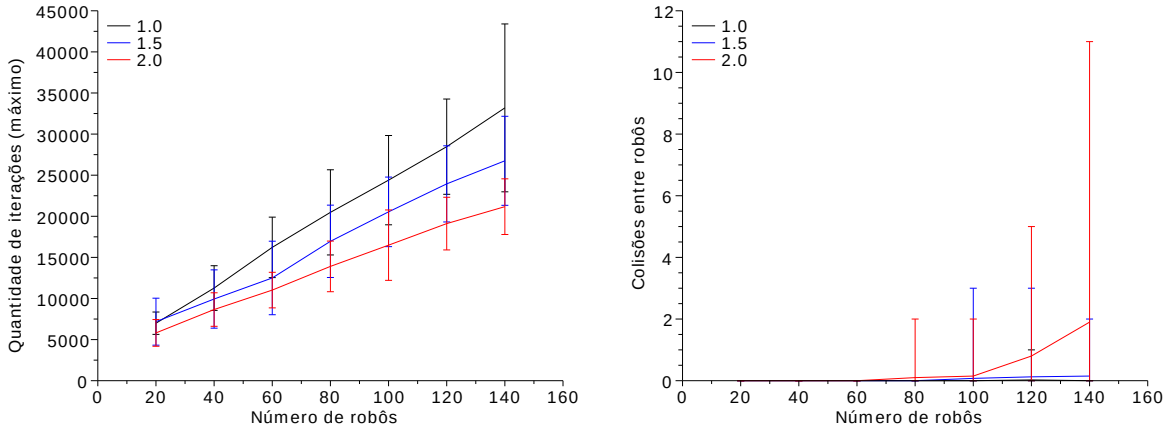
mesma que observada na Figura 4.17(b), os valores apresentados são maiores. Isto ocorre devido ao valor de a ser maior. Este efeito será discutido na Seção 4.2.2.

Percebe-se que existe uma relação entre o espaço entre as vias da espiral (dado pelo valor de a) e o valor a ser escolhido para K. Como explicado na Seção 3.2, o valor da região-ϵ é proporcional a a, assim como a região livre. Se o valor de K não for escolhido levando em conta o tamanho destas regiões, surgirão os problemas acima discutidos.

4.2.2 Largura da espiral

Foram realizados testes variando o valor de a igual a 1.0, 1.5 e 2.0. O valor de a está relacionado com a largura entre as voltas da espiral e o tamanho da região-ϵ e da região de livre. Valores pequenos de a fazem os robôs ficarem muito próximos na fila em espiral e valores grandes resultam em voltas mais distantes umas das outras.

A Figura 4.20 apresenta os resultados para este experimento. Observa-se que a quantidade de colisões é proporcional ao valor de a e o tempo de execução é inversamente proporcional. Quanto maior o valor de a, menor é a distância a ser percorrida até o centro fazendo com que os robôs levem menos tempo para chegar ao alvo. A Figura 4.19 apresenta as espirais e a disposição de 100 robôs para os três valores de a usados neste experimento. Percebe-se também que quanto maior o valor de a, a região livre – a área circular ao redor do alvo – também fica maior criando mais congestionamentos. Nota-se também que quando a é maior, devido a distância a ser percorrida ser menor, aumenta-se a concentração de robôs, resultando em mais colisões.



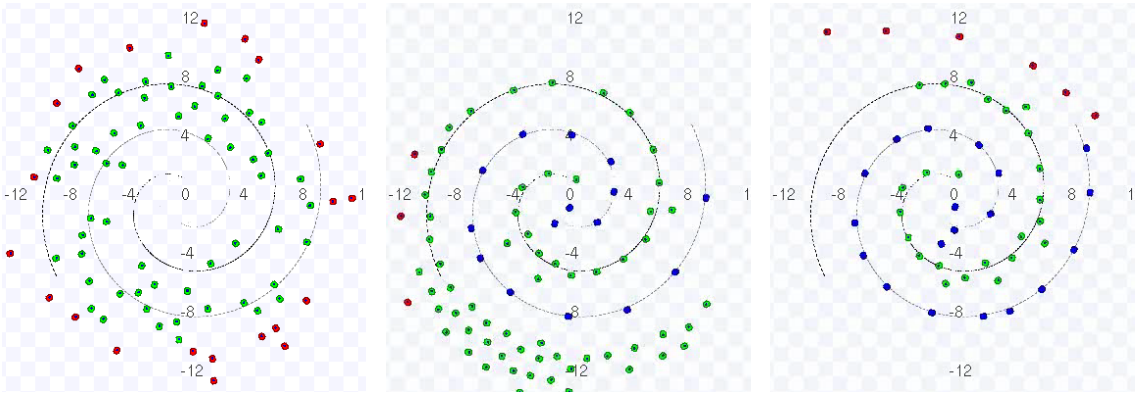
(a) Tempo de execução (b) Colisões entre robôs

Figura 4.20. Análise de a para tempo de execução.

4.2.3 Execução do algoritmo

A Figura 4.21 apresenta algumas imagens da execução do algoritmo `Espiral`. Como descrito no início deste capítulo, os robôs só obedecem o algoritmo após alcançar no mínimo 10 metros de distância do alvo. Sendo assim, Robôs que ainda não estão obedecendo o algoritmo, são representados pela cor vermelha. Robôs que estão tentando chegar ao alvo pela espiral são representados pela cor verde e os que já alcançaram o alvo e estão saindo pela espiral, pela cor azul.

Na Figura 4.21(a), tem-se o início da execução. Apenas os robôs que estão a no máximo 10 m do alvo começam a seguir a espiral. Os outros robôs são atraídos ao alvo usando um campo potencial atrativo. Quando estes alcançam 10 m do alvo, passam a tentar seguir a espiral. Algumas iterações após o início, a Figura 4.21(b) mostra a fila em forma de espiral sendo formada. Após começarem a tentar seguir a espiral, mesmo que eles se distanciem mais de 10 m, eles permanecem tentando seguir a fila espiral até alcançar o alvo. Nota-se que alguns robôs, no canto esquerdo da espiral, ainda não estão tentando seguir a espiral (em vermelho). Isto ocorre porque ainda não conseguiram alcançar o mínimo de 10 metros, pois estão sob efeito da repulsão dos que já estão na espiral. Observa-se também que a maioria dos robôs está no final da fila espiral. Isto ocorre devido a influência da força tangencial inversa (F_I), discutida na Seção 3.2. A Figura 4.21(c) apresenta um estágio próximo do final do experimento. Percebe-se que os robôs que saíram do alvo e se distanciaram de 10 m do alvo anterior seguem seu caminho para próximos alvos sem obedecer o algoritmo `Espiral`.



(a) Início da execução (b) Formação da espiral (c) Execução mais à frente

Figura 4.21. Imagens da execução (vídeo disponível em http://youtu.be/3q5jxqUppUg).

4.2.4 Conclusões

Nos testes realizados, percebeu-se que o valor de K para a força repulsiva não influencia de modo proporcional, como ocorre com o algoritmo `Região`. Contudo, valores muito acima ou muito abaixo de um valor ideal atrasam o tempo de execução do algoritmo. Também notou-se que este valor ideal depende do valor de a utilizado. Isto ocorre porque quando se utiliza valores altos para K, os robôs tendem a ser empurrados para o final. Contudo, a mesma força repulsiva que libera este tráfego, também é responsável por fazer com que os robôs sofram desvios no trajeto em forma da espiral. Outra consequência que surge com o uso de valores altos para K é o atraso da chegada ao alvo na região próxima ao alvo. A quantidade de robôs na região livre em conjunto com uma maior magnitude da força repulsiva (devido ao valor alto de K) produzem uma força contrária a atração exercida pelo alvo. Quando os valores de K são menores que o ideal, a região-ϵ fica com mais robôs. Estes robôs avançam em direção ao alvo gerando congestionamentos próximo do alvo, visto que mais robôs continuam avançando dentro da região-ϵ.

Por sua vez, o valor de a influencia de modo inversamente proporcional o tempo de execução. Assim como observado para os parâmetros do algoritmo `Região`, a medida que o tempo de execução é melhorado, a chance dos robôs se chocarem aumenta. Isto ocorre porque o valor de a está relacionado com a largura entre as voltas da espiral e o tamanho da região-ϵ e da região de livre. Quanto maior o valor de a, menor é a distância a ser percorrida até o centro fazendo com que os robôs levem menos tempo para chegar ao alvo e aumenta-se a concentração de robôs, resultando em mais colisões.

Este algoritmo, possui como vantagens não trocar mensagens e possuir poucos

parâmetros a serem configurados. Contudo, como será visto na Seção 4.4, o tempo de execução deste algoritmo não é adequado para controle de congestionamento.

4.3 Algoritmo 3: Sirene

Seguindo o método de experimentação dos algoritmos anteriores, foram testados os seguintes parâmetros para o Algoritmo `Sirene`:

- *Ciclos para mensagens* (T_M): Quantidade de iterações que o robô deve esperar até mandar a próxima mensagem;
- *Ciclos para força de expulsão* (T_E): Quantidade de iterações que o robô deve seguir a força de expulsão;
- Ângulo da região-α_R (α_R): Ângulo de percepção para réplica de mensagem;
- Ângulo da região-α_A (α_A) Ângulo de percepção para ignorar expulsão.

Nesta seção, os valores padrões utilizados para as variáveis estão apresentados na Tabela 4.3.

T_M	T_E	α_R	α_A
25	25	90°	90°

Tabela 4.3. Valores padrões para os experimentos com o Algoritmo `Sirene`.

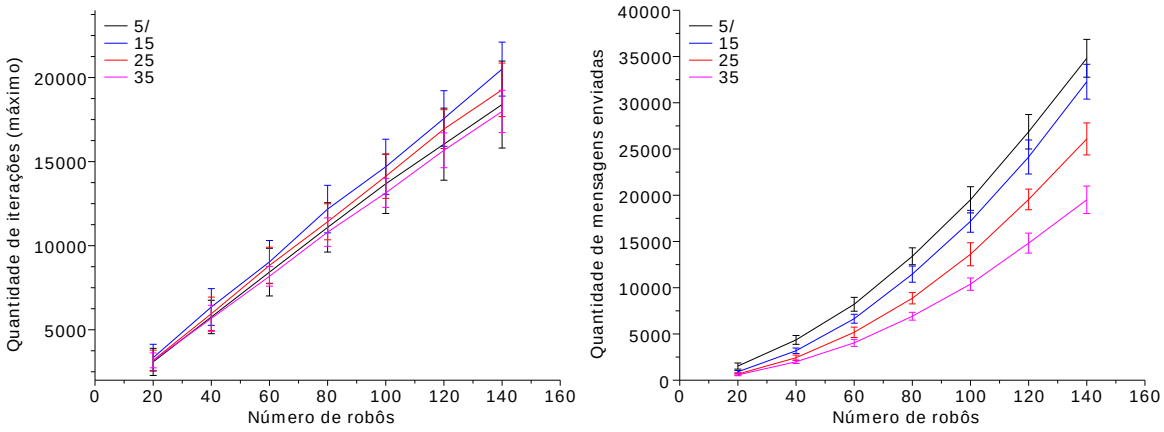
4.3.1 Ciclos para mensagens

Semelhante ao algoritmo `Região`, *Ciclos para mensagens* (T_M) é um parâmetro útil para diminuir a quantidade de mensagens enviadas entre os robôs.

Com $T_E = 25$, foram feitos testes para T_M com valores entre 5 e 35 em saltos de 5. Figura 4.22 mostra a influência no tempo de execução e quantidade de mensagens. Observa-se que o tempo de execução possui diferentes valores para cada valor de T_M, mas não segue uma relação inversa ou diretamente proporcional ao valor de T_M. Assim como ocorre com o tempo de execução, a quantidade de mensagens utilizada é menor quando se usa $T_M = 35$.

Como esperado, o número de mensagens é inversamente proporcional ao valor de T_M. O que realmente faz os robôs abrirem espaço para os demais é o tempo de atuação da força de expulsão. Como $T_E = 25$, quando $T_M < 25$, o robô irá escutar as mensagens posteriores, mas não irá obedecer à força de expulsão enviada visto que ele

ainda está atuando a mensagem anterior. Portanto, qualquer mensagem de expulsão enviada dentro dos 25 ciclos de atuação da força de expulsão será ignorada. Contudo, ele pode enviar novas mensagens de réplica enquanto está sendo empurrado pela força de expulsão.



(a) Tempo de execução (b) Quantidade de mensagens

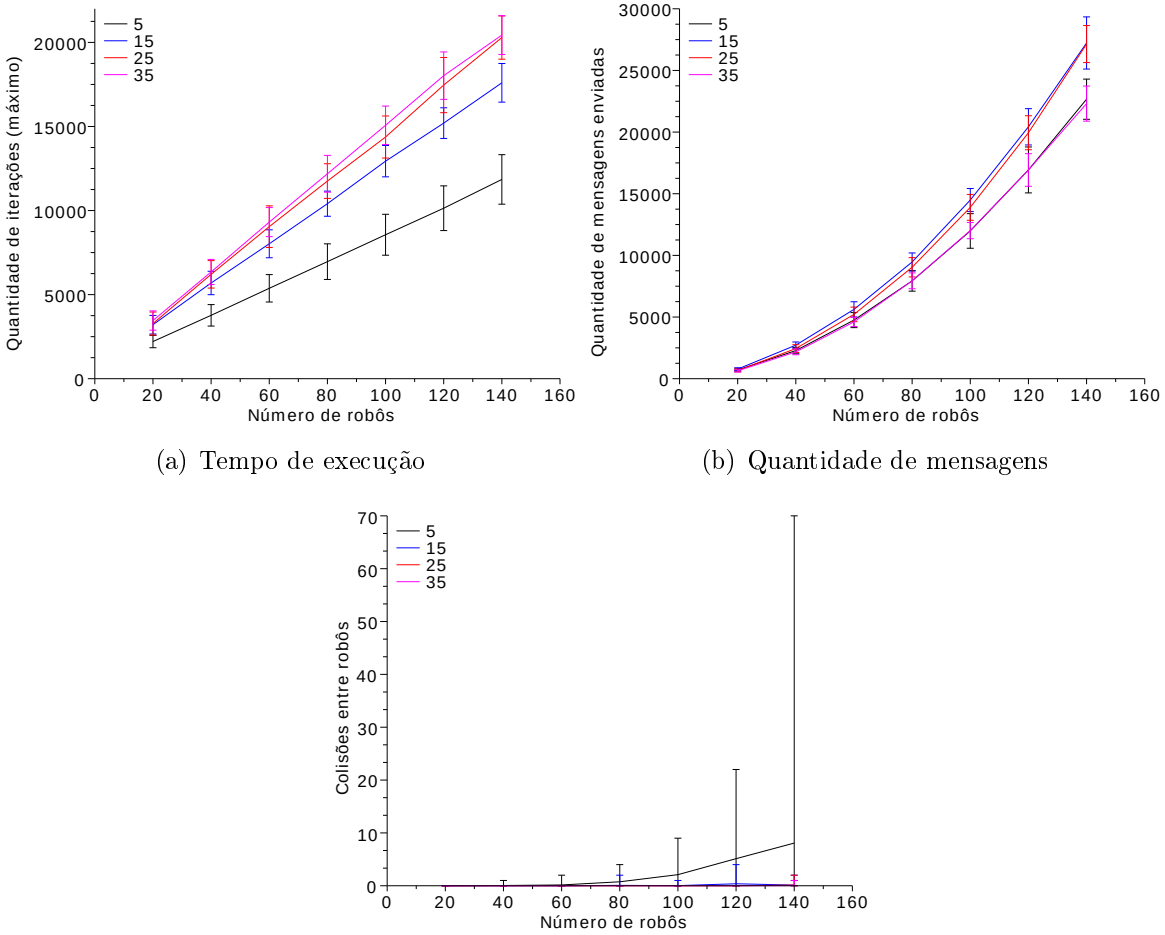
Figura 4.22. Análise de *Ciclos para mensagens* para algoritmo `Sirene`.

4.3.2 Ciclos para força de expulsão

O *ciclo para força de expulsão* (T_E) é um valor usado para limitar o tempo em que o robô é expulso da frente de um robô que emite a mensagem de expulsão.

Foram experimentados valores de T_E iguais a 5, 15, 25 e 35. A Figura 4.23 apresenta o resultado deste experimento quanto ao tempo de execução, quantidade de mensagens e colisões. Observa-se que o tempo de execução cai a medida que se diminui o valor de T_E. A quantidade de mensagens não segue proporcionalmente em relação aos valores de T_E, pois os valores mais baixos foram quando T_E é igual a 5 e a 35. A quantidade de colisões durante o experimento foi inversamente proporcional ao valor de T_E.

Quando os robôs utilizam pouco tempo para serem expulsos, eles tendem a não se afastar o bastante do alvo. A dispersão dos robôs no ambiente é inversamente proporcional ao valor de T_E. Sendo assim, se eles estão mais próximos do alvo, eles levam menos tempo para alcançá-lo, diminuindo o tempo de execução. Contudo, como eles se concentram mais na região em torno do alvo, a chance deles se chocarem é maior.



(a) Tempo de execução

(b) Quantidade de mensagens

(c) Colisões

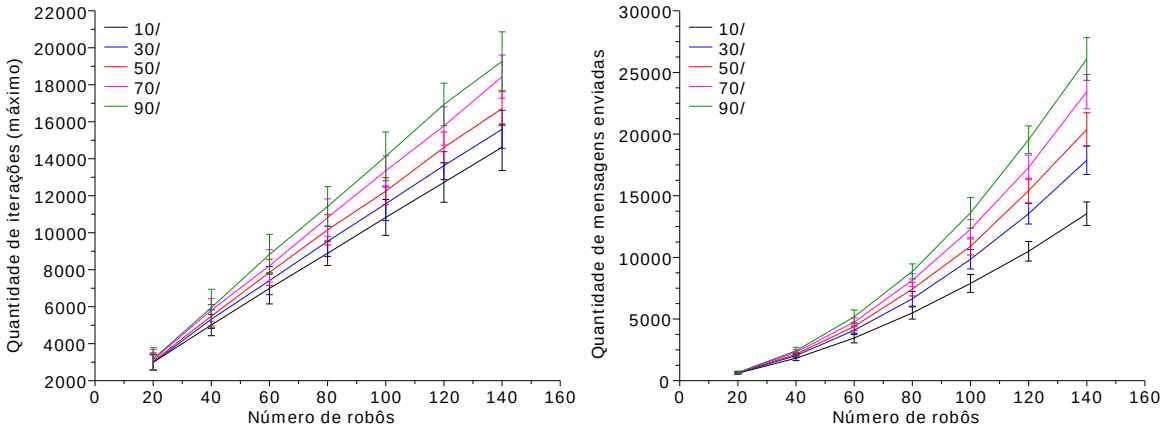
Figura 4.23. Análise de T_E para algoritmo `Sirene`.

4.3.3 Ângulo de percepção para réplica de mensagem

O *ângulo da região*-α_R, denotado pelo ângulo α_R, indica o ângulo de percepção para réplica de mensagem. Os robôs no sentido do vetor da força de expulsão, que não estão dentro da região-α_R, não obedecem a mensagem de expulsão (como explicado na Seção 3.3).

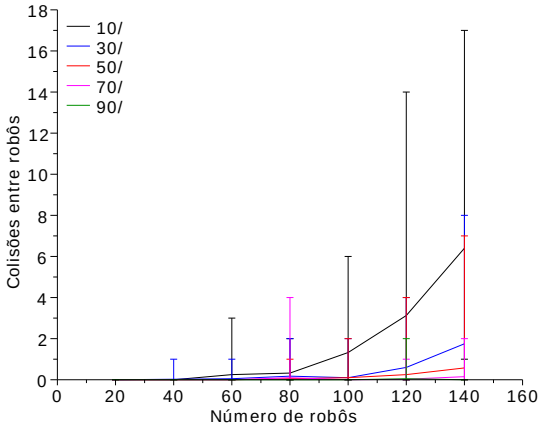
Foram realizados experimentos com α_R de 10 a 90 com saltos de 10 graus. A Figura 4.24 apresenta o resultado deste experimento quanto ao tempo de execução, quantidade de mensagens e colisões. Observa-se que o tempo de execução e quantidade de mensagens são proporcionais ao valor de α_R. Já, a quantidade de colisões é inversamente proporcional.

Quando o ângulo de réplica é pequeno, menos robôs recebem a mensagem e consequentemente poucos farão a sua réplica. Como poucos robôs obedecem a mensagem,



(a) Tempo de execução

(b) Quantidade de mensagens



(c) Colisões

Figura 4.24. Análise de α_R para algoritmo `Sirene`.

uma maior quantidade de robôs se concentram na região do alvo. Por sua vez, mais robôs na mesma região aumentam as chances de colisões entre eles.

4.3.4 Ângulo de percepção para ignorar expulsão

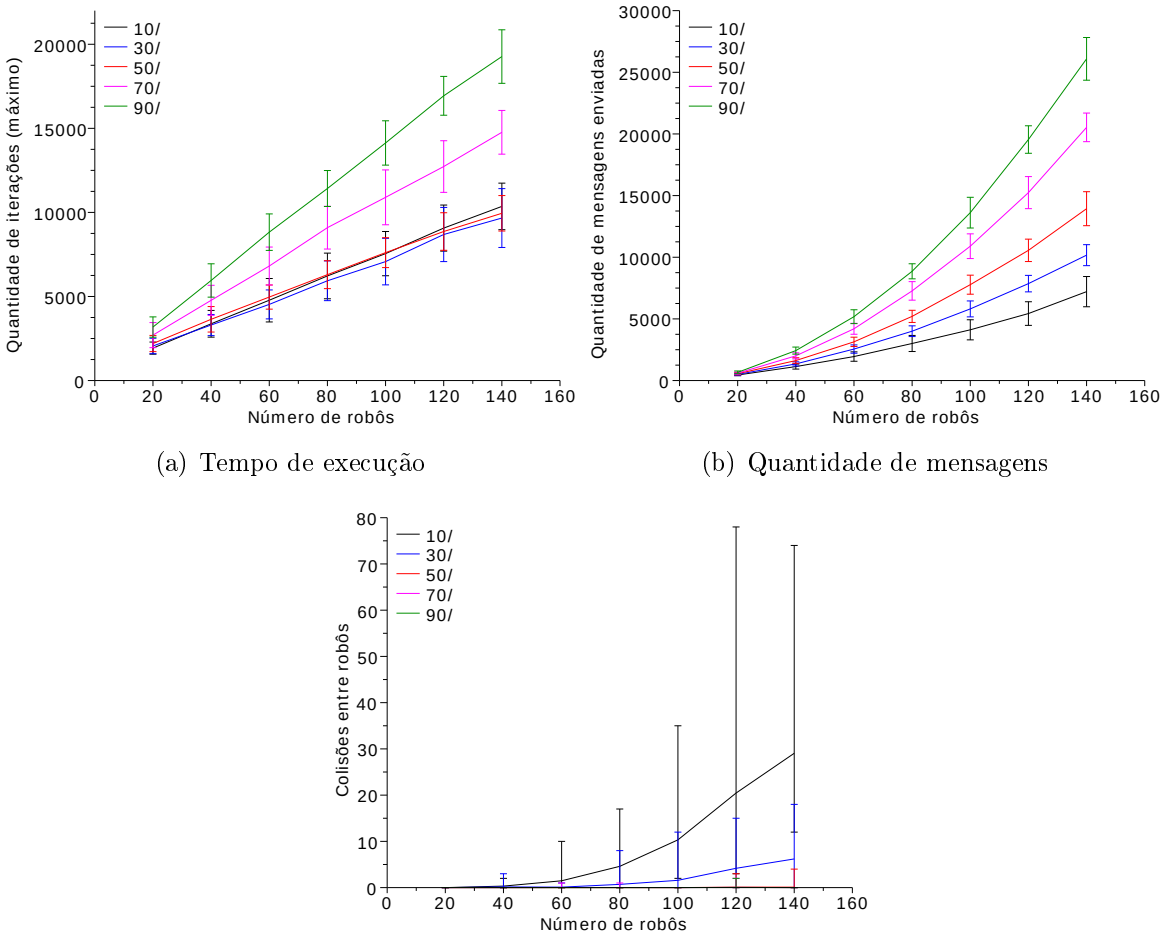
O *ângulo da região-α_A*, representado por α_A, exprime o ângulo de percepção para ignorar expulsão. Quando um robô não percebe algum robô entre ele e o alvo, com tolerância de α_A graus, ele avança até o alvo, ignorando mensagens de expulsão.

Foram realizados experimentos com α_A de 10 a 90 com saltos de 10 graus. A Figura 4.25 apresenta o resultado deste experimento quanto ao tempo de execução, quantidade de mensagens e colisões. Observa-se que o tempo de execução é pequeno quando se usou ângulos entre 10 e 50. A quantidade de mensagens é proporcional ao valor de α_R. Já, a quantidade de colisões é inversamente proporcional.

O método ANOVA aplicado ao resultado com α_A entre 10 e 50 retorna:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0,000 & 0,000 & 0,001 & 0,013 & 0,000 & 0,051 & 0,000 \end{pmatrix}$$

indicando que apesar de próximos, as médias dos resultados não são iguais entre si.



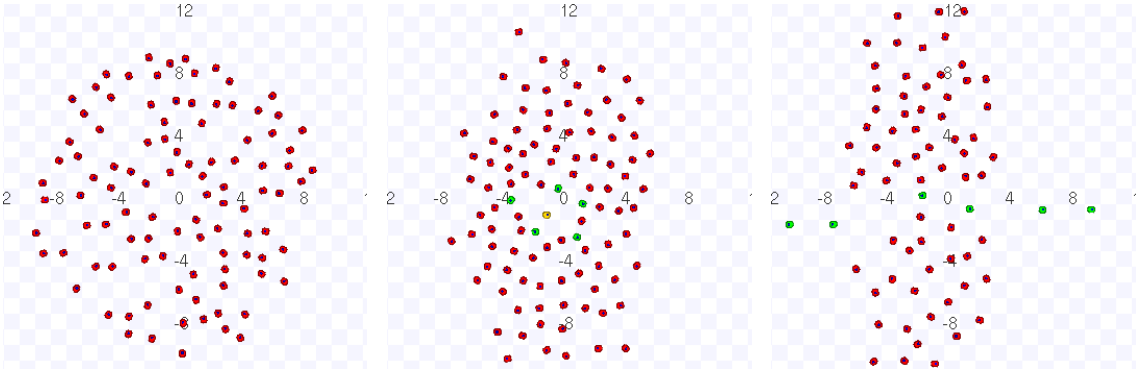
(a) Tempo de execução

(b) Quantidade de mensagens

(c) Colisões

Figura 4.25. Análise de α_A para algoritmo `Sirene`.

Quando um robô possui um α_A pequeno, tem mais chances dele ignorar os robôs ao seu lado, fazendo-o avançar até o alvo. Quando α_A é pequeno, os robôs mudam para o estado IGNORANDO com mais frequência. Isto faz com que mais mensagens sejam ignoradas e consequentemente menos réplicas são enviadas. Como as réplicas são enviadas com menos frequência, poucos robôs escutam a expulsão, fazendo com que se concentrem mais na região próximo ao alvo, aumentando a chance de colisões.



(a) Início da execução (b) Formação de regiões (c) Execução mais à frente

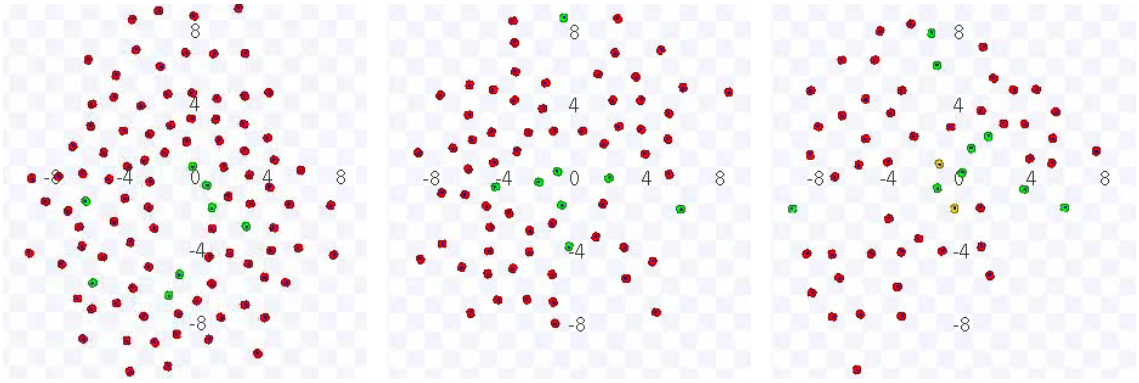
Figura 4.26. Imagens da execução (vídeo disponível em http://youtu.be/RQi0mw2oitc).

4.3.5 Execução do algoritmo

A Figura 4.26 apresenta algumas imagens do algoritmo `Sirene`. Robôs nos estados NORMAL, IGNORANDO, ENTRANDO e SAINDO são representados pelas cores vermelho, azul, amarelo e verde respectivamente.

Na Figura 4.26(a) tem-se o cenário inicial. Todos os robôs estão distribuídos aleatoriamente ao redor do alvo. Como nenhum alcançou a região de perigo, eles seguem em direção ao alvo sem seguir nenhum protocolo. A Figura 4.26(b) ilustra algumas iterações após o início. Cinco robôs alcançaram o alvo e estão tentando sair, emitindo mensagens de expulsão. Enquanto eles andam, o espaço vai se abrindo para que eles possam passar. No centro, há um robô que entrou no estado ENTRANDO. Ele comunica aos seus vizinhos sua mudança de estado e se um robô escutar sua mensagem é expulso. A Figura 4.26(c) apresenta uma etapa mais adiante, onde alguns robôs já fizeram seu trajeto de entrada e saída da região do alvo. Depois de algumas iterações, observa-se que os robôs passam a evitar as áreas por onde saem robôs (esquerda e direita do alvo). Observa-se que são formadas regiões de entrada e saída semelhantes às que são formadas propositalmente pelo algoritmo `Região`. As regiões formadas são resultantes de um comportamento emergente, visto que o algoritmo `Sirene` não foi implementado para formar esta região.

Quando o algoritmo `Sirene` é usado em um cenário onde os robôs seguem, não apenas para esquerda e direita, mas para cima e baixo após alcançar o alvo, as regiões formadas são diferentes. A Figura 4.27 apresenta algumas imagens de uma execução para este cenário. A Figura 4.27(a) apresenta algumas iterações após o início da execução, onde os primeiros robôs ainda estão abrindo espaço para saída. A Figura 4.27(b) e a Figura 4.27(c) mostram estágios posteriores, na quais notam-se que existem 4 regiões



(a) Início da execução (b) Formação de regiões (c) Formação de regiões mais à frente

Figura 4.27. Imagens da execução para quatro direções após alcançar o alvo.

separadas pelos robôs no estado SAINDO.

4.3.6 Conclusões

Nos experimentos, verificou-se que o tempo de execução não é influenciado pelo ciclo de mensagens. Isto ocorre porque quando se usa valores de T_M abaixo do valor de T_E usado, o robô irá escutar as mensagens posteriores, mas não irá obedecer à força de expulsão enviada visto que ele ainda está atuando a mensagem anterior. Portanto, qualquer mensagem de expulsão enviada dentro dos T_E ciclos de atuação da força de expulsão será ignorada. Contudo, ele pode enviar novas mensagens de réplica enquanto está sendo empurrado pela força de expulsão.

Além disso, os testes realizados mostraram que um valor alto de T_E faz com que os robôs se dispersem mais, pois levam mais tempo seguindo a força de expulsão. Isso faz com que os robôs se afastem mais do alvo, aumentando o tempo de chegada ao alvo. Quando os robôs utilizam pouco tempo para serem expulsos, eles tendem a não se afastar o bastante do alvo. A dispersão dos robôs no ambiente é inversamente proporcional ao valor de T_E. Sendo assim, se eles estão mais próximos do alvo, eles levam menos tempo para alcançá-lo, diminuindo o tempo de execução. Contudo, como eles se concentram mais na região em torno do alvo, a chance deles se chocarem é maior.

O ângulo α_R influencia inversa e proporcionalmente o tempo de execução enquanto que α_A piora o tempo de execução a partir de 50°. Quando o ângulo de réplica é pequeno, menos robôs recebem a mensagem e consequentemente poucos farão a sua réplica. Como poucos robôs obedecem a mensagem, uma maior quantidade de robôs se concentram na região do alvo. Por sua vez, mais robôs na mesma região aumentam

as chances de colisões entre eles. Quando um robô possui um α_A pequeno, tem mais chances dele ignorar os robôs ao seu lado, fazendo-o avançar até o alvo. Quando α_A é pequeno, os robôs mudam para o estado IGNORANDO com mais frequência. Isto faz com que mais mensagens sejam ignoradas e consequentemente menos réplicas são enviadas. Como as réplicas são enviadas com menos frequência, poucos robôs escutam a expulsão, fazendo com que se concentrem mais na região próximo ao alvo, aumentando a chance de colisões.

Também observou-se que, assim como nos algoritmos anteriores, à medida que um parâmetro melhora o tempo de execução, por outro lado aumenta a chance de haver colisão entre robôs.

Quando se utiliza o algoritmo `Sirene` para controlar o tráfego próximo do alvo, se os robôs tiverem novos alvos alinhados com as direções de saída do alvo original, regiões de entrada e saída semelhantes ao proposto no algoritmo `Região` são formadas.

4.4 Comparação entre os algoritmos

Os três algoritmos foram comparados entre si para analisá-los quanto às três métricas apresentadas neste capítulo. Inicialmente, para cada algoritmo, foram escolhidos os parâmetros que melhoram o desempenho quanto ao tempo de execução. As Tabelas 4.4-4.6 mostram estes parâmetros para os três algoritmos. Estes valores foram escolhidos de acordo com os experimentos realizados apresentados nas seções anteriores. O valor de K foi mantido igual em todos os experimentos, mas foi escolhido de tal forma que diminuísse a quantidade de colisões entre robôs. O valor de T_M para os algoritmos que trocam mensagens foi mantido igual para não enviesar a análise de quantidade de mensagens trocadas.

Parâmetro	K	ρ	*Raio*	T_P	T_M	T_E	*Alc. do sensor*
Valor	1,0	0,3	(3,5; 1,5)	40	25	30	3,0

Tabela 4.4. Valores dos parâmetros para o algoritmo `Região`.

Parâmetro	K	a
Valor	1,0	2,0

Tabela 4.5. Valores dos parâmetros para o algoritmo `Espiral`.

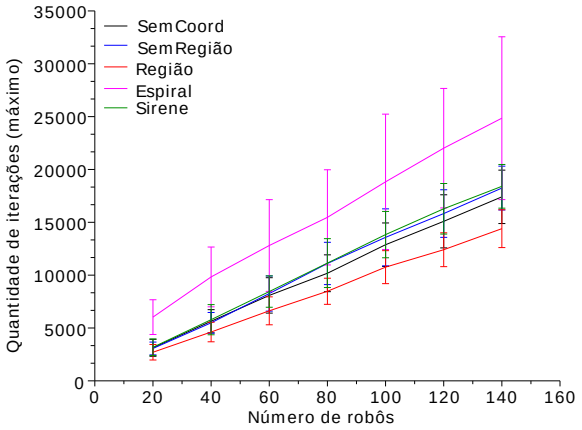
Estes três algoritmos também foram comparados com um algoritmo sem coordenação, isto é, os robôs possuem apenas campo repulsivo e campo atrativo para seu alvo

Parâmetro	K	T_M	T_E	α_R	α_A
Valor	1,0	25	5	50°	50°

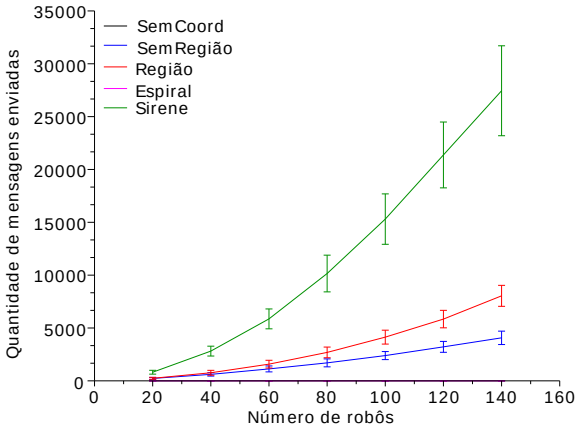
Tabela 4.6. Valores dos parâmetros para o algoritmo `Sirene`.

(`SemCoord`) e o algoritmo de coordenação original (`SemRegião`), proposto por Marcolino & Chaimowicz [2009], ambos usando o mesmo valor de K.

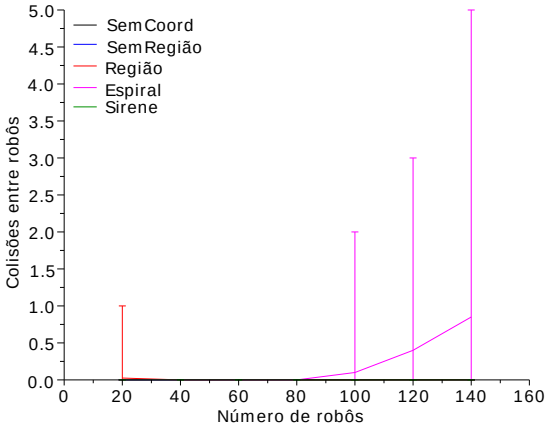
A Figura 4.28 apresenta a comparação entre estes algoritmos com robôs não holonômicos e a Figura 4.29, para robôs holonômicos. Na Figura 4.29, os resultados para colisões foram omitidos pois são todos iguais a zero.



(a) Tempo de execução



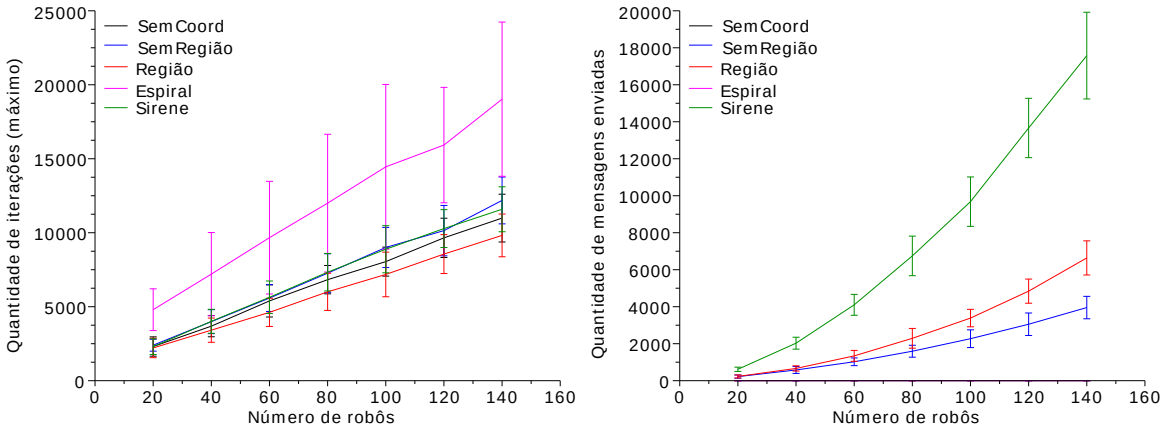
(b) Quantidade de mensagens



(c) Quantidade de colisões entre robôs

Figura 4.28. Comparação para robôs não holonômicos.

Em ambos os casos, o algoritmo `Região` é o melhor em tempo de execução. Os algoritmos `Sirene` e `SemRegião` tiveram um desempenho semelhantes, mas em relação à ausência de coordenação, o tempo de execução é maior. Ambos os algoritmos tiveram desempenho pior devido a ausência de uma área de saída para os robôs que



(a) Tempo de execução (b) Quantidade de mensagens

Figura 4.29. Comparação para robôs holonômicos.

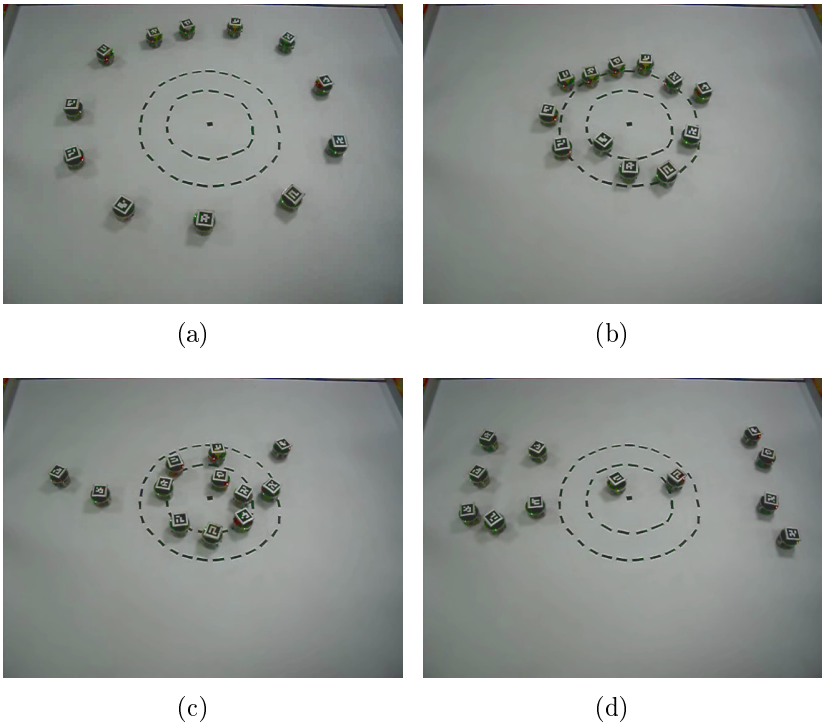
já alcançaram o alvo. Dentre os algoritmos que usam troca de mensagens, o algoritmo `Sirene` é o que mais envia mensagens e o algoritmo `SemRegião` o que menos usa. O algoritmo `Espiral` teve mais colisões entre robôs, visto que foi utilizado $a = 2,0$ e, como discutido na Seção 4.3, este valor reduz o tempo de execução, mas aumenta a quantidade de colisões. Portanto, dentre os três algoritmos, o algoritmo `Região` é o melhor a ser utilizado para controle de tráfego para enxames de robôs com alvos em comum.

4.5 Experimentos reais

Como prova de conceito, os três algoritmos foram testados com robôs *e-puck*, com o intuito de testar a aplicabilidade em ambiente reais imersos com erro de localização, comunicação e atuação. Além destes três algoritmos propostos, também foram testados em *e-pucks* os algoritmos `SemCoord` e `NoCoord`.

Para localização, foi usado um sistema para enxames em ambientes internos [Garcia & Chaimowicz, 2009]. Com este sistema, a posição dos robôs podem ser consideradas num referencial global, tornando direta a implementação dos algoritmos da forma em que foram discutidos no Capítulo 3. Os sensores infravermelhos dos *e-pucks* possuem baixo alcance. Em vista disso, foram implementados sensores virtuais a partir das posições obtidas do sistema de localização. O cenário dos experimentos é o mesmo usado nas simulações. Foi usado um valor de $D = 70$ cm e os robôs iniciam a mais de 45 cm do alvo.

A Figura 4.30 apresenta algumas imagens da execução do algoritmo `SemCoord`. A



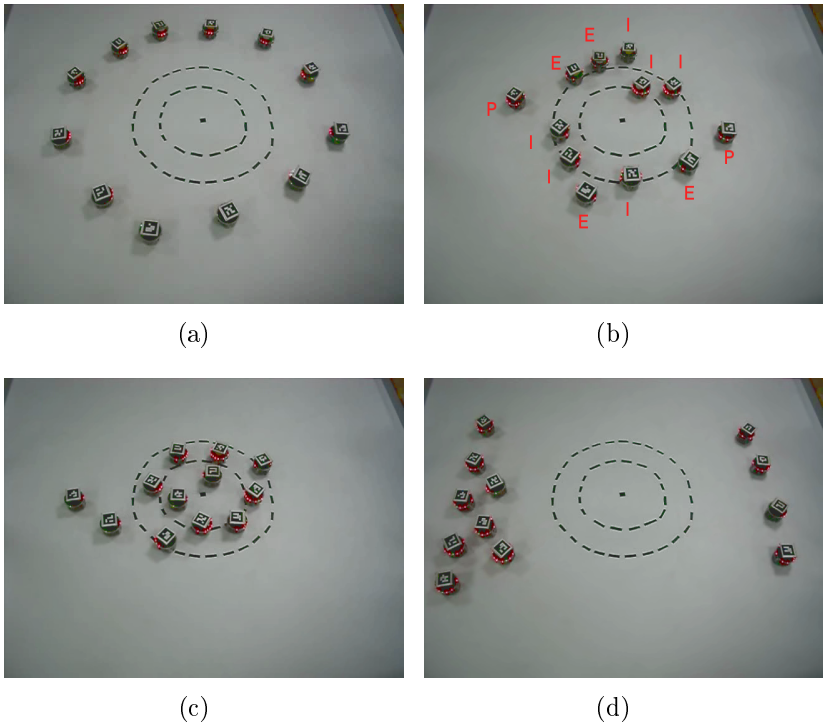
(a) (b)

(c) (d)

Figura 4.30. Experimento com *e-pucks* para o algoritmo `SemCoord` (vídeo disponível em http://youtu.be/GvzgMhxjqk0).

Figura 4.30(a) apresenta a configuração inicial. A Figura 4.30(b) mostra todos os robôs indo em direção ao alvo. Como este algoritmo não possui coordenação, todos vão em direção ao alvo usando apenas forças repulsivas para não se chocarem. A Figura 4.30(c) mostra um estágio onde a maioria alcançou o alvo e os robôs estão tentando sair desta região, que está congestionada por outros robôs que também estão tentando entrar nesta região. A Figura 4.30(d) ilustra uma etapa final onde a maioria dos robôs já alcançou o alvo.

A Figura 4.31 apresenta algumas imagens da execução do algoritmo `SemRegião`. A Figura 4.31(a) apresenta a configuração inicial. A coroa demarcada ao redor do alvo possui raios de 15 cm e 25 cm, aproximadamente. Esta coroa corresponde respectivamente a região de perigo e a região livre. A Figura 4.31(b) mostra alguns segundos depois, quando os robôs alcançam o alvo e mudam de estado. Os robôs estão representados pelas iniciais dos estados em que se encontram: (E)SPERANDO, (P)RESO e (I)MPACIENTE. Como existe erro de localização, os robôs no estado ESPERANDO não ficam exatamente dentro da coroa. A Figura 4.31(c) apresenta um estágio posterior, onde nota-se que alguns robôs desejam sair da região do alvo, como ocorre para o algoritmo `SemCoord`. A Figura 4.31(d) ilustra uma etapa final onde a maioria dos robôs já alcançou o alvo.



(a) (b)

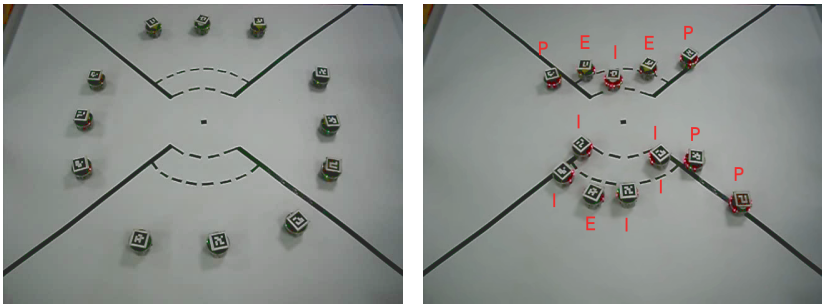
(c) (d)

Figura 4.31. Experimento com *e-pucks* para o algoritmo `SemRegião` (vídeo disponível em http://youtu.be/Ikp4WuwfBz0).

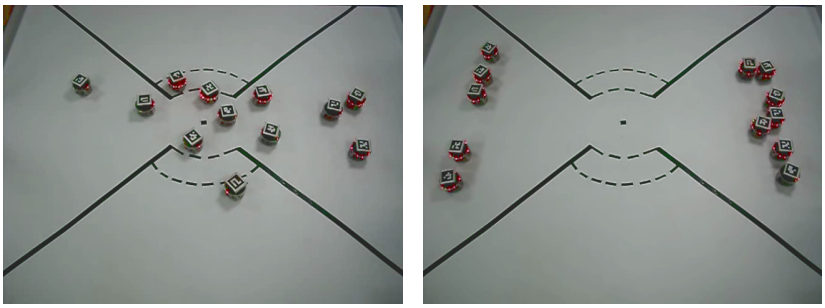
A Figura 4.32 apresenta algumas imagens da execução do algoritmo `Região`. A Figura 4.32(a) apresenta a configuração inicial. A coroa demarcada ao redor do alvo possui as mesmas dimensões para o experimento anterior e também representa a região de perigo e a região livre. As linhas que seccionam estas coroas representam a linha que demarca as regiões de entrada e saída. A Figura 4.32(b) mostra alguns segundos depois, quando os robôs imediatamente vão para região de entrada, mudando de estado. A Figura 4.32(c) apresenta os robôs usando as regiões de saída para alcançar as extremidades. A Figura 4.32(d) ilustra a etapa final, depois de todos robôs terem alcançado o alvo central.

A Figura 4.33 apresenta algumas imagens da execução do algoritmo `Espiral`. A Figura 4.33(a) apresenta a configuração inicial. Como nos algoritmos anteriores, os robôs se encontram inicialmente ao redor do alvo. A Figura 4.33(b) mostra alguns segundos depois, durante a formação da fila em espiral. A Figura 4.33(c) apresenta o final da formação da espiral, quando a maioria dos robôs já alcançaram o alvo e estão saindo da espiral pela parte superior. Os robôs após alcançarem o alvo, devem ir para pontos aleatórios na esquerda ou direita, evitando a espiral. A Figura 4.33(d) ilustra o momento final, onde os robôs já alcançaram o alvo e se encontram nas laterais.

A Figura 4.34 apresenta algumas imagens da execução do algoritmo `Sirene`. A

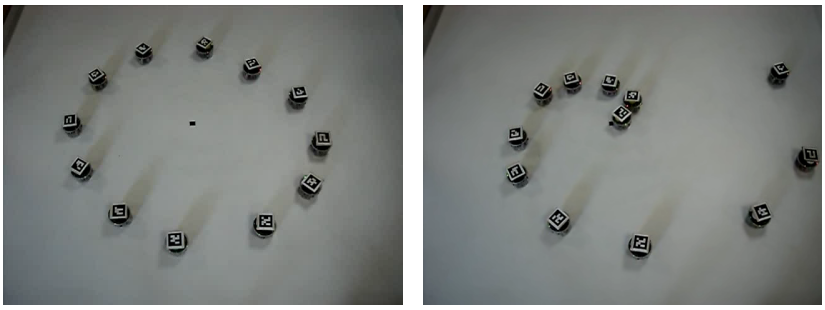


(a) (b)

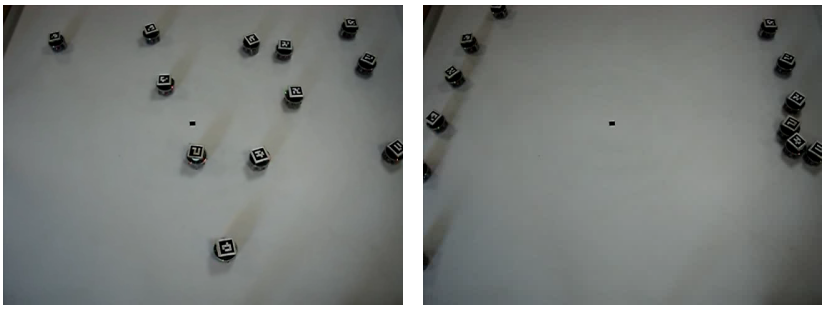


(c) (d)

Figura 4.32. Experimento com *e-pucks* para o algoritmo `Região` (vídeo disponível em http://youtu.be/WOIxwrn43J8).

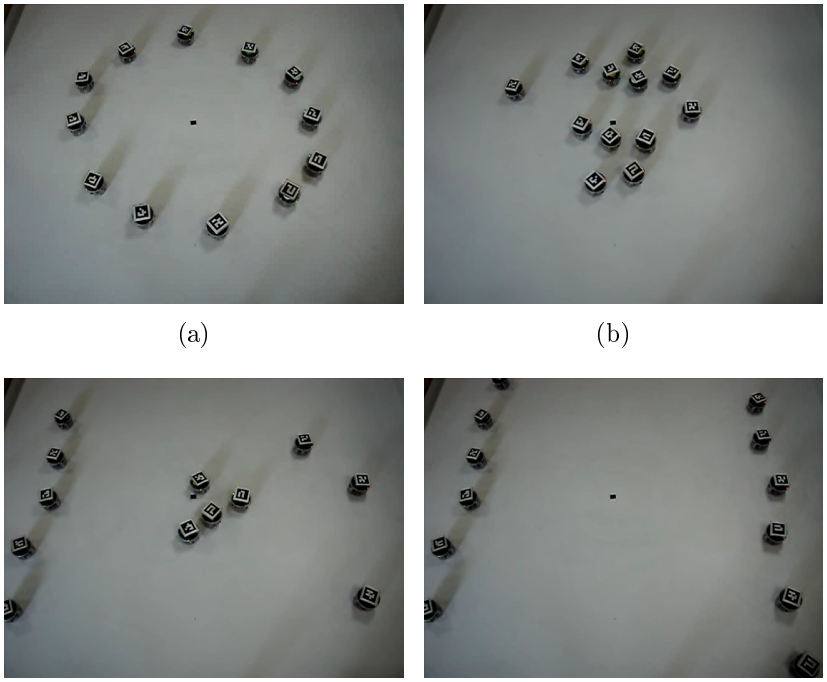


(a) (b)



(c) (d)

Figura 4.33. Experimento com *e-pucks* para o algoritmo `Espiral` (vídeo disponível em http://youtu.be/AWwjl24a0l4).



(a) (b)

(c) (d)

Figura 4.34. Experimento com *e-pucks* para o algoritmo `Sirene` (vídeo disponível em http://youtu.be/QcqnX9Y5QNA).

Figura 4.34(a) apresenta a configuração inicial, semelhante aos algoritmos anteriores. A Figura 4.34(b) mostra alguns segundos depois, onde dois robôs estão saindo do alvo. Um está indo para direita e outro para a esquerda. Nota-se que os demais robôs se afastaram do alvo para dar passagem para estes dois. A Figura 4.34(c) apresenta um momento onde um dos robôs se aproxima do alvo central e emite a mensagem de expulsão. Os três robôs abaixo dele, que estavam se aproximando do alvo, ao escutarem mudam de direção para se afastar do alvo e dar passagem ao robô que alcançou primeiro o alvo. A Figura 4.34(d) ilustra o momento final, onde os robôs já alcançaram o alvo e se encontram nas laterais.

Durante os testes reais, percebeu-se que ocorre o mesmo padrão de comportamento ocorrido durante as simulações. O algoritmo `Região` é executado num tempo de execução menor que os algoritmos `Sirene`, `SemCoord` e `SemRegião` (sendo que estes três possuem tempos de execução semelhantes). O algoritmo `Espiral` foi o mais lento dos demais.

Capítulo 5

Conclusões

Este trabalho apresentou três soluções para o problema do controle de tráfego para enxames de robôs quando há um alvo em comum. O primeiro algoritmo consiste em usar máquinas de estado finito probabilísticas e formação de regiões de entrada e saída próximo a região do alvo (algoritmo `Região`). O segundo algoritmo consiste em formar uma fila em forma de espiral ao redor do alvo (algoritmo `Espiral`). O terceiro algoritmo visa usar emissão de mensagens para expulsar os robôs que se encontram próximo do alvo, simulando as sirenes de ambulâncias (algoritmo `Sirene`).

Experimentos com simulação foram realizados para os três algoritmos, visando observar a influência dos parâmetros desses algoritmos no: (i) tempo de execução, (ii) quantidade de mensagens enviados e (iii) quantidade de colisões entre os robôs. Para o algoritmo `Região`, notou-se que:

- o valor de K para a força repulsiva influencia diretamente no tempo de execução, mas, existe a desvantagem de aumentar a chance dos robôs se chocarem – visto que a área de atuação do robô aumenta proporcionalmente ao valor de K;
- a probabilidade de transição para o estado IMPACIENTE ρ deve ser escolhida levando em conta o tamanho da área da região livre – pois quanto menor o valor de ρ mais rápido ele vai em direção ao alvo;
- os experimentos realizados levantam a hipótese de que a partir de um ρ^*, o desempenho do algoritmo não se altera – visto que o tempo médio que o robô espera para entrar no estado IMPACIENTE compensa o tempo de chegada no alvo quando há congestionamentos;
- assim como ocorre com ρ, o valor de T_P também depende do tamanho da região livre – visto que o tempo que um robô leva para ir em direção ao alvo influencia:

(i) a quantidade de robôs que irão ocupar a região próxima do alvo, gerando congestionamentos e (ii) o tempo em que o sistema como um todo leva para alcançar e sair do alvo;

- os experimentos também mostram que existe uma relação entre T_M e T_E e caso eles sejam muito diferentes o tempo de execução é aumentado – pois durante o tempo de envio entre uma mensagem e outra, os robôs podem seguir até o alvo (como se não existisse robôs em espera), fazendo com que os robôs avancem, atrapalhando o trajeto de robôs que já estão na região de perigo ou estão saindo pela região de saída.

Para o algoritmo `Espiral`, percebeu-se que:

- o valor de K para a força repulsiva não influencia de modo proporcional, como ocorre com o algoritmo `Região`, mas muda a depender do valor de a utilizado – pois mais robôs ocupam a região-ϵ quando se aumenta K e mais robôs ficam próximo da região de perigo, gerando mais congestionamentos;

- o valor de a influencia de modo inversamente proporcional o tempo de execução e de modo proporcional a quantidade de colisões – pois quanto maior o valor de a, menor é a distância a ser percorrida até o centro fazendo com que os robôs levem menos tempo para chegar ao alvo e aumenta-se a concentração de robôs, resultando em mais colisões.

Os testes em simulação para o algoritmo `Sirene` mostraram que:

- o tempo de execução não é influenciado pelo ciclo de mensagens (T_M) de maneira proporcional, mas a quantidade de mensagens enviadas durante a execução é inversamente proporcional – pois quando se usa valores de T_M abaixo do valor de T_E usado, o robô irá escutar as mensagens posteriores, mas não irá obedecer à força de expulsão enviada, visto que ele ainda está atuando a mensagem anterior;

- um valor alto para o ciclo de expulsão (T_E) faz com que os robôs se dispersem mais, aumentando o tempo de chegada ao alvo, visto que um valor alto de T_E faz com que os robôs se dispersem mais – pois levam mais tempo seguindo a força de expulsão, fazendo com que os robôs se afastem mais do alvo, aumentando o tempo de chegada ao alvo;

- o ângulo α_R influencia inversa e proporcionalmente o tempo de execução – visto que quando o ângulo de réplica é pequeno, menos robôs recebem a mensagem e consequentemente poucos farão a sua réplica;

- o valor de α_A piora o tempo de execução a partir de 50° – já que os robôs mudam para o estado IGNORANDO com mais frequência, fazendo com que mais mensagens sejam ignoradas e consequentemente menos réplicas são enviadas;
- as regiões formadas por este algoritmo são resultantes de um comportamento emergente – visto que ele não foi implementado para formar esta região.

As três abordagens foram comparadas e concluiu-se que: o algoritmo `Região` possui menor tempo de execução; o algoritmo `Espiral` não precisa de mensagens, apesar de não possuir bom desempenho quanto o tempo de execução; e o algoritmo `Sirene` pode formar regiões de entrada e saída semelhantes às formadas pelo algoritmo `Região`.

Pretende-se testar estes algoritmos utilizando-se referencial local, visto que foi considerado que os robôs usam localização baseada num referencial global. Para isso, os robôs devem ser equipados com um dispositivo que indique um sentido comum a todos (por exemplo uma bússola). Assim, os algoritmos podem ser implementados usando referencial local e, posteriormente, transformando-os para o referencial global obtido pelo dispositivo.

Os algoritmos apresentados necessitam que seus parâmetros sejam estabelecidos antes de uma implementação real. Estes parâmetros dependem de medidas do ambiente (tamanho do alvo, tamanho do espaço livre em torno do alvo) e do robô (raio de alcance do sensor, raio de alcance do envio e recepção de mensagens, área ocupada pelo robô, fragilidade do robô - para estabelecer um K que evite choques entre robôs). Um trabalho futuro a ser sugerido é a descoberta automática destes parâmetros, dadas estas medidas.

Outro possível trabalho futuro é analisar o comportamento dos robôs utilizando teoria das filas. Dessa forma, pode-se deduzir um limite de tempo mínimo que pode ser alcançado pelo algoritmo e comparar com os resultados obtidos pelo algoritmo `Região`, verificando assim se é possível novas melhorias.

O algoritmo `Sirene` considera que todos os robôs possuem igual prioridade para alcançar o alvo. Alem disso, apenas os robôs mais próximos do alvo podem enviar mensagem de expulsão para entrar na região do alvo. Como trabalho futuro, pretende-se modificar o algoritmo `Sirene` para permitir uma mudança de prioridade entre robôs, de tal forma que robôs mais distantes do alvo possam ter prioridade maior do que aqueles que se encontram próximo do alvo.

O algoritmo `Região` foi publicado no *X Simpósio Brasileiro de Automação Inteligente - SBAI 2011* [Passos & Chaimowicz, 2011]. O problema do controle de tráfego em enxames de robôs não foi abordado na literatura de forma específica, por conseguinte

o presente trabalho sugere um diferente subtópico de estudo para a área de enxames de robôs.

Referências Bibliográficas

Asama, H.; Habib, M.; Endo, I.; Ozaki, K.; Matsumoto, A. & Ishida, Y. (1991). Functional distribution among multiple mobile robots in an autonomous and decentralized robot system. In *ICRA*, pp. 1921 –1926 vol.3.

Barnes, J. W. (1994). *Statistical analysis for engineers and scientists: a computer-based approach.* McGraw-Hill.

Beni, G. & Wang, J. (1989). Swarm intelligence in cellular robotic systems. In *Proceed. NATO Advanced Workshop on Robots and Biological Systems.*

Caloud, P.; Choi, W.; Latombe, J.-C.; Le Pape, C. & Yim, M. (1990). Indoor automation with many mobile robots. In *IROS*, volume 1, pp. 67–72.

Camazine, S.; Deneubourg, J.-L.; Franks, N. R.; Sneyd, J.; Theraulaz, G. & Bonabeau, E. (2001). *Self-Organization in Biological Systems.* Princeton University Press.

Cao, Y. U.; Fukunaga, A. S. & Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:226--234.

Cui, X.; Hardin, C. T.; Ragade, R. K. & Elmaghraby, A. S. (2004). A swarm approach for emission sources localization. In *ICTAI '04: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 424--430, Washington, DC, USA. IEEE Computer Society.

Garcia, R. F. & Chaimowicz, L. (2009). Uma infra-estrutura para experimentação com enxames de robôs. In *SBAI09.*

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

Goldberg, D.; Cicirello, V.; Dias, M. B.; Simmons, R.; Smith, S.; Smith, T. & Stentz, A. (2002). A distributed layered architecture for mobile robot coordination: Application

to space exploration. In *In Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*.

Grossman, D. (1988). Traffic control of multiple robot vehicles. *Robotics and Automation, IEEE Journal of*, 4(5):491 –497.

Guo, Y. & Parker, L. E. (2002). A distributed and optimal motion planning approach for multiple mobile robots. In *In Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2612--2619.

Hayes, A. T. (2002). *Self-Organized Robotic System Design and Autonomous Odor Localization*. PhD thesis, California Institute of Technology.

Hoshino, S. (2011). Multi-robot coordination methodology in congested systems with bottlenecks. In *IROS*, pp. 2810–2816. IEEE.

Kato, S.; Nishiyama, S. & Takeno, J. (1992). Coordinating mobile robots by applying traffic rules. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, volume 3, pp. 1535–1541.

Kennedy, J.; Eberhart, R. C. & Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann.

Krontiris, A. & Bekris, K. E. (2011). Using minimal communication to improve decentralized conflict resolution for non-holonomic vehicles. In *IROS*, pp. 3235–3240. IEEE.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA.

Luca, A. D. & Oriolo, G. (1994). Local incremental planning for nonholonomic mobile robots. In *In Proceedings of 1994 International Conference on Robotics and Automation*, pp. 104--110.

Marcolino, L. & Chaimowicz, L. (2009). Traffic control for a swarm of robots: Avoiding target congestion. In *IROS*, pp. 1955 –1961.

Mondada, F.; Floreano, D.; Guignard, A.; Deneubourg, J.-L.; Gambardella, L.; Nolfi, S. & Dorigo, M. (2002). Search for rescue: an application for the swarm-bot self-assembling robot concept. Technical report, Swiss Federal Institute of Technology.

Murphy, R.; Blitch, J. & Casper, J. (2002). Aaai/robocup-2001 urban search and rescue events: reality and competition. *AI Mag.*, 23(1):37--42.

Murphy, R. R. (2000). *An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Olmi, R.; Secchi, C. & Fantuzzi, C. (2009). A coordination technique for automatic guided vehicles in an industrial environment. In *IFAC 9th International IFAC Symposium on Robot Control*.

Pallottino, L.; Scordio, V. G.; Bicchi, A. & Frazzoli, E. (2007). Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics*, 23(6):1170–1183.

Parker, L. (2008). Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems. *Journal of Physical Agents*, 2(1):5--14.

Parker, L. E. (2003). The effect of heterogeneity in teams of 100+ mobile robots. In *MultiRobot Systems Volume II: From Swarms to Intelligent Automata*, pp. 205--215. Kluwer.

Passos, Y. T. d. & Chaimowicz, L. (2011). Controle de congestionamento para enxames de robôs no acesso a alvos em comum. In *SBAI11*.

Peasgood, M.; Clark, C. & McPhee, J. (2008). A complete and scalable strategy for coordinating multiple robots within roadmaps. *Robotics, IEEE Transactions on*, 24(2):283–292.

Pickover, C. A. (2005). *A Passion for Mathematics*. John Wiley & Sons, Inc., New Jersey.

Rahmani, M. (2004). Introduction search and rescue using swarm robots - complex systems seminar. Disponível em http://130.203.133.150/viewdoc/download?doi=10.1.1.129.4477&rep=rep1&type=pdf.

Sahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics'04*, pp. 10–20.

Sahin, E.; Girgin, S.; Bayindir, L. & Turgut, A. E. (2008). Swarm robotics. In *Swarm Intelligence*, pp. 87–100. Springer.

Siegwart, R. & Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA.

Simmons, R.; Singh, S.; Hershberger, D.; Ramos, J. & Smith, T. (2000). First results in the coordination of heterogeneous robots for large-scale assembly. In *In Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Songdong, X. & Jianchao, Z. (2008). Sense limitedly, interact locally: the control strategy for swarm robots search. In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pp. 402 –407.

Truszkowski, W.; Hinchey, M.; Rash, J. & Rouff, C. (2004). Nasa's swarm missions:the challenge of building autonomous software. *IT Professional*, 6:47–52.

Wang, J. (1991). Fully distributed traffic control strategies for many-agv systems. In *IROS*, pp. 1199 –1204 vol.3.