Um Modelo Baseado em Autômatos Celulares e Algoritmos Genéticos para a Navegação de um Time de Robôs visando o Controle de Formação e o Desvio de Obstáculos

Reslley Gabriel Oliveira Silva

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2015

Reslley Gabriel Oliveira Silva

Um Modelo Baseado em Autômatos Celulares e Algoritmos Genéticos para a Navegação de um Time de Robôs visando o Controle de Formação e o Desvio de Obstáculos

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof.ª Dr.ª Gina Maira Barbosa de Oliveira

Uberlândia
2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S586m 2015

Silva, Reslley Gabriel Oliveira, 1991-
Um modelo baseado em autômatos celulares e algoritmos genéticos para a navegação de um time de robôs visando o controle de formação e o desvio de obstáculos / Reslley Gabriel Oliveira Silva. - 2015.
136 f. : il.

Orientadora: Gina Maira Barbosa de Oliveira.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Robôs - Teses. I. Oliveira, Gina Maira Barbosa de. II. Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Este trabalho é dedicado a todas as pessoas que buscam desenvolver ideias novas e de alguma forma contribuir para a evolução da ciência.

Agradecimentos

Agradeço primeiramente a minha mãe Hellizan e meu pai Clésio pelo apoio em todos os momentos. Minha namorada Joyce pela compreensão e companheirismo. À minha orientadora Dr[a]. Gina Maira Barbosa de Oliveira pela sua ajuda e ideias relevantes para a conclusão deste trabalho. À CAPES pelo apoio financeiro. Por fim, gostaria de agradecer também às pessoas que de uma forma ou de outra também foram importantes para o término desta etapa: Laurence Amaral, Juan Manuel, Giordano Bruno, Micael Couceiro. A todos, meu muito obrigado!

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original."
(Albert Einstein)

Resumo

A tarefa de controle de formação trata de como coordenar robôs autônomos, fazendo com que eles mantenham uma formação específica enquanto percorrem o ambiente. Em nosso trabalho, investigamos modelos baseados em autômatos celulares aplicados a esse problema. Implementamos métodos previamente propostos na literatura e encontramos algumas limitações que impossibilitaram que os robôs executassem boas trajetórias. Desta forma, criamos um método evolutivo-cooperativo, com o objetivo de utilizar as vantagens de convergência e dinâmica de um algoritmo genético aliadas às regras compactas e processamento simplificado dos autômatos celulares. O novo método foi testado tanto em simulação quanto em experimentos com robôs reais. Como resultado, obtivemos uma melhoria no sistema de deslocamento dos robôs e custo computacional baixo, tornando o sistema adequado para cenários reais.

Palavras-chave: Autômatos Celulares. Controle de Formação. Time de Robôs Cooperativos.

Abstract

Formation control is the task of coordinating a group of robots to get into and to maintain a formation with a specific shape while moving in the environment. In this work we investigated models based on cellular automata applied to this problem. We implemented methods previously proposed in the literature and found some key limitations that worsened the robot's trajectory quality. Thus, we created an evolutionary-cooperative method, using the convergence and dynamics of a genetic algorithm along with the compact rules and simplified processing of cellular automatas. The new method required low computational infrastructure and was tested both in simulation and in real world experiments. At the end, the new model exhibited better behaviours than their precursors in several scenarios, improving the robot's trajectory and formation maintenance.

Keywords: Cellular Automata. Formation Control. Cooperative Robot Team.

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

O planejamento de trajetórias livres de obstáculos e a movimentação coordenada de times de robôs estão entre as áreas mais estudadas e desafiadoras da robótica cooperativa (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011a). Em diversas aplicações, é necessária a utilização de um grupo de robôs capaz de manter e dinamicamente reconfigurar sua organização em time de modo que formem algum padrão espacial de posicionamento. Nesse âmbito, a tarefa de controle de formação trata de como coordenar o grupo de robôs, fazendo com que eles realizem determinadas ações apresentando comportamento cooperativo e mantendo uma formação específica (linear ou triangular, por exemplo), enquanto percorrem uma trajetória e desviam de obstáculos pelo ambiente (NAVARRO; MATIA, 2013). A cooperação entre os robôs visa aumentar a robustez do grupo e pode ser realizada de várias formas, sendo geralmente feita via comunicação direta.

Um sistema multiagente composto por pequenos robôs em formação apresenta diversas vantagens em relação a sistemas que usam apenas um robô, pois permitem maior flexibilidade e adaptabilidade, resolvendo tarefas complexas de forma mais eficiente e robusta. O custo total do sistema também é reduzido, pois os robôs são relativamente mais baratos do que um robô maior que seria utilizado sozinho (CHEN; WANG, 2005).

1.1 Motivação

No decorrer dos últimos anos, o tópico de controle em times de robôs tem se tornado uma área de crescente interesse aos pesquisadores, principalmente por sua aplicação em uma grande variedade de tarefas como mapeamento, busca e resgate, vigilância, exploração e agricultura (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011b). Diversos métodos foram propostos para tentar resolver o problema de controle de formação: sistemas não lineares (OGREN; EGERSTEDT; HU, 2002), lógica fuzzy (FU; LI; MA, 2006), (MEHRJERDI; SAAD; GHOMMAM, 2011), redes neurais artificiais (DIERKS; JAGANNATHAN, 2009), (CAPI; MOHAMED, 2012), aprendizagem por reforço (ZUO; HAN; HAN, 2010), campos potenciais (SCHNEIDER; WILDERMUTH, 2003), (REZAEE; AB-

DOLLAHI, 2014). No entanto, a tarefa ainda é considerada bem difícil, pois envolve diversos fatores: evitar obstáculos estáticos e dinâmicos, manter e retomar a formação, adaptar-se dinamicamente às mudanças no número de robôs e minimizar o gargalo da comunicação (NAVARRO; MATIA, 2013).

A maioria dos métodos citados anteriormente apresenta alta complexidade e demanda muito recurso computacional, o que impossibilita a sua utilização em tempo real. Métodos descentralizados e que utilizam informações locais são mais eficientes para lidar com os cenários reais, pois apresentam maior tolerância a falhas de comunicação e mudanças no ambiente (GUANGHUA et al., 2013). Abordagens baseadas em Autômatos Celulares (ACs) têm sido estudadas e apresentam resultados promissores, com baixo custo computacional e bom desempenho em aplicações utilizando robôs reais (MEAD; LONG; WEINBERG, 2009), (DING; HE, 2010), (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), (FERREIRA, 2014). Após analisarmos vários trabalhos na área, percebemos que além dos desafios intrínsecos ao problema de controle de formação existem outros aspectos que influenciam fortemente na qualidade dos métodos, tais como, a calibragem dos sensores, erros na previsão das posições, dependência de parâmetros pré-fixados e inconsistências entre simulação e experimentos reais. Considerando tais desafios, decidimos propor um método evolutivo-cooperativo baseado em Algoritmo Genético (AG) (GOLDBERG, 1989) e Autômatos Celulares (WOLFRAM, 1983a) para tratar o problema de planejamento de caminho e controle de formação de um time de robôs. Numa primeira etapa desse trabalho, realizamos adaptações e refinamentos no modelo anterior descrito em (FERREIRA, 2014). Posteriormente, parâmetros e estratégias do modelo adaptado foram otimizadas através de um algoritmo genético, de forma a obter o novo modelo evolutivo-cooperativo.

1.2 Objetivos e Desafios da Pesquisa

Este trabalho apresenta a investigação e implementação de modelos evolutivo-cooperativos baseados em AC e AG para a realização do planejamento de rotas e controle de formação em robôs autônomos. A seguir estão descritos os objetivos traçados para que pudéssemos atingir nossa proposta de trabalho.

1.2.1 Objetivo Geral

Contribuir para a melhoria dos sistemas de navegação de times de robôs, na tarefa de planejamento de trajetórias e controle de formação, por meio do desenvolvimento e aplicação de um novo modelo evolutivo-cooperativo baseado em AC e AG. Esse novo modelo teve como ponto de partida o modelo cooperativo baseado em AC proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e aperfeiçoado em (FERREIRA, 2014). Cabe ressaltar que os dois modelos anteriores são baseados em ACs, mas não fazem uso de AGs.

Os objetivos específicos a seguir foram criados de modo a guiar a pesquisa nas diversas etapas de desenvolvimento.

1.2.2 Objetivos Específicos

- ❏ Implementar e investigar os modelos baseados em ACs propostos em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e (FERREIRA, 2014), identificando os parâmetros e características dos modelos passíveis de aperfeiçoamento e melhoria;
- ❏ Efetuar modificações no modelo de autômato celular, através da inclusão de novos estados e regras, para obter um modelo onde a navegação de cada robô no desvio de obstáculos apresentasse um comportamento mais eficiente, em relação às deficiências identificadas nos modelos anteriores;
- ❏ Realizar simulações na plataforma Webots e experimentos envolvendo um único robô e-puck para validar o novo modelo quanto ao comportamento no desvio de obstáculos;
- ❏ Verificar o comportamento do time de robôs utilizando o novo modelo realizando simulações em ambientes virtuais;
- ❏ Implementar novas estratégias de comunicação entre os robôs para que eles apresentem comportamento cooperativo e permitam uma implementação efetiva do mesmo em ambientes reais, uma vez que o modelo anterior descrito em (FERREIRA, 2014) não foi avaliado em robôs reais (apenas em simulação);
- ❏ Validar o desempenho cooperativo do time de robôs através de simulações e experimentos envolvendo um time de três robôs e-pucks reais em diversos cenários;
- ❏ Identificar os parâmetros do novo modelo passíveis de otimização através de um método evolutivo;
- ❏ Otimizar parâmetros do modelo cooperativo baseado em ACs com o auxílio de um AG, gerando um modelo híbrido evolutivo-cooperativo;
- ❏ Validar o desempenho do modelo evolutivo-cooperativo através de simulações envolvendo um time de três e-pucks em diversos cenários.

.

1.3 Hipótese

A abordagem proposta utiliza as vantagens de convergência e dinâmica do AG aliadas às regras compactas e processamento simplificado dos ACs. Nossa hipótese era de que a

utilização desse modelo evolutivo-cooperativo possibilitaria a melhoria no sistema de deslocamento de times de robôs cooperativos, permitindo a geração de caminhos otimizados, controle da formação com desvio de obstáculos e custo computacional razoável, tornando o sistema adequado para cenários reais.

1.4 Contribuições

Inicialmente investigamos profundamente os métodos propostos por (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e por (FERREIRA, 2014), implementando-os com o simulador Webots (CYBERBOTICS, 2015). Depois, aplicamos os modelos a robôs e-puck (LAUSANNE, 2015), analisamos o comportamento do método quando aplicado à times em diversos cenários e, a partir dessa análise, detectamos 4 principais pontos fracos nas abordagens passíveis de melhorias.

O primeiro ponto foi a qualidade da trajetória gerada, na qual o robô não executava o caminho desejado, em linha reta, devido ao alto número de rotações e movimentos em zigue-zague. Nesse ponto, nossa contribuição foi a criação de um novo modelo de AC, com a inclusão de cinco novos estados, com suas respectivas regras para o desvio de obstáculos, e duas novas possibilidades de atualização da vizinhança, permitindo a movimentação contínua do robô em 45º ou -45º. Essas características do novo modelo possibilitaram um comportamento mais adequado do robô na tarefa de contorno de um obstáculo, efetuando o completo desvio da quina de forma mais natural e a retomada do eixo de deslocamento com um menor erro residual.

O segundo ponto diz respeito ao alto grau de dependência em parâmetros fixos dos métodos analisados, em especial aos valores limites para sensores de distância utilizados na identificação de obstáculos. Tendo em vista que uma das principais fases do modelo é a definição dos estados das células vizinhas baseado na leitura dos sensores, a inclusão de uma camada de inteligência que substitua a utilização de valores arbitrários aumenta a robustez do método a diferentes ambientes e iluminação. Deste modo, desenvolvemos um AG para a classificação automática da vizinhança capaz de gerar regras e repassá-las ao AC que então escolhe a regra adequada.

O terceiro ponto, e consequente terceira contribuição em nosso trabalho, se deu quanto à estrutura rígida de mestre-escravos presente nos métodos anteriores. O robô líder era responsável por toda a tomada de decisão quanto ao controle de formação, sendo que para isso a comunicação era realizada a cada passo de tempo. Com o intuito de aumentar a autonomia do time e diminuir o número de mensagens necessárias para trocar informações modificamos a abordagem de cooperação, permitindo decisões locais em cada robô e reduzindo a comunicação no time. O modelo final foi testado tanto em simulação quanto em experimentos em cenários reais utilizando um time de robôs e-puck, resultando em um sistema de navegação mais robusto e preciso para robôs autônomos.

O quarto ponto diz respeito à estratégia de reconfiguração do time, quando um ou mais robôs são forçados a quebrar a formação para poderem desviar de obstáculos em seu trajeto. Por um lado, o método descrito em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) lança mão de uma troca de eixos de deslocamento quando existe uma aproximação entre o robô central e um dos robôs laterais. Além disso, existe uma inversão de responsabilidades (mestre/escravo) entre os dois robôs envolvidos nessa troca. Por outro lado, em (FERREIRA, 2014) a estratégia na situação de desvio por parte de um dos membros do time é fazer o restante dos robôs aguardar que o obstáculo seja contornado completamente para que o time volte a caminhar em formação, sem a troca dos eixos de deslocamento ou das responsabilidades. Após analisarmos as duas estratégias, concluímos que existiam situações que favoreciam a adoção de uma ou outra. Assim, elaboramos um AG para evoluir uma estratégia de decisão de quando deve ser feita uma reconfiguração nas posições dos robôs para que a formação seja retomada, ou quando os robôs devem manter a configuração apenas aguardando que o robô desvie do obstáculo. Além disso, observamos que existiam outros parâmetros de movimentação do time (velocidade) que interferiam na manutenção da formação pelo maior tempo possível durante a navegação do time. Esses parâmetros também foram otimizados pelo AG.

1.5 Organização da Dissertação

Esta dissertação foi organizada em seis capítulos, como descrito a seguir. No presente capítulo apresentamos a introdução sobre o tema a ser investigado, ressaltando nossa motivação para o estudo do tema, nossos objetivos com a pesquisa, hipóteses e contribuições. O segundo capítulo refere-se à fundamentação teórica, iniciando com uma descrição das principais características dos ACs. Depois, descrevemos o conceito e as várias fases de um AG. Por fim, fez-se uma revisão da área de robótica autônoma, descrevendo os principais conceitos da área e introduzindo o simulador Webots e os robôs e-puck. A seguir, no terceiro capítulo, apresentamos uma revisão da literatura correlata, analisando-se diversos trabalhos na área, escolhidos principalmente por sua aderência ao tema e por utilizarem alguma técnica evolutiva ou um modelo baseado em Autômato Celular. Em nossa análise focamos na técnica utilizada, os objetivos do trabalho, o tratamento quanto a obstáculos, implementação em simulação, experimentos e plataforma utilizada com robôs reais.

No capítulo 4, segmentamos nossas modificações para o modelo aplicado a somente um robô, descrevendo a inclusão de cinco novos estados ao AC, cenários de construção da vizinhança e novos conjuntos de regras de atualização locais. Também descrevemos o novo método para classificação automática da vizinhança utilizando um AG. Na segunda parte do capítulo, focamos nas otimizações do modelo quanto as características cooperativas do time. Primeiramente, descrevemos a nova estrutura de comunicação entre robôs, com o protocolo Bluetooth e o protocolo Wifi. Depois, introduzimos nossa proposta de

otimização bioinspirada da reconfiguração da formação, descrevendo um novo AG para encontrar o melhor conjunto de parâmetros para manutenção da formação quando em presença de obstáculos.

Os resultados e análises são exibidos no capítulo 5, onde demonstramos a utilização do modelo proposto em ambientes no simulador Webots e experimentos com robôs e-puck, para a tarefa de controle de formação em cenários com obstáculos.

O sexto e último capítulo discorre sobre as principais contribuições e conclusões deste trabalho, além de propor sugestões para trabalhos futuros visando a continuidade da pesquisa para robôs móveis utilizando um AC em conjunto com estratégias evolutivas.

Capítulo 2

Fundamentação Teórica

Neste capítulo, serão definidos os principais conceitos necessários para o entendimento e desenvolvimento do trabalho proposto. A teoria apresentada a seguir está dividida com base nos pontos-chaves que fazem parte de nossa abordagem: ACs, AGs e robótica autônoma.

2.1 Autômatos Celulares (ACs)

Ulam e von Neumann introduziram os ACs na década de 50 e tinham como objetivo projetar mecanismos artificiais de auto reprodução (WOLFRAM, 1983b). Um AC é uma estrutura computacional que apresenta as seguintes características:

- Tempo, espaço e estados discretos;
- Reticulado n-dimensional (espaço celular) de componentes idênticos (células);
- Conectividade local (conceito de vizinhança de células);
- Regra de transição determinística e síncrona;
- Condições de contorno para definir as vizinhanças nas bordas do reticulado.

Cada célula possui uma vizinhança, que é geralmente composta pela célula analisada e seus vizinhos mais próximos. A partir dessa vizinhança uma regra de transição é aplicada e produz o próximo estado da célula em questão (célula central da vizinhança), sendo que esse processo é feito de forma sincronizada em todas as células do reticulado, contando-se um passo de tempo. A evolução temporal do AC se dá pela aplicação das regras de transição por vários passos de tempo (OLIVEIRA, 2003).

Segundo a notação formal adotada por (MITCHELL, 1996), temos:

- Σ é o conjunto de todos os estados possíveis em cada célula, e $\boldsymbol{k}$ é o tamanho do conjunto Σ;

- Para cada célula é atribuído um índice $\boldsymbol{i}$ e em um dado tempo $\boldsymbol{t}$ seu estado é definido por S_i^t;
- O estado da célula $\boldsymbol{i}$ junto com os estados das células vizinhas são chamados de vizinhança da célula $\boldsymbol{i}$, sendo denotada por $\eta^{t,i}$;
- A regra de transição de estados é representada por φ;
- Ao aplicar $\varphi(\eta_i)$ obtém-se o estado $S^{t+1,i}$ para a célula $\boldsymbol{i}$;
- Uma unidade de tempo $\boldsymbol{t}$ consiste em aplicar $\varphi(\eta_i)$ em todas as células.

Algumas características interessantes dos ACs são que: *(i)* eles têm natureza paralela e custo computacional baixo; *(ii)* o caráter discreto permite maior portabilidade entre cenários de simulação e cenários reais; *(iii)* possuem comportamento simples baseado em regras e não precisam de informação completa do ambiente. Dessa forma, os ACs se mostraram uma estratégia promissora para tratar o problema de planejamento de caminhos e controle de formação em times de robôs autônomos (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), (FERREIRA; VARGAS; OLIVEIRA, 2014) e (OLIVEIRA; VARGAS; FERREIRA, 2015).

2.1.1 Autômatos Celulares Unidimensionais

O reticulado desse tipo de autômato normalmente é representado por um vetor (de 0s e 1s, no caso binário), no qual cada posição representa uma célula. Nos ACs unidimensionais, o tamanho da vizinhança é normalmente escrito como $\boldsymbol{m} = 2\boldsymbol{r} + 1$, onde $\boldsymbol{r}$ é chamado de raio do AC. A Figura 1 apresenta um exemplo de reticulado binário que utiliza a vizinhança de raio 1 ($\boldsymbol{m} = 3$).

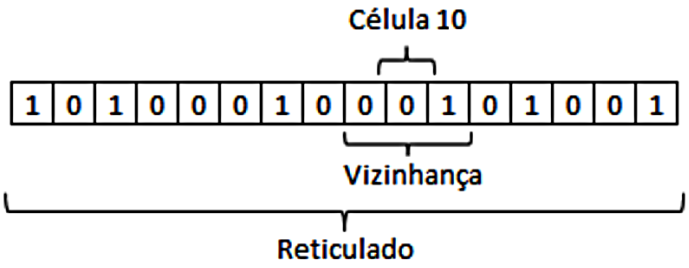

Figura 1 – Reticulado unidimensional e vizinhança com raio 1.

Antes de aplicar a regra de transição é necessário definir a condição de contorno do reticulado, que se refere à vizinhança das células nas extremidades. Uma das condições de contorno mais utilizadas é a periódica, vista na Figura 2, na qual as extremidades do reticulado são conectadas formando um anel.

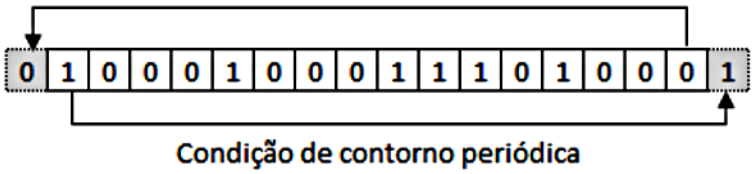

Figura 2 – Exemplo de vizinhança periódica. (JÚNIOR, 2010).

O número de vizinhanças possíveis é dado por $\boldsymbol{k^m}$; no caso do exemplo anterior, temos $\boldsymbol{k} = 2$ e $\boldsymbol{m} = 3$, então o número de vizinhanças é $2^3 = 8$. A regra de transição do AC define o estado da célula central de acordo com cada uma das vizinhanças possíveis. A Figura 3 mostra o estado da célula central no próximo passo de tempo de acordo a vizinhança.

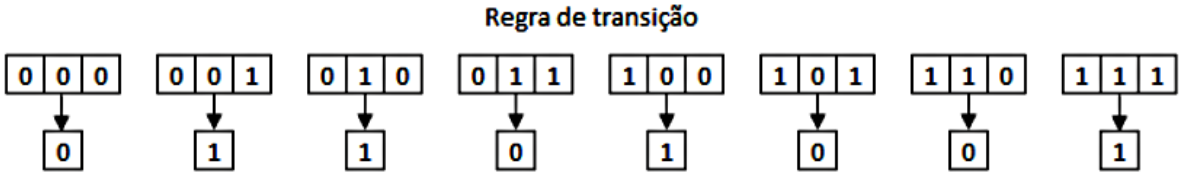

Figura 3 – Exemplo de regra de transição (JÚNIOR, 2010).

A partir do estado das células do reticulado em um tempo inicial *(t0)*, a evolução temporal acontece aplicando a regra de transição de forma sucessiva. A Figura 4 mostra a evolução temporal, durante um passo de tempo, do reticulado apresentado na Figura 1 usando a condição de contorno periódica e a regra de transição da Figura 3.

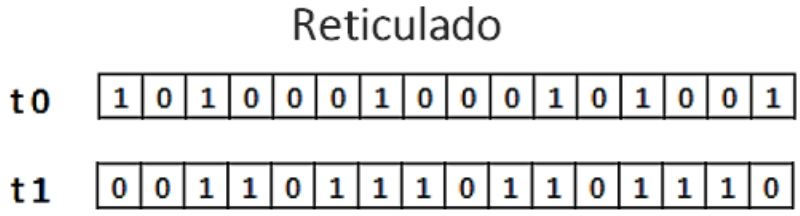

Figura 4 – Evolução do reticulado após aplicação da regra de transição.

Existem 256 ACs unidimensionais, com $\boldsymbol{k} = 2$ e $\boldsymbol{r} = 1$, esses são chamados de autômatos celulares elementares. Os ACs também podem ter mais de uma dimensão sendo o mais comum o caso bidimensional, descritos a seguir.

2.1.2 Autômatos Celulares Bidimensionais

Os ACs bidimensionais são similares aos unidimensionais, onde a principal diferença é o espaço celular. Geralmente, são representados por uma matriz de 0s e 1s, na qual cada

posição representa um célula. A vizinhança não se restringe apenas às células vizinhas da direita e da esquerda, sendo que as mais conhecidas são as vizinhanças de von Neumann e de Moore, mostradas na Figura 5. Na vizinhança de von Neumann, apresentada na Figura 5(a), apenas as células ao norte, sul, leste e oeste são consideradas. Na vizinhança de Moore, representada na Figura 5(b), as células nordeste, noroeste, sudeste e sudoeste são incluídas.

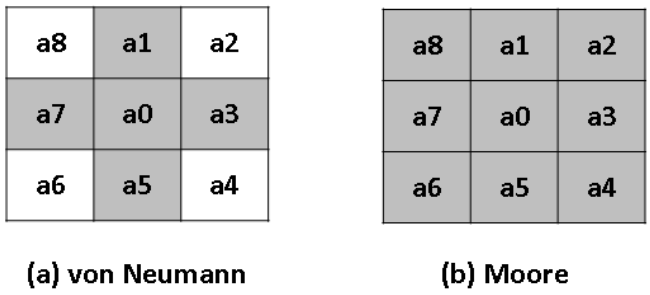

Figura 5 – Exemplo de tipos de vizinhança.

A condição de contorno periódica no AC bidimensional conecta as linhas extremas do reticulado (norte e sul) e também as colunas (leste e oeste), deste modo a evolução temporal também fica similar aos ACs unidimensionais. Veja, por exemplo, na evolução temporal do AC na Figura 6, a vizinhança de von Neumann de raio 1 foi utilizada. A parte superior da figura mostra 32 saídas ligadas a cada possível vizinhança, enquanto a parte inferior mostra a configuração do reticulado para 2 passos de tempo consecutivos. Por exemplo, considerando-se a célula na segunda linha e quinta coluna do reticulado t0, podemos observar a mudança de seu estado no reticulado t2. Neste caso, de acordo com a vizinhança e a respectiva regra de transição, o estado da célula central passou de 0 para 1.

Por se tratarem de estruturas computacionais de implementação extremamente simples, mas capazes de exibir uma dinâmica complexa, os ACs se tornaram importantes ferramentas para o estudo de sistemas naturais (OLIVEIRA, 2003). Alguns exemplos de aplicações que podem ser investigadas utilizando ACs são: simulação de tráfego urbano (WAHLE et al., 2001); escoamento de fluidos (FRISCH; HASSLACHER; POMEAU, 1986); propagação de incêndios em florestas (GREEN; TRIDGELL; GILL, 1990); desenvolvimento urbano (WARD; MURRAY; PHINN, 2000); formação de padrões em conchas e peles de animais (WOLFRAM, 2002); planejamento de caminhos (AHMED; AKHTER; KUNWAR, 2012); controle de formação de times de robôs cooperativos (FERREIRA, 2014). A estrutura de regras de transição permite também sua utilização em conjunto com métodos de otimização, como por exemplo os AGs.

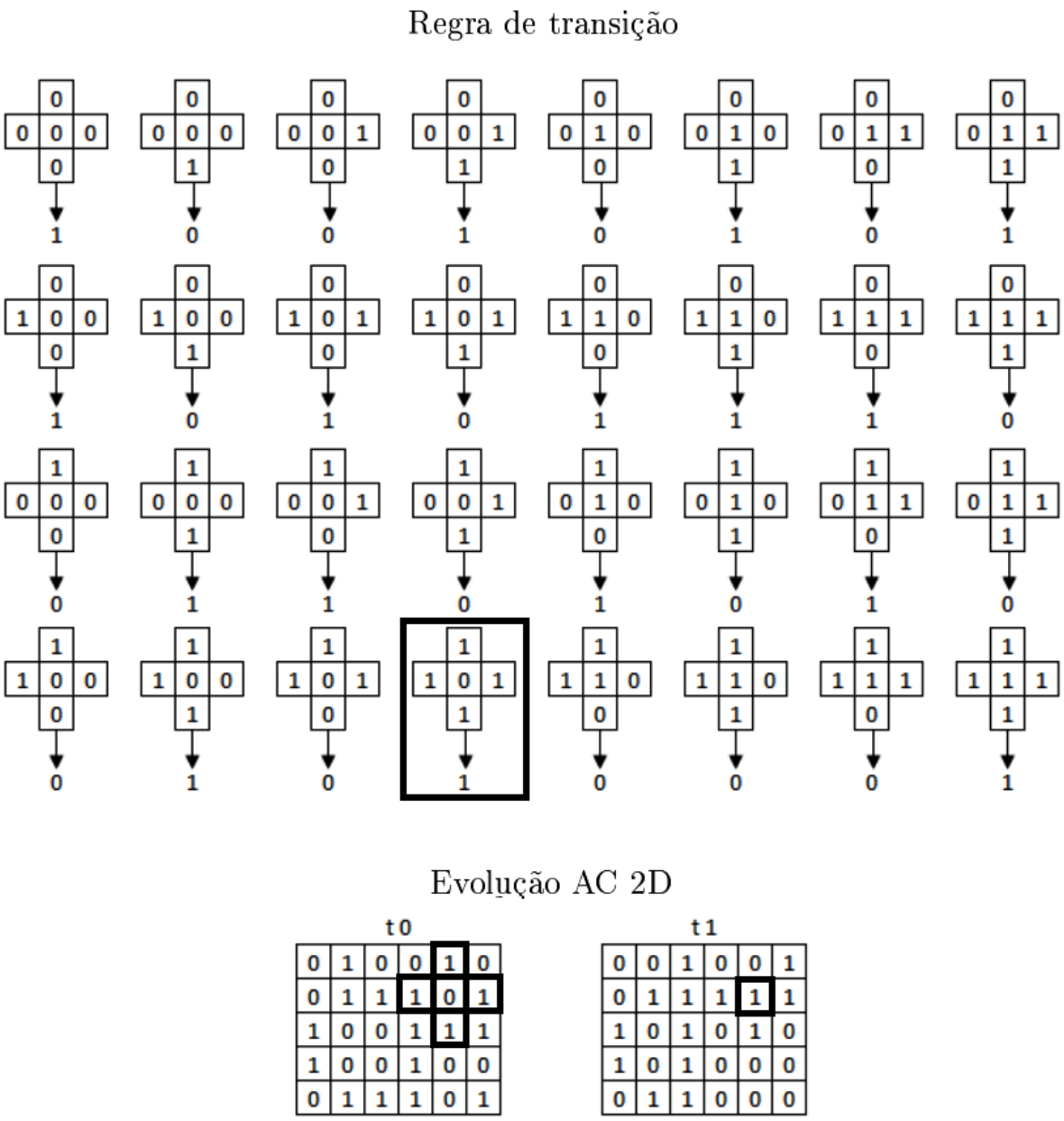

Figura 6 – Evolução do reticulado após aplicação da regra de transição em AC bidimensional (JÚNIOR, 2010).

2.2 Algoritmos Genéticos (AGs)

Os AGs são definidos como métodos de busca e otimização baseados em evolução natural e mecanismos genéticos (GOLDBERG, 1989), simulando a teoria da evolução proposta por (DARWIN, 1859). Os AGs fazem parte da área da Inteligência Artificial (IA), proveniente da interseção entre a Biologia Evolutiva e a Ciência da Computação, denominada Computação Evolutiva (CE).

A estratégia empregada nos AGs trabalha com o conceito de grupo de indivíduos, também chamado de população, que representa uma amostra de possíveis soluções. Com

o objetivo de evoluir a população para áreas mais promissoras do espaço de busca e encontrar soluções melhores a cada geração, combina técnicas de reprodução, operadores genéticos e função de aptidão (GOLDBERG, 1989).

Alguns termos inerentes à estrutura do algoritmo e do processo evolutivo são descritos a seguir:

- Espaço de Busca - conjunto ou espaço que compreende as possíveis soluções para o problema a ser resolvido;
- Cromossomo - ponto no espaço de busca que representa uma possível solução para o problema;
- Gene - unidade básica que compõe o cromossomo, representando uma variável do problema;
- Aptidão - valor que representa o quão bom é o cromossomo para solução do problema;
- Indivíduo - união do cromossomo e sua aptidão;
- População - conjunto de indivíduos;
- Operadores Genéticos - introduzem variabilidade genética aos cromossomos, visando promover a evolução do indivíduo. Os mais comuns são *crossover* e mutação;
- Geração - Cada ciclo do processo evolutivo descrito pelo algoritmo;
- Reprodução - Estratégia que decide quais indivíduos passarão para a próxima geração.

O ciclo básico de execução de um AG pode ser visto na Figura 7. O ideal é que o AG pare quando encontrar a melhor solução, porém, não se pode afirmar que a solução encontrada representa o ótimo global ou se é somente um ótimo local (KOZA, 1992). Existem vários parâmetros que podem ser utilizados como critérios de parada, como a estagnação da população, encontrar uma solução satisfatória, atingir um número máximo de gerações ou o tempo limite de processamento. Para melhor entendimento do processo, descreveremos a seguir as principais etapas no desenvolvimento de um AG.

2.2.1 Representação dos indivíduos

A forma de representar e codificar cada ponto do espaço de busca varia de acordo com o problema, mas algumas técnicas já são bastante difundidas e podem ser aplicadas caso se adequem ao problema em questão.

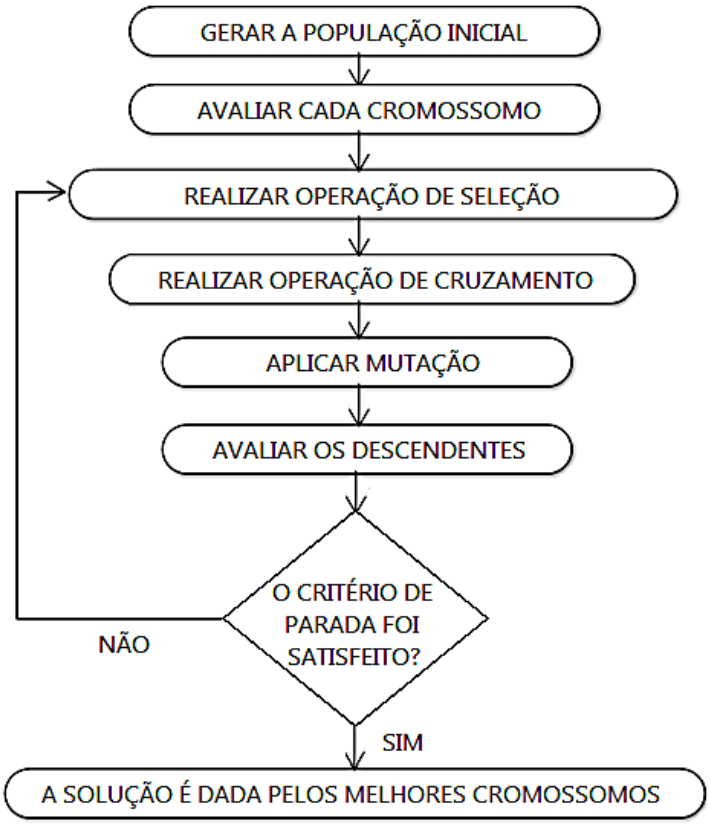

Figura 7 – Etapas do processo evolutivo de um AG.

1	0	1	1	0	0	1	0

Figura 8 – Exemplo de cromomossomo binário.

Na representação binária cada cromossomo é formado por uma cadeia de bits, em que cada gene pode assumir apenas os valores 1 ou 0, como ilustrado na Figura 8.

A representação com base na ordem é indicada para problemas mais complexos, que utilizam informações históricas e sequenciais. Os números representam os caminhos e a ordem é gerada pela execução das ações no algoritmo, a Figura 9 mostra um exemplo desse tipo de representação.

3	5	1	7	4	2	6	8

Figura 9 – Exemplo de representação com base na ordem.

Caso as características do problema não possam ser discretizadas em valores binários,

é possível representar diretamente os valores reais das variáveis envolvidas. A Figura 10 exemplifica esse tipo de representação.

2,4	7,5	1,3	4,7	9,1	2,8	6,6	8,9

Figura 10 – Exemplo de representação por valores.

2.2.2 Função de Avaliação

Por meio da função de avaliação calcula-se o valor de aptidão de cada indivíduo da população e se mede o quão próximo um indivíduo está da solução desejada ou quão boa é esta solução.

A definição de como será feito esse cálculo está intrinsecamente ligada ao problema a ser resolvido, porém a precisão da avaliação é de suma importância, pois a má diferenciação entre boas e más soluções pode direcionar o algoritmo para caminhos diferentes, fazendo com que soluções interessantes possam ser perdidas.

Um exemplo de função de avaliação para a representação binária é a soma de todos os valores contidos nos genes. Se 0 representa a ausência de certa característica e 1 representa a presença, o melhor indivíduo seria o que apresentasse o maior valor para a função de avaliação (caso o objetivo fosse encontrar um indivíduo que apresentasse todas as características analisadas). A Figura 11 ilustra essa situação, na qual o cromossomo 2 seria considerado melhor que o cromossomo 1, pois apresenta uma aptidão maior.

								Aptidão
Cromossomo 1	1	0	1	1	0	0	1	0 = 4
Cromossomo 2	1	1	1	1	1	1	1	1 = 8

Figura 11 – Exemplo de aplicação da função de avaliação.

2.2.3 Seleção dos Indivíduos

Como os AGs baseiam-se no princípio da seleção natural eles devem ser capazes de identificar os melhores e os piores indivíduos, de modo que isso faça com que eles permaneçam ou não no processo evolutivo (BERNARDINI, 2006).

A ideia principal do operador de seleção em um AG é oferecer, aos indivíduos com melhores índices de aptidão, preferência para o processo de reprodução, permitindo que

estes indivíduos passem as suas características às próximas gerações. Assim, indivíduos cada vez mais aptos serão produzidos, enquanto indivíduos menos aptos tendem a desaparecer. Para realizar essa escolha, utiliza-se o valor de aptidão de cada cromossomo. A seguir, serão apresentados e brevemente explicados três dos principais métodos de seleção: roleta, torneio e *ranking*.

Na seleção por roleta, cria-se uma roleta virtual, como podemos observar na Figura 12, que representa a soma das aptidões dos indivíduos da população. Cada indivíduo recebe uma fatia proporcional à sua aptidão, ou seja, os indivíduos mais aptos terão mais chances de serem sorteados. Um número, entre 1 e a soma das aptidões, é sorteado pelo algoritmo representando a rotação da roleta, o setor no qual esse número estiver será o cromossomo selecionado. Na Figura 12, o indivíduo 3 obteve a maior porção da roleta devido sua aptidão ser maior, o indivíduo 2 por sua vez ficou com a menor fatia, desta forma a chance do ponto de seleção parar no setor onde se encontra o indivíduo 2 se reduz.

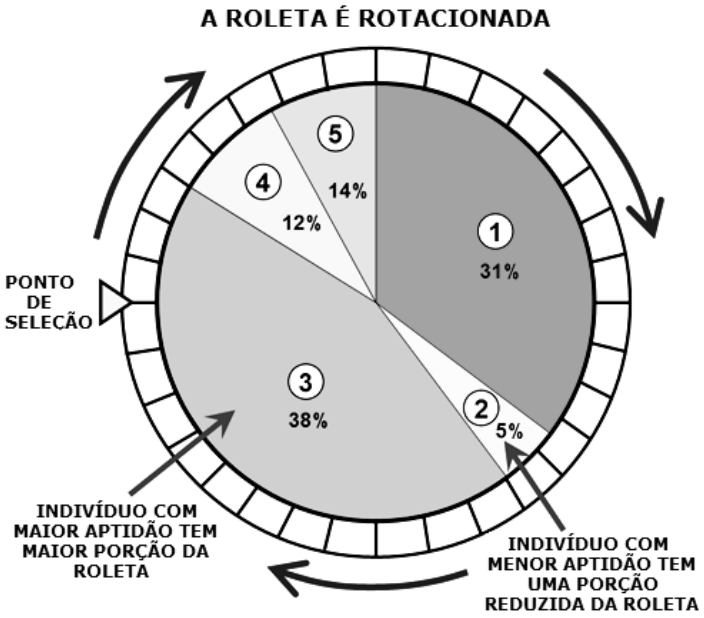

Figura 12 – Exemplo de aplicação da roleta (GONZALES; ALBERTO, 2011).

O processo de seleção por torneio leva em consideração uma variável chamada *tour*. A cada torneio um subgrupo de *tour* indivíduos é selecionado e deste subgrupo o melhor é escolhido. Esse tipo de seleção pode ser feita de forma simples ou estocástica. No torneio simples, o subgrupo de indivíduos é selecionado aleatoriamente, enquanto, no torneio estocástico, cada indivíduo do subgrupo é selecionado através do método da roleta.

Existem outros métodos de seleção, tais como a seleção por classificação, na qual a probabilidade de seleção é relacionada à posição no ranking de ordenação dos indivíduos da população por aptidão, ou seja, primeiro classifica-se a população e então atribui a cada indivíduo um valor de adequação determinado pela sua classificação no ranking. O

pior terá adequação igual a 1, o segundo pior 2, de forma que o melhor terá adequação igual a N (número de indivíduos na população).

2.2.4 Operadores Genéticos

Os operadores genéticos são responsáveis por transformar a população através de sucessivas gerações, fornecendo variabilidade genética aos indivíduos.

A troca de material genético entre dois indivíduos (pais) é feita por meio do operador *crossover*, no qual segmentos de um indivíduo são trocados pelo trecho equivalente de outro indivíduo. Os principais tipos de recombinação genética para indivíduos com representação binária são o *single-point crossover* e o *multi-point crossover*. No primeiro tipo, o cruzamento é feito conforme mostrado na Figura 13. O ponto de cisão escolhido foi entre o terceiro e o quarto gene dos cromossomos selecionados, e após a recombinação genética houve a geração de dois novos indivíduos, os quais possuem segmentos dos dois progenitores.

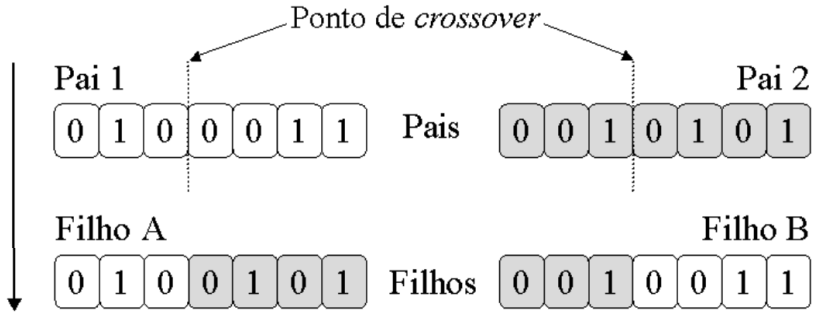

Figura 13 – Exemplo de *crossover* simples (SOUTO, 2003).

No *multi-point crossover* são selecionados mais de um ponto de cisão e a troca de material genético é feita nesses vários pontos. A Figura 14 apresenta um *crossover* de 2 pontos. Nesse tipo de *crossover* um dos descendentes fica com a parte central de um dos pais e extremidades do outro pai; o contrário ocorre para o outro filho.

Outra forma de introduzir variabilidade genética na população é através do operador de mutação, que consiste em alterar arbitrariamente um ou mais genes de um indivíduo. A Figura 15 ilustra o processo de mutação para um indivíduo binário. O quarto gene do indivíduo foi escolhido para a mutação. Como a representação cromossômica é binária, a operação simplesmente trocou o 1 pelo 0.

2.2.5 Reprodução

As estratégias de reprodução decidem quais serão os indivíduos que continuarão no processo de evolução. As duas principais estratégias são elitismo e sobrevivência dos

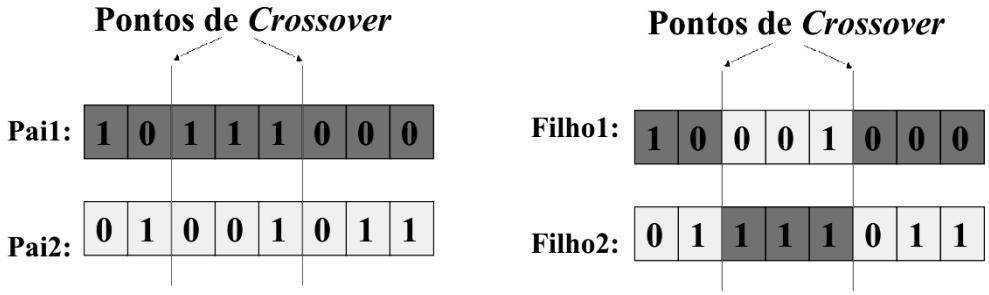

Figura 14 – Exemplo de *crossover* duplo

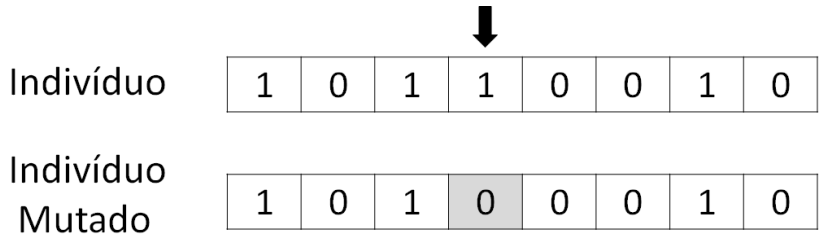

Figura 15 – Exemplo de aplicação do operador mutação

melhores. No elitismo uma porcentagem dos melhores indivíduos da geração pai é conservada para a próxima geração sem nenhuma alteração, impedindo assim que boas soluções desapareçam. A abordagem de sobrevivência dos melhores considera a população total (pais + filhos) e ordena os indivíduos com base em sua aptidão, permitindo que somente os melhores sigam para a próxima geração.

Esse processo de reprodução e evolução dos melhores indivíduos propicia a otimização da população, o resultado de soluções cada vez melhores a cada geração pode ser muito interessante quando utilizados em conjuntos de parâmetros de times de robôs.

2.3 Robótica Autônoma

O estudo de sistemas com a capacidade de executar determinadas tarefas de forma independente e adaptativa é uma das áreas mais fortes em robótica. Nesse contexto, um robô autônomo é definido como um sistema que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos (MATARIĆ, 2007).

Robôs autônomos podem ser utilizados em diversas tarefas, tais como: ajudar pessoas idosas em situações cotididianas, guiar pessoas em fábricas ou museus, desarmar bombas, limpar casas e piscinas, explorar superfícies de planetas (CABREIRA; AGUIAR; DIMURO, 2013). Entretanto, o grande desafio é projetar algoritmos de controle para que os robôs sejam capazes de se adaptar em ambientes totalmente desestruturados, dinâmicos e parcialmente observáveis (SUKHATME; MATARIC, 2002). Um controlador, no

âmbito da robótica autônoma, é considerado um processo que mapeia entradas (obtidas através dos sensores do robô) em saídas, que se traduzem em ações por meio de atuadores ou mudanças de estado (HAASDIJK A. E. EIBEN, 2011).

Para o estudo e desenvolvimento de métodos de controle geralmente é necessário definir um modelo de ambiente mais fácil de trabalhar, ou seja, um modelo onde o robô, o mundo e suas interações são feitas de forma mais simples, e para isso são utilizadas ferramentas bastante importantes na área de robótica, chamados de simuladores (TIKHANOFF et al., 2008).

2.3.1 Simulador Webots

Em nosso trabalho utilizamos o simulador Webots (CYBERBOTICS, 2015) para realizar nossos experimentos, pois ele é uma plataforma de simulação que permite modelagens realísticas de robôs e ambientes, incluindo parâmetros de diferentes sensores e variáveis físicas do mundo real (DASGUPTA; WHIPPLE; CHENG, 2011).

No geral, o simulador oferece uma forma prática e rápida de prototipação, os ambientes são facilmente configuráveis e a precisão das variáveis físicas permite a construção de controladores transferíveis a robôs reais. Para usufruir da flexibilidade provida pelo simulador foi necessária a obtenção de uma licença EDU, deste modo, conseguimos reduzir o tempo de desenvolvimento utilizando a modelagem do robô e-puck já fornecida pelo Webots. O ambiente de simulação pode ser visualizado na Figura 16. Do lado esquerdo da figura, podemos ver a modelagem do ambiente e de um robô e-puck, o lado direito é destinado à codificação do controlador do robô e abaixo fica o console de saída dos resultados.

Apesar da alta precisão e níveis de detalhes da simulação, o código desenvolvido em simuladores pode apresentar comportamentos diferentes quando executado em experimentos no mundo real, principalmente quando se trata de algoritmos de coordenação de times cooperativos (PUGH; MARTINOLI, 2007), (BERG; KARUD, 2011). Portanto é essencial utilizar técnicas que reduzam essa inconformidade e validar o método com robôs em diferentes cenários reais.

2.3.2 Robôs e-puck

Para realizar os testes em cenários reais usamos os pequenos robôs e-puck (LAUSANNE, 2015), uma vez que inúmeros trabalhos já utilizaram essa arquitetura e a mesma se mostrou adequada para a aplicação do nosso método. Os robôs e-puck são equipados com uma variedade de sensores e sua estrutura de hardware é considerada simples, sendo adaptado para experimentos com vários agentes (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011a) (ASL; MENHAJ; SAJEDIN, 2014).

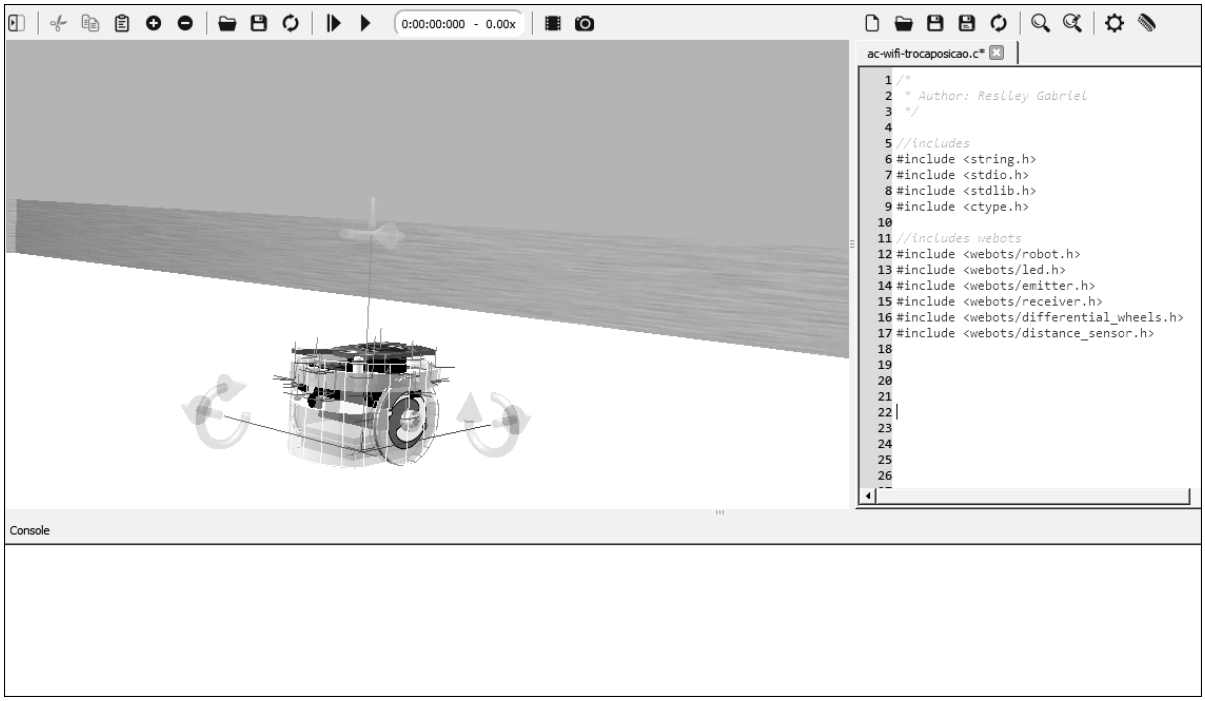

Figura 16 – Tela principal do simulador Webots.

Os robôs e-puck são suportados por um compilador C customizado e são equipados com um micro controlador dsPIC integrado com 8kB de RAM e 144kB de memória flash. Outras características são apresentadas na Figura 17, como por exemplo: (*i*) câmera com resolução de 640x480 e capacidade de captura de imagens em 4 quadros por segundo, (*ii*) alto-falante que emite diferentes tipos de som e (*iii*) o anel de LEDs que pode ser utilizado para retornar sinais visuais para determinadas ações.

Figura 17 – Exemplar de um robô e-puck.

A configuração básica presente nos robôs fornece um alto nível de aplicabilidade, tendo

como características adicionais o custo baixo, tamanho pequeno e estrutura de código aberto. Todos esses fatores fazem com que o e-puck seja uma das plataformas robóticas mais usadas em laboratórios de pesquisa (COUCEIRO; VARGAS; ROCHA, 2014).

Uma funcionalidade importante para a navegação é a capacidade de identificar as distâncias entre o robô e outro objeto. Nos robôs e-puck isso é feito utilizando-se os sensores infravermelho, porém, conforme ilustrado na Figura 18, os 8 sensores existentes estão distribuídos de forma não constante ao redor do robô, o que pode diminuir a acurácia da identificação de objetos próximos. Existe uma concentração de sensores na parte frontal do robô (sensores *ps0*, *ps1*, *ps7* e *ps6*), o que permite uma melhor detecção de obstáculos nessa área. Quando se trata das laterais, a estrutura do e-puck só apresenta um sensor para cada lado (*ps5* na esquerda e *ps2* na direita). Na parte traseira, somente dois sensores estão presentes (*ps3* e *ps4*). Dessa forma, a detecção de obstáculos laterais e nas diagonais traseiras fica prejudicada. Essa limitação, no entanto, pode ser superada utilizando técnicas em software que auxiliam na detecção de colisões.

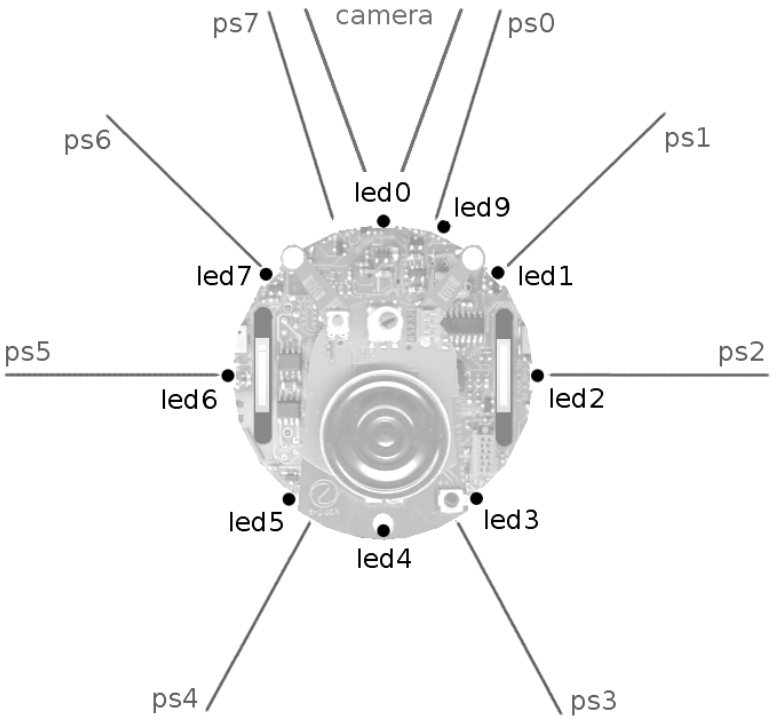

Figura 18 – Sensores de distância (ps0 a ps7) dos robôs e-puck.

Capítulo 3

Revisão da Literatura Correlata

O objetivo deste capítulo é evidenciar e analisar alguns dos principais trabalhos encontrados na literatura relacionados a planejamento de caminhos e controle de formação, avaliando suas características mais importantes.

As abordagens criadas visando o planejamento de trajetória e controle de formação em times de robôs cooperativos apresentam aspectos que influenciam diretamente nos resultados e na aplicabilidade em cenários reais. Entre tais fatores, pode-se citar a técnica utilizada para manter a formação enquanto percorre o ambiente, adaptabilidade a diversos tipos de obstáculos e a forma como é feita a comunicação entre os robôs.

3.1 Métodos que Não Empregam Modelos Baseados em ACs

A seguir faremos a análise de alguns trabalhos publicados na área, que abordam o planejamento de caminhos com controle de formação, porém sem a utilização de ACs.

Os autores em (BALCH; ARKIN, 1998) implementaram um modelo de comportamentos reativos com o objetivo de fazer com que o time de robôs fosse capaz de navegar pelo ambiente, desviando de obstáculos, enquanto mantinha formação. No trabalho foram utilizados quatro tipos de formação (linha, coluna, diamante e em "v") e cada robô tinha uma posição definida na formação baseada em seu número de identificação. A manutenção da formação é obtida em duas etapas, a primeira é a detecção de posicionamento baseada nas leituras dos sensores dos robôs e a segunda é a movimentação do robô para sua posição correta. Para determinar sua posição na formação os robôs podem utilizar três abordagens diferentes: (*i*) fazer a soma das coordenadas de todos os robôs e ajustar sua posição em relação ao centro da formação, (*ii*) utilizar a referência de posicionamento de um robô líder e segui-lo pelo ambiente ou (*iii*) manter seu posicionamento baseando-se na posição dos robôs vizinhos mais próximos. Para se movimentar pelo ambiente os robôs utilizam quatro comportamentos básicos: (*i*) mover em direção à meta, (*ii*) evitar

obstáculo estático, (*iii*) evitar outro robô e (*iv*) manter formação. Os experimentos foram realizados em simulação, com robôs em laboratório e com veículos terrestres militares.

O método proposto em (DAS et al., 2002) utiliza a abordagem líder-seguidores com o objetivo de realizar o controle da formação em um time de robôs. Para permitir mudanças na formação, o método alterna entre diferentes controladores descentralizados simples e usa a informação de um único tipo de sensor, uma câmera omnidirecional, para todos os esses controladores. A relação entre os robôs é definida por um grafo direcionado não cíclico, no qual o líder é o nó que só recebe arestas. Desta forma, o movimento dos robôs dentro da formação é definido em referência ao robô líder. Os experimentos foram realizados utilizando robôs Clodbuster, os sinais de vídeo da câmera câmera omnidirecional são enviados para um computador remoto e os sinais de velocidade e posição de controle são enviados a partir do computador para os robôs.

No trabalho (DUAN et al., 2013), o método proposto utiliza otimização por enxame de partículas (PSO) e um AG com objetivo de encontrar um conjunto de comandos de controle e minimizar o tempo para que, dada uma formação inicial qualquer, os veículos aéreos não tripulados (VANTS) sejam capazes de atingir uma determinada formação final, como pode ser visto na Figura 19.

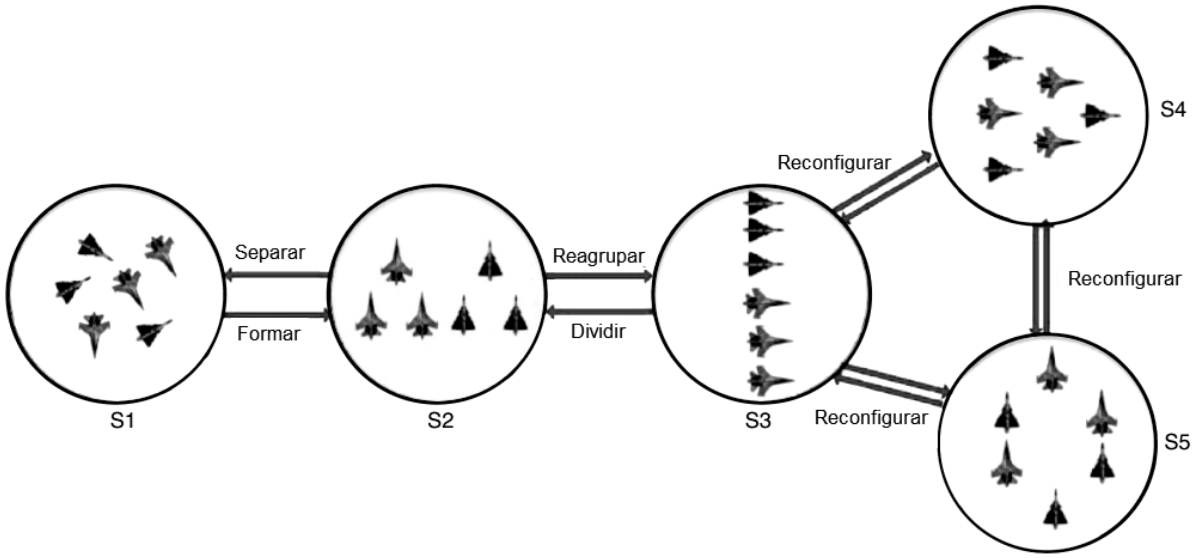

Figura 19 – Formação e reconfiguração de um grupo de VANTS (DUAN et al., 2013).

Caso haja falha de comunicação ou falha de hardware a formação é reconfigurada para manter a coesão. A população inicial, que contém as instruções para que os VANTS se organizem em formação, é repartida em duas, a primeira parte é otimizada através do algoritmo PSO e a segunda por um AG. Os dois algoritmos executam em paralelo e ao final as populações são unidas para encontrar a melhor solução. A distância mínima para evitar colisão e a distância máxima para que não haja perda de comunicação são inseridas como restrições. A Figura 20 mostra o comportamento dos VANTS em relação à distância

entre eles, onde distância entre qualquer um dos VANTS sempre respeita as restrições de comunicação e colisão. O método consegue tratar obstáculos estáticos e dinâmicos, porém foi testado somente em simulação.

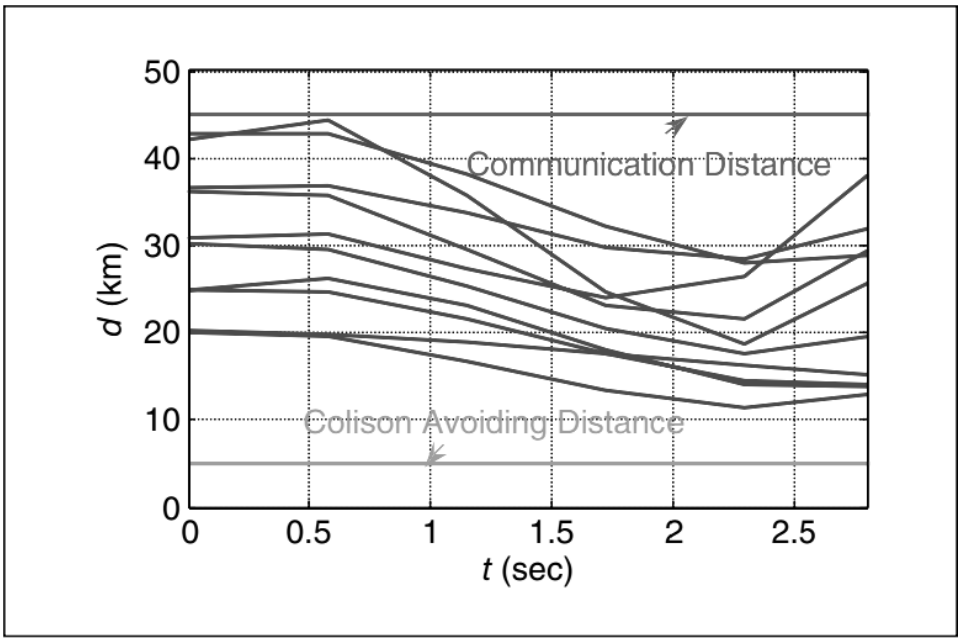

Figura 20 – Limite de distância de comunicação e colisão entre os VANTS (DUAN et al., 2013).

Uma abordagem baseada em morfogênese e rede de regulação genética foi proposta em (MENG; GUO; JIN, 2013) com o objetivo de fazer com que os robôs atinjam determinada formação fixa. No trabalho, cada robô é visto como uma célula contendo DNA virtual na qual a formação está codificada. Apesar da formação desejada ser repassada a cada robô, eles não têm conhecimento de todo o ambiente nem posição fixa, sendo assim, eles devem descobrir seu posicionamento através de processamento e comunicação local. Utiliza-se um modelo não-uniforme racional B-spline (NURBS) para representar formas complexas a serem construídas pelos robôs. A Figura 21 apresenta um exemplo de formação, no qual os robôs e-puck são capazes de gerar a formação no formato da letra R

Os robôs podem conter (em sua representação no método) dois tipos de proteínas (P e G) e elas contém instâncias que representam as coordenadas x e y do robô. Um morfogene (que representa a formação desejada) controla a produção de cada uma das proteínas. Foi utilizado o algoritmo NSGA-II (DEB et al., 2002) para otimizar a distância percorrida pelos robôs e o tempo para que a formação fosse atingida. Obstáculos estáticos e dinâmicos são tratados pelo método, que foi testado em simulação e em cenários reais. A Figura 22 mostra o processo de desvio de obstáculo em uma formação. A medida que o obstáculo se aproxima da formação os robôs se separam para evitar a colisão e depois retomam suas posições.

A vizinhança é adaptável, como cada robô não tem conhecimento do número total de

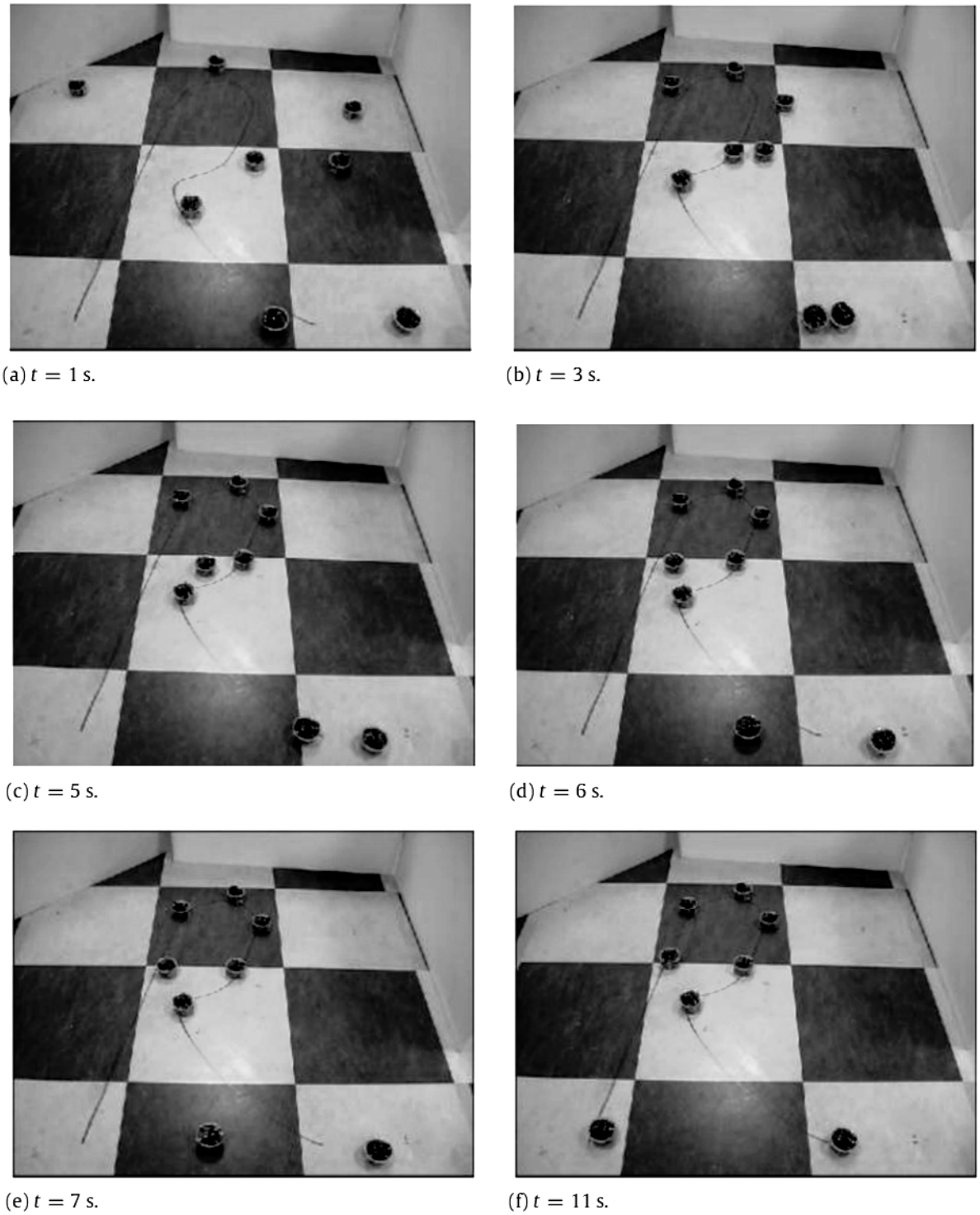

Figura 21 – Exemplo de formação (MENG; GUO; JIN, 2013).

robôs, primeiramente é definida uma vizinhança arbitrária e com base no perímetro da formação são definidos pontos nos quais os robôs devem se posicionar. Se a vizinhança for muito grande pode ser que o número de pontos seja maior que o número de robôs, se for pequena haverá robôs competindo pelo mesmo ponto, ficando a cargo dos robôs se comunicarem até encontrar uma vizinhança satisfatória.

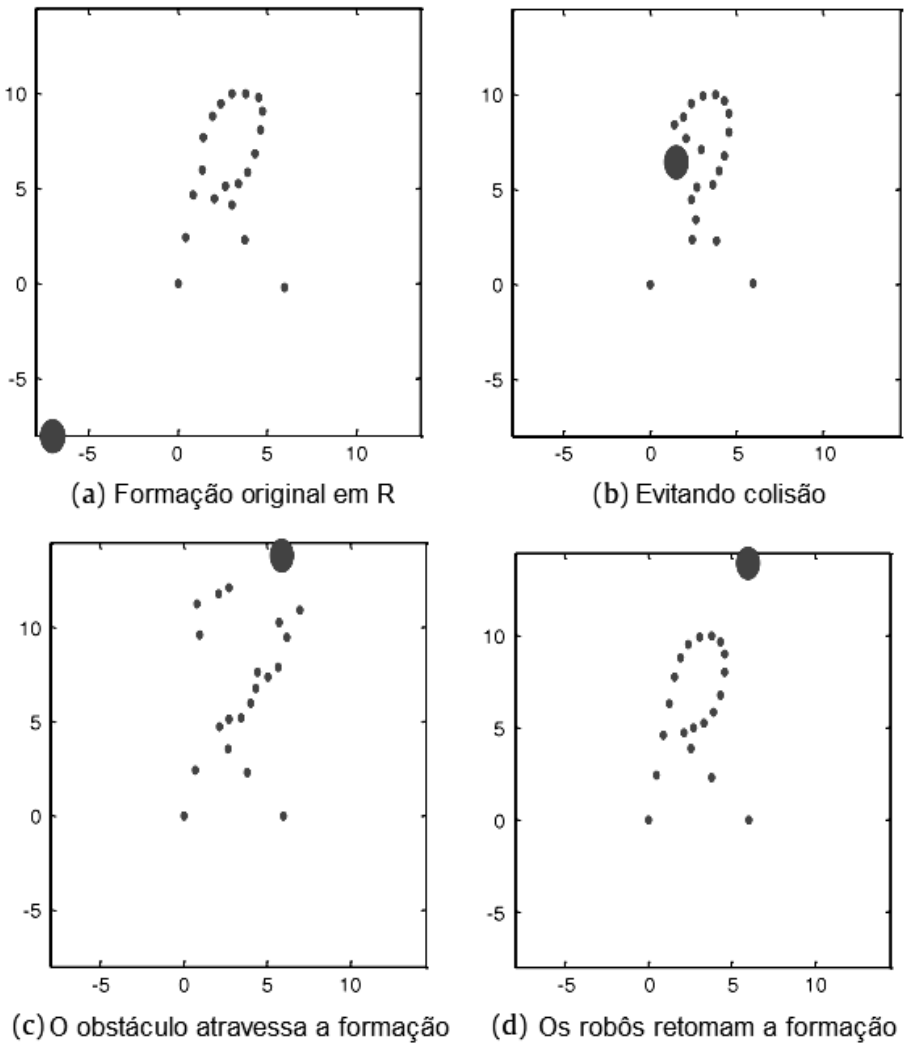

Figura 22 – Exemplo de desvio de obstáculo (MENG; GUO; JIN, 2013).

Em (LIN; HSIAO; CHEN, 2010) os autores apresentam um método baseado em AG, funções potenciais e diagramas de Voronoi para planejamento de caminhos livres de obstáculo e controle de formação. A abordagem é dividida em duas tarefas, planejamento de caminho e planejamento de movimento, na primeira tarefa é obtido um caminho (utilizando diagramas de Voronoi) no qual o centro da formação deve seguir mantendo a formação triangular, o planejamento de movimento é feito por um AG baseado em funções potenciais de atração (meta) e repulsão (obstáculos). O caminho gerado é divido em vários pontos e em cada ponto o AG encontra a melhor configuração para o time de robôs, ou seja, o formato triangular pode se adequar ao ambiente. A Figura 23 mostra o resultado da aplicação do método em simulação, onde a formação é mantida do início ao fim. Como podemos observar, o AG gerou diferentes configurações para a formação no decorrer do caminho, reconfigurando as posições dos robôs para manter o aspecto triangular.

O cromossomo que representa o caminho final é definido pelas coordenadas de to-

dos os robôs. O método foi testado somente em simulação com ambientes previamente conhecidos, colisões com obstáculos estáticos e com outros robôs são tratadas.

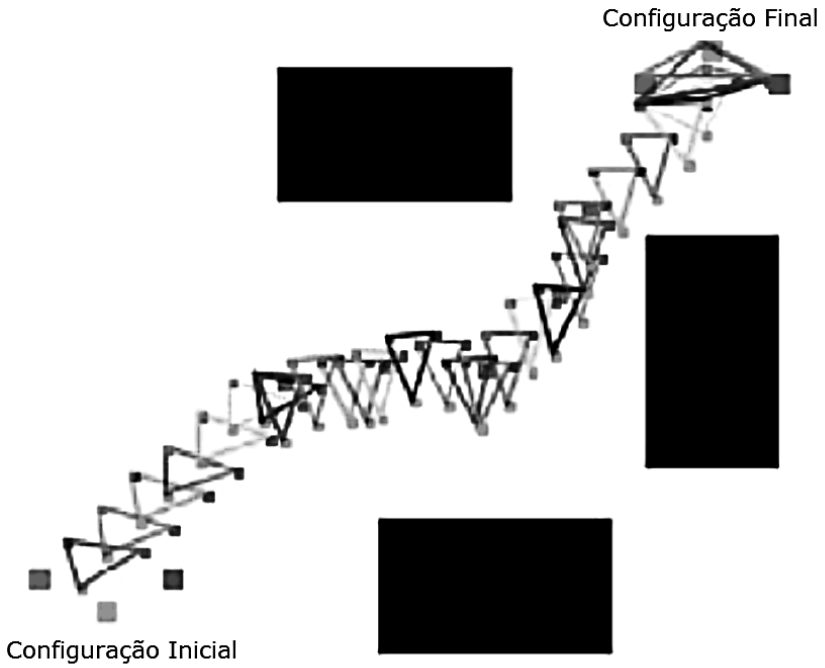

Figura 23 – Exemplo de aplicação do método em simulação (LIN; HSIAO; CHEN, 2010).

Em (ASL; MENHAJ; SAJEDIN, 2014) os autores propõem um método para planejamento de caminho e controle de formação baseado em otimização por reprodução assexuada e funções potenciais. Existe um robô líder que é tratado como um alvo que se movimenta e os robôs seguidores devem seguir usando funções potenciais. O líder e os seguidores devem manter certo ângulo em relação à meta para que a formação seja mantida. A abordagem busca otimizar a trajetória e posição de cada robô na formação. A Figura 24 apresenta a trajetória do time de robôs em um ambiente com obstáculos, na qual a formação acaba sendo desfeita momentaneamente.

Na técnica de otimização por reprodução assexuada existe somente um indivíduo que se auto reproduz: um novo indivíduo é gerado pela mistura do pai com uma versão de si mesmo que sofre mutação em determinadas partes. O pai original e o filho gerado competem (baseado em uma função de aptidão) e o melhor indivíduo sobrevive. O método é capaz de tratar obstáculos estáticos e dinâmicos (outros robôs) além de ser aplicável a situações em tempo real.

A abordagem líder-seguidores é utilizada também em (CAPI; MOHAMED, 2012) com o objetivo de fazer com que o time de robôs e-puck seja capaz de entrar e manter determinada formação triangular fixa. Um algoritmo evolucionário multiobjetivo (NSGA-II) evolui controladores baseados em redes neurais artificiais para a tarefa de controlar múltiplos robôs em formação. A função de aptidão busca minimizar a diferença entre a posição e ângulo atual do robô com os valores definidos em relação ao líder na formação

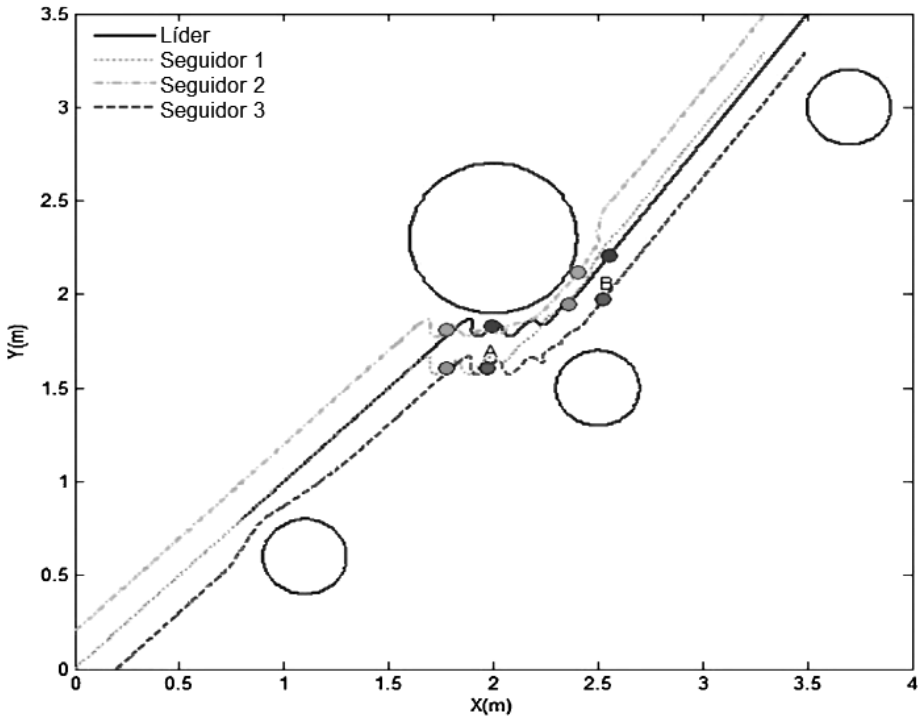

Figura 24 – Exemplo de trajetória em um ambiente com obstáculos (ASL; MENHAJ; SAJEDIN, 2014).

desejada. Os robôs iniciam em posições aleatórias no ambiente, devem entrar em formação e depois manter a formação em relação ao líder até que atinjam a meta. A Figura 25 mostra a organização do sistema, na qual a distância e ângulo em relação ao líder são determinados utilizando uma câmera instalada no robô líder, sendo que os robôs se comunicam utilizando Bluetooth e um computador central. O método foi testado em simulação e em sistemas reais, porém não leva em consideração o desvio de obstáculos.

O problema de controle de formação, especificamente o de retomar a formação após desviar de obstáculos, é tratado em (KOBAYASHI; TOMITA; KOJIMA, 2003). O método se divide em dois modos de operação, o primeiro é um AG responsável pelo controle da formação e o segundo utiliza a técnica de aprendizagem por reforço Q-Learning para desvio de obstáculos. O ambiente é desconhecido e os robôs utilizam apenas as informações de seus sensores para se movimentar no ambiente. No modo AG o método evolui um indivíduo que minimiza o erro entre a posição atual e a posição desejável dos robôs na formação. No modo Q-Learning eles aprendem de acordo com as informações do ambiente e ações predeterminadas, apresentadas na Figura 26. Os movimentos preestabelecidos pelo método foram: A - ir em frente, B - ir para trás, C - virar para esquerda, D - virar à direita, E - rotacionar para esquerda, F - rotacionar para direita. O método foi testado somente em simulação, analisando a capacidade dos robôs de manter a formação inicial baseando apenas nas informações de seus sensores, posicionamento dos robôs vizinhos e processamento local. Somente tipo de formação fixa foi testada, sendo que a abordagem

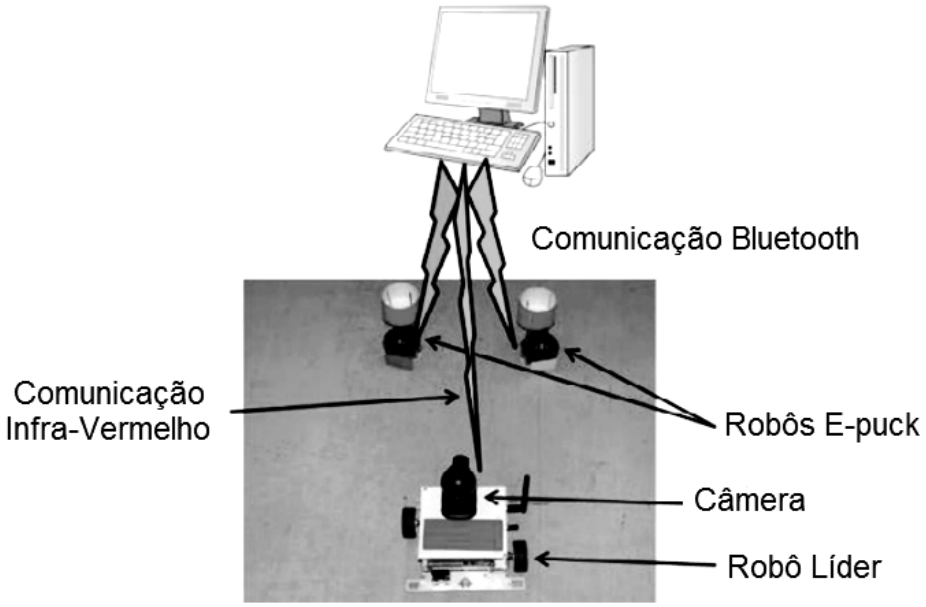

Figura 25 – Organização do sistema quanto a comunicação (CAPI; MOHAMED, 2012).

consegue tratar obstáculos estáticos e dinâmicos.

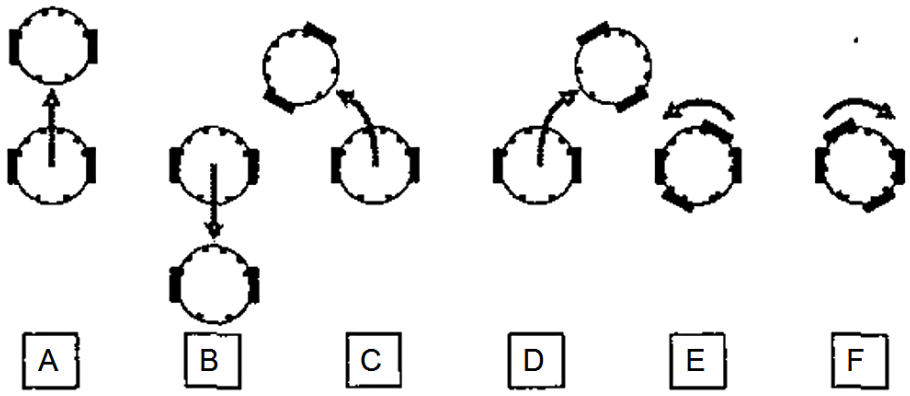

Figura 26 – Conjunto de movimentos preestabelecidos (KOBAYASHI; TOMITA; KOJIMA, 2003).

No trabalho (FU; LI; MA, 2006), os autores buscaram realizar tarefas cooperativas entre múltiplos robôs, se adaptando a obstáculos e mantendo a formação. Foi utilizado o conceito de líder-seguidor, onde um robô é apontado como líder (o que está mais próximo da meta) e os outros o seguem sincronizando suas ações e mantendo a formação (ângulo e velocidade em relação ao líder). Obstáculos estáticos e dinâmicos são tratados, sendo que a ação de desvio é escolhida conforme o obstáculo. Foram definidos quatro comportamentos principais dos robôs: *(i)* mover para a meta, *(ii)* evitar obstáculo estático, *(iii)* evitar obstáculo dinâmico e *(iv)* evitar robô.

Quando um robô detecta um obstáculo ele pode solicitar a troca de líder, sendo o líder

atual quem decide. Após a mudança os outros robôs começam a cooperar com o novo líder em sua trajetória para a meta, mantendo a formação. As ações são aplicadas a partir das informações coletadas pelos sensores em tempo real. Utiliza um controlador baseado em lógica fuzzy para alterar os ângulos de movimento e velocidade dos robôs, inserindo um conceito de intensidade de inclinação para lidar com os vários comportamentos. O método foi avaliado somente em simulação.

O objetivo do trabalho (FREDSLUND; MATARIC, 2002) é que os robôs estabeleçam e mantenham determinada formação, utilizando um robô "amigo" como referência em determinado ângulo de visão. Os robôs trabalham com o mínimo uso de sensores e comunicação, sendo que cada um não tem conhecimento global sobre posicionamento ou direção dos outros robôs, o que gera menor confiança na formação, mas maior simplicidade e robusteza. Cada robô repassa seu ID regularmente. Um robô "condutor" (que não segue nenhum outro robô "amigo") decide sua direção e repassa a formação a ser realizada, sendo que cada robô tem uma lista de formações possíveis a priori.

Os robôs utilizam um sensor giratório que mantém o robô "amigo"sempre no centro, e também as informações locais e comunicação mínima. A simulação foi feita utilizando um servidor que conecta os robôs, sensores e controladores através de uma rede. Os experimentos com robôs reais foram feitos com robôs Pioneer2 DX, com câmera giratória e sensor de distância.

O método proposto em (GE; FUA, 2005) tem como objetivo fazer com que o time de robôs seja capaz de seguir uma meta que se move, desviando de obstáculos e mantendo a formação. O método assume três premissas: *(i)* cada robô é capaz de repassar informações para outros, *(ii)* os robôs têm uma meta em comum, na qual a posição, velocidade e orientação são conhecidos, *(iii)* um robô é capaz de determinar sua própria posição.

Geralmente as formações realizadas por robôs têm formas geométricas e podem ser subdivididas em vários segmentos de linhas. Aproveitando-se desse fato os autores consideraram cada um dos segmentos de linha como uma uma pilha. A proporção de todos os N robôs a serem alocados em cada pilha é determinada pelo usuário e pode mudar durante a execução caso seja necessário (o que aumenta a flexibilidade e escalabilidade). A técnica utilizada foi uma modificação do conceito de campos potenciais para trincheiras potenciais, o que atrai os robôs e faz com que eles se movam na direção das pilhas. Os obstáculos também são tratados por uma força de repulsão. O método foi avaliado somente em simulação com obstáculos estáticos.

Em (MASTELLONE; STIPANOVIC; SPONG, 2007) foi apresentado um método com rastreamento estável descentralizado que consegue realizar formações de forma dinâmica e evitar obstáculos (em tempo real) por robôs não holonômicos. Foi utilizado um robô líder, que continha informações de velocidade e posição da meta, e outros robôs seguidores para manter a formação utilizando informações de distância e ângulo em relação ao líder.

Para realizar o deslocamento em formação, o padrão geométrico da formação foi repas-

sado a priori e então foi calculado o centro de massa da formação, definindo uma trajetória desse centro para a meta. Para atingir o objetivo de forma descentralizada, o controle foi realizado localmente em cada robô. O método foi testado somente em simulação e falhou ao tentar determinar a proximidade de um objeto (inviabilizando a tarefa de evitar obstáculos).

Um abordagem líder-seguidores com comunicação feita através de sinais de som foi utilizada em (EDWARDS et al., 2004). O objetivo do trabalho foi manter a formação do time de veículos submarinos enquanto navega por pontos predefinidos pela missão de operações de varredura e descoberta de minas aquáticas.

A comunicação foi feita via sinais de sons emitidos pelo líder, caso ele falhe os seguidores podem reverter para o modo independente de navegação ou um deles se torna o novo líder. Todos os robôs sabem os pontos pelos quais o líder deve passar e ajustam sua trajetória na formação para se adequar à trajetória. O método foi testado em sistemas reais, porém não leva em consideração o desvio de obstáculos.

No trabalho (BALLARD; REN, 2009), os autores implementam uma estratégia descentralizada para mudança dinâmica entre formações. A posição de cada robô na formação é derivada dos dados compartilhados por seus vizinhos, sendo que o número de robôs na formação é descoberto via *consensus* (baseado em vetores de confiança).

Além de adaptar a formação dinamicamente, o método também consegue lidar com diminuição ou aumento no número de robôs da formação. Primeiramente, é utilizado um computador externo para definir a formação e repassar aos robôs (parâmetros que tem como fundamento a forma elíptica), e o restante do processamento é realizado localmente em cada robô. O método foi testado em experimentos com os robôs P3-DX e amigoBots, porém não trata de desvio de obstáculos.

Em (MEHRJERDI; SAAD; GHOMMAM, 2011) o objetivo é controlar um time de robôs que se movimentam em formação e desviam de obstáculos. Foi proposta a utilização de lógica fuzzy para decifrar as informações do ambiente e um algoritmo de controle e coordenação fuzzy para manipular os robôs em diferentes formações.

O modelo fuzzy utiliza três comportamentos básicos: *(i)* seguir caminho, *(ii)* coordenação de grupo e *(iii)* evitar obstáculo. O caminho de cada robô é dividido em vários pontos, nos quais ele deve passar. O controlador fuzzy faz com que todos os robôs cheguem a seus pontos ao mesmo tempo, quando dois robôs estão em rota de colisão o que tiver menor prioridade deve parar e esperar o outro robô sair da área de colisão. O método foi testado em simulação e trata obstáculos estáticos e dinâmicos.

Com o objetivo de controlar um time de robô para entrar em formação a partir de posições aleatórias iniciais, o trabalho (MONTEIRO; BICHO, 2008) utiliza a abordagem líder-seguidores. Cada robô deve encontrar uma posição na formação de forma autônoma. A formação é representada por uma matriz que identifica quem o robô deve seguir, a distância e os ângulos a serem mantidos entre eles.

As posições dos robôs são decididas através de um leilão. Após as posições serem definidas os robôs escolhem um robô que esteja próximo para seguir. Caso haja falha de robôs, o líder pode escolher entre manter a formação, alternar para outra formação ou abortar a missão. O método foi testado somente em simulação e trata de obstáculos estáticos e dinâmicos.

3.2 Métodos que Empregam Modelos Baseados em ACs

Nesta seção, apresentamos abordagens utilizadas na tarefa de planejamento de caminhos com controle de formação em times de robôs, destacando as diferentes técnicas para cálculo da rota e manutenção das posições dos robôs enquanto navegam pelo ambiente.

Na abordagem utilizada em (MEAD; LONG; WEINBERG, 2009) os robôs são células em um AC, não como a abordagem trivial, na qual o espaço em que o robô existe é topologicamente segmentado em uma grade de células, mas cada agente (robô) representando um ponto da vizinhança. Essas vizinhanças são dinâmicas e baseadas em leilões. O objetivo do trabalho é realizar o controle de formação capaz de lidar com eventos inesperados do ambiente, desviar de obstáculos e se adaptar a alterações no número de robôs. Os parâmetros da formação são repassados a todos os robôs. Cada robô deve enviar seu estado e manter o estado atual de seus vizinhos. A relação de vizinhança é construída considerando um robô no centro de um círculo e seus vizinhos sendo os robôs que estão a um raio de distância.

O método é capaz de suportar diversas formações, porém o número de robôs na vizinhança é definido manualmente, sendo que há problemas em transições entre formações e desvios de obstáculos. Para contornar os problemas citados é proposta uma abordagem baseada em leilões e vizinhanças dinâmicas. Para evitar obstáculos, as relações entre vizinhos são enfraquecidas ou desfeitas permitindo que o robô desvie. Por exemplo, na Figura 27(1) um robô quebra e perde comunicação. Na Figura 27(2) começa o leilão para a posição perdida. Na Figura 27(3), uma célula ganha o leilão e desfaz suas antigas relações. Finalmente, na Figura 27(4), a posição é tomada pelo novo robô. O método foi testado somente em simulação e trata apenas de obstáculos estáticos.

No trabalho (DING; HE, 2010) os autores propõem uma abordagem, com o objetivo de controlar um time de robôs que seja capaz de manter formação e desviar de obstáculos, utilizando dois ACs diferentes; em um a célula é um robô e no outro a célula é referente ao ambiente. A formação é definida por uma função matemática, sendo que o robô deve se manter na curva representada por essa função e a uma certa distância de seus vizinhos, como pode ser visto na Figura 28. Foi definido um círculo em volta de cada robô e a distância entre robôs vizinhos é a distância do raio desse círculo.

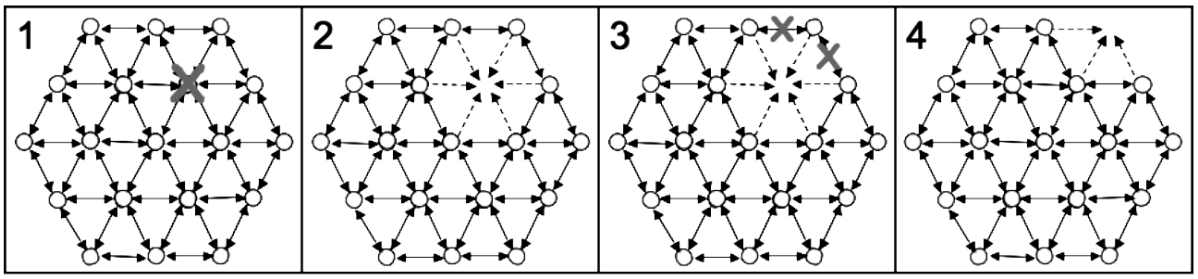

Figura 27 – Exemplo de vizinhança dinâmica (MEAD; LONG; WEINBERG, 2009).

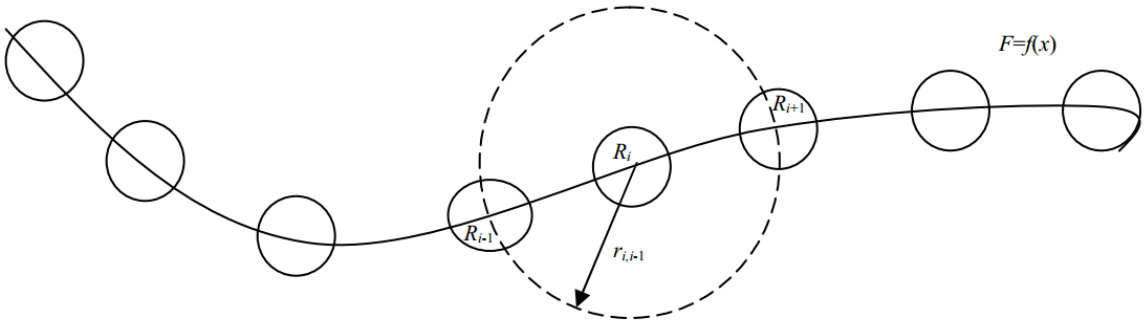

Figura 28 – Exemplo de distância da vizinhança (DING; HE, 2010).

Caso seja detectado um obstáculo entre os robôs, o raio do círculo é aumentado permitindo que haja mais distância entre os robôs para um possível desvio. Os testes foram realizados apenas em simulação e com obstáculos estáticos.

Estratégias baseadas em ACs foram investigadas também em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008), na qual os robôs se movem de forma coordenada e trocam de funções dinamicamente para se adaptar a eventos inesperados do ambiente e manter a formação. O ambiente é primeiramente dividido em uma grade retangular de células idênticas, mostrada na Figura 29.

Uma das vantagens do método é que ele não precisa de informação completa do ambiente, os modelos utilizam apenas as informações locais dos sensores para identificar a vizinhança da célula onde está o robô e aplicam regras de atualização local que decidem qual será o próximo movimento a ser realizado. A abordagem foi testada tanto em simulação quanto em cenários reais com robôs e-puck, tratando de obstáculos estáticos e dinâmicos.

O método proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011b) além de planejar caminhos livres de obstáculos para um time de robôs mantendo formação, também implementa uma tarefa adicional que é o processamento digital de imagens (interpolação), usando cooperativamente as câmeras dos vários robôs. Todos os robôs devem

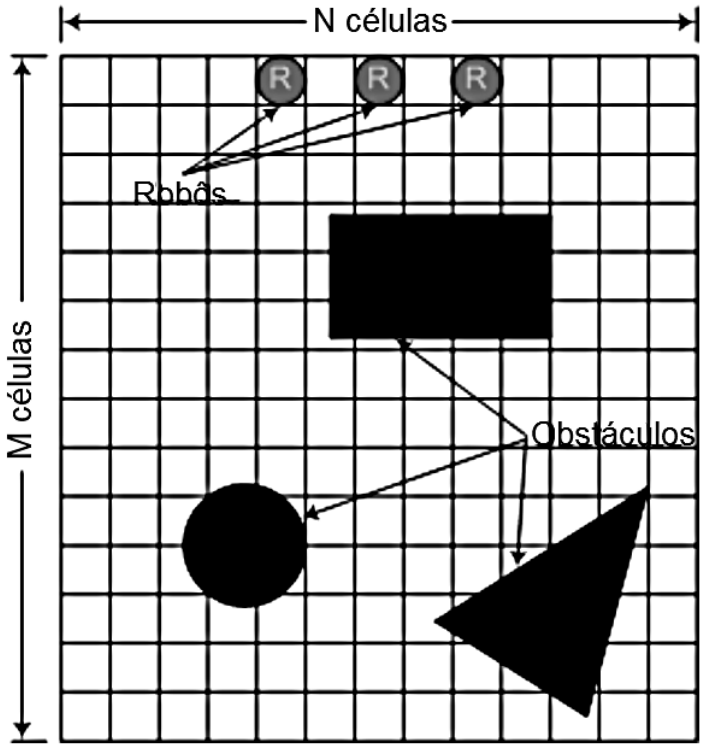

Figura 29 – Discretização do ambiente e do time de robôs (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2008).

percorrer certa distância enquanto mantêm a formação, o desvio de obstáculos e posterior retomada de formação são alcançados utilizando um AC simples mas com regras de transição diferentes para cada tarefa. Por exemplo, a Figura 30(A) representa uma regra para desvio de obstáculo e a Figura 30(B) representa uma regra para controle de formação. As células com estado *"o"* e *"f"* são células com obstáculo e livres, respectivamente, enquanto as células com *"r"*, *"r1"* e *"r2"* são células com robôs

Em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011a) o método para planejamento de caminho para um time de robôs cooperativos mantendo formação e desviando de obstáculos é baseado em ACs e Colônia de Formigas. Os robôs são divididos em subgrupos e cada um deixa um rastro de feromônio (em aplicações reais pode ser utilizado rastros de calor) em seu caminho, permitindo que outros robôs possam seguir (utilizando sensores) esse rastro, diminuindo a necessidade de comunicação entre robôs. A Figura 31 apresenta um exemplo desse comportamento, onde os robôs são identificados pelas células *"R"* e seu respectivo número na formação. O objetivo é manter a formação em linha enquanto se movimenta até o objetivo (células N). As marcas de feromônio vão ficando mais fracas de acordo com o tempo (tons mais claros de cinza).

Os robôs seguem as células com feromônios maiores, sendo que os rastros de feromônio vão se evaporando com o tempo (regras do AC se encarregam de diminuir o valor de feromônio na célula). Caso não haja feromônio no caminho do robô ele pode se comunicar com seus vizinhos a fim de escolher uma ação. As regras do AC são divididas em 3

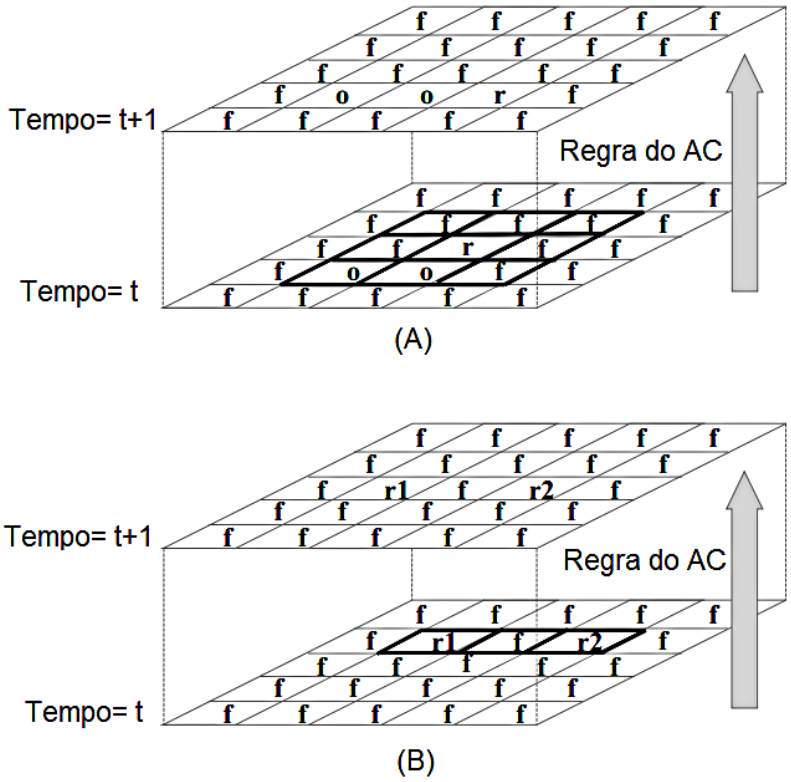

Figura 30 – Exemplo de aplicação de regras de transição (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011b).

operações básicas:

- ❏ Evitar colisão;
- ❏ Deixar rastro de feromônio;
- ❏ Manter formação.

Uma desvantagem do método é o aumento do custo de hardware pois foi necessário aumentar em mais de quatro vezes o número de sensores nos robôs, chegando a 36 sensores (a estrutura original do e-puck contém 8 sensores). O método foi capaz de tratar obstáculos estáticos e dinâmicos, sendo testado somente em simulação.

Outro método que utiliza AC foi proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) com o objetivo de realizar a computação eficiente de caminhos livres de colisão para um grupo de robôs e-puck. Cada robô do time se move sem conhecimento prévio do ambiente, utilizando somente os sensores para detectar e desviar de obstáculos em tempo real. Cada robô deve percorrer determinada distância (em linha reta) e não há controle central do sistema. Utiliza-se a informação de posição e estado dos vizinhos para manter e retomar a formação. Basicamente, as regras do AC se dividem em evitar colisão (utiliza sensores e desvia) e controle de formação (cooperação entre robôs).

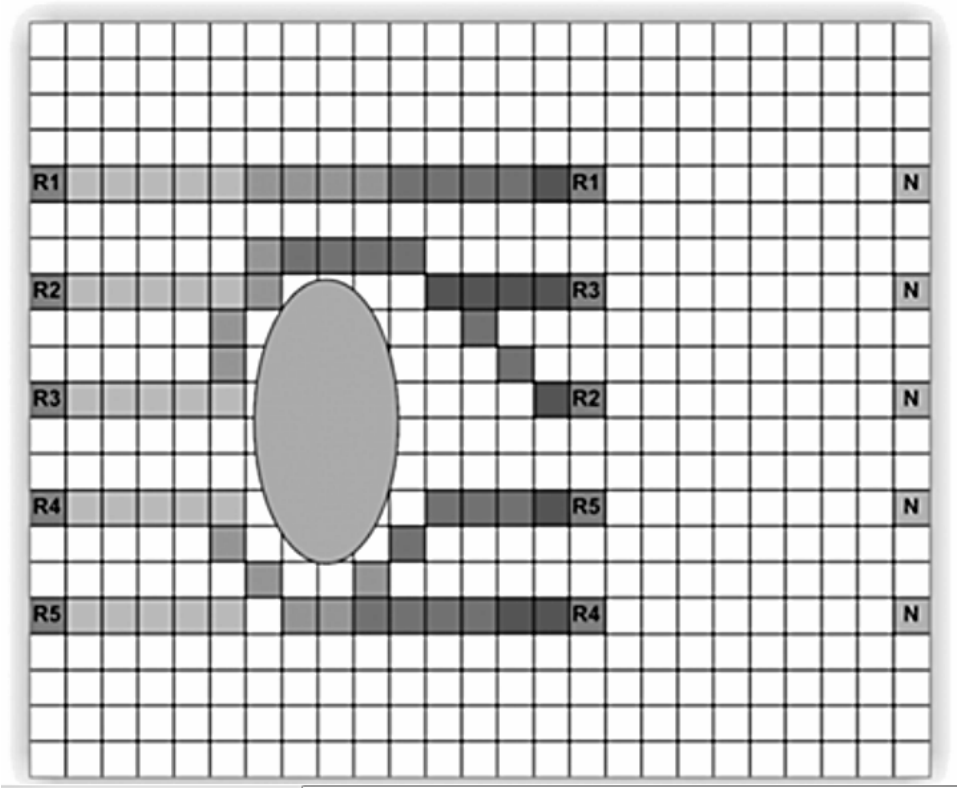

Figura 31 – Exemplo de aplicação de regras para rastro de feromônio (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011a).

O espaço bidimensional no qual os robôs se movimentam foi dividido em uma grade retangular composta por células idênticas, sendo que a definição do tamanho da célula do AC é um fator importante para a precisão do método, pois influencia na detecção e definição de estado de uma célula como obstáculo (os sensores tem distância limitada). Existem basicamente 3 opções de estado no qual a célula pode estar: Robô-i (r_i), Livre (l) ou Obstáculo (o), sendo i o numero do robô. A cada passo de tempo cada robô sabe sua posição no plano cartesiano e sua posição na formação, podendo assim tomar diferentes ações de acordo com seus dados e os obtidos de seus vizinhos. Tais ações se resumem a duas opções: deslocar-se para uma célula vizinha, mantendo sua orientação atual, ou realizar uma rotação sobre seu eixo (+-45°, +-90°), mantendo sua posição atual.

Os robôs que estão em posições mais interiores da formação têm papel de mestre e reúnem informações trocadas (via Bluetooth) com seus vizinhos. Após recolher todos os dados e baseado nas regras do AC o mestre decide a próxima ação a ser feita (troca de posições para retomar formação ou continuar seguindo em direção a meta) e repassa as instruções aos robôs escravos. As duas formações utilizadas no trabalho foram linha e triângulo, a Figura 32 apresenta o comportamento do método para formação em linha em robôs reais.

O método utiliza duas variáveis que ajudam na manutenção da formação, a primeira é referente à distancia a ser percorrida pelos robôs em linha reta e a outra define a

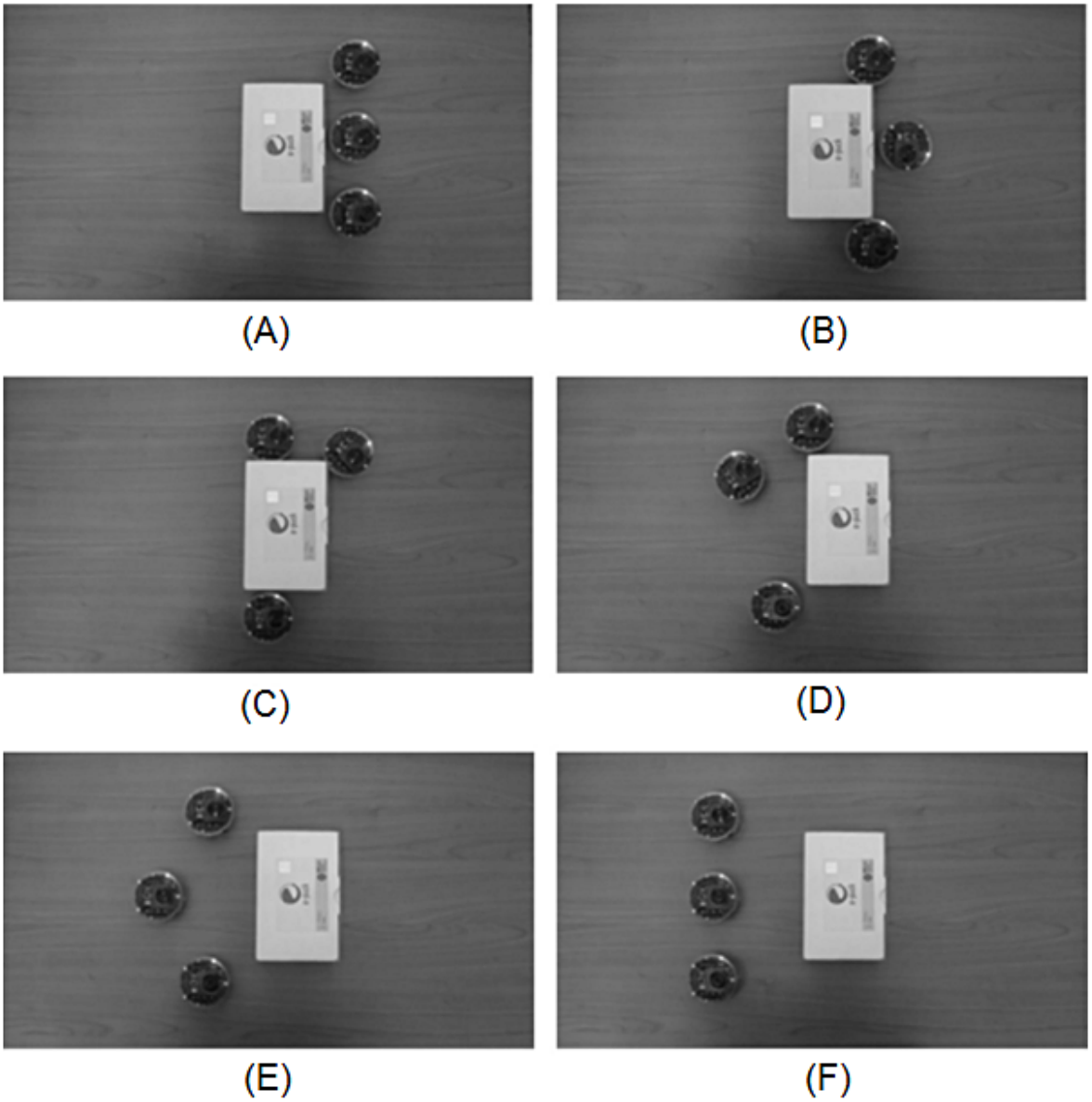

Figura 32 – Exemplo de manobra de desvio de obstáculo e retomada da formação (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c).

distância entre robôs, caso essa distância entre robôs ultrapasse um limite predefinido (no caso de um robô se aproximar de outro devido à um desvio de obstáculo, por exemplo) os robôs aplicam uma manobra de troca de posições, o que, segundo os autores, diminui o tempo necessário para a retomada da formação. Os experimentos foram realizados com obstáculos estáticos e dinâmicos, tanto em simulação quanto em cenários reais.

Em (FERREIRA, 2014), além da análise e categorização das abordagens que utilizam ACs para a tarefa de planejamento de caminhos, foram realizadas contribuições para modelos propostos anteriormente. O foco dessas contribuições foi na modificação do modelo proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), analisando o comportamento do modelo quando aplicado a um único robô e quando aplicado em time.

A principal contribuição do trabalho, quanto ao modelo de AC baseado em regras de transição locais, foi a inclusão dos estados Robô-Rotacionado-i e posteriormente do estado Robô-Alinhado (esses estados não estavam presentes no modelo original de (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c)). No primeiro caso foi identificado que os sensores laterais do robô tinham dificuldade em reconhecer o final de um obstáculo (a quina), fazendo com que o robô entrasse em um estado de rotação contínua (*deadlock*). Desta forma, o robô não conseguia ultrapassar o obstáculo. Um exemplo dessa situação pode ser visto na Figura 33, na qual o robô se depara com o obstáculo à frente, realiza a rotação a -90°e se movimenta à esquerda até que os sensores do robô não mais identifiquem a presença de obstáculo à direita. Nessa situação, de acordo com a regra de desvio de obstáculo, o robô deve rotacionar voltando à orientação original de deslocamento (0°). Porém, ao realizar essa manobra, os sensores da frente do robô voltam a identificar o obstáculo à frente, gerando uma rotação -90°novamente. Desta forma, o robô voltará a repetir essas rotações indefinidamente.

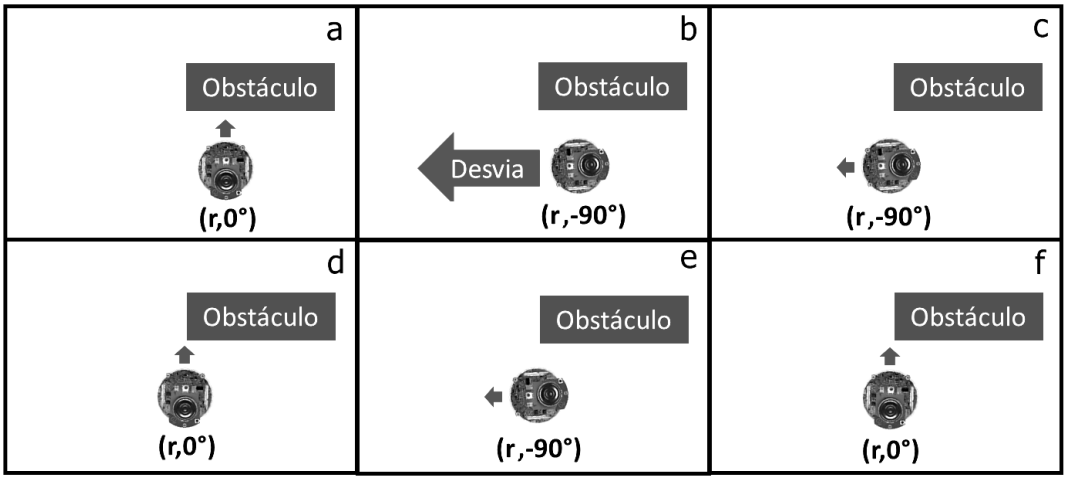

Figura 33 – Exemplo de *deadlock* na movimentação do robô.

Para tratar essa questão foi feita a inclusão de mais um estado possível por célula chamado de "Robô_Rotacionado_i" e a alteração das regras de desvio de obstáculo para trabalharem com esse novo estado, permitindo que o desvio da quina fosse realizado de forma mais eficiente. O novo estado indica ao robô que, caso esteja na situação de *deadlock*, realize i passos se distanciando do obstáculo para que depois aplique as regras de controle de formação. O tamanho utilizado para célula do AC foi de 1 cm e o tamanho do raio do robô e-puck é de 3,5 cm, portanto para que o robô ultrapassasse completamente a quina o valor de i foi definido como 4, ou seja, o robô executa 4 passos no estado Robô_Rotacionado. Entretanto, i é um parâmetro que pode ser ajustado para outros robôs e ambientes.

Na Figura 34 podemos observar um exemplo desse comportamento: (a) o robô detecta o obstáculo, (b) a manobra de desvio de obstáculos começa, (c) o robô detecta o limite

do obstáculo, (d) ele evita a quina executando 4 passos (e), o robô completa o desvio da borda do obstáculo e inicia a rotação de volta para 0°, (f) o robô está pronto para aplicar regras de controle de formação.

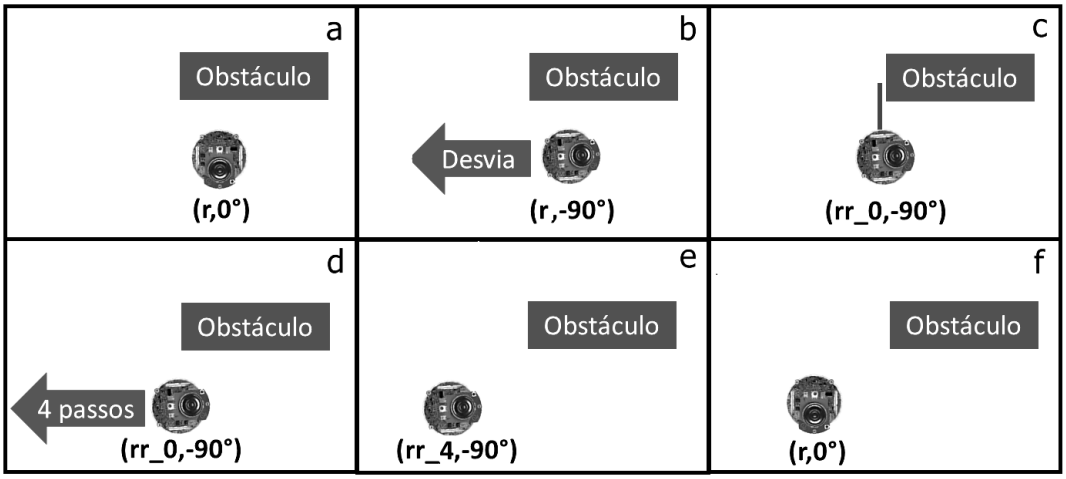

Figura 34 – Exemplo de utilização do estado Robô-Rotacionado-i na movimentação do robô.

Outro estado chamado "Robô_Alinhado"foi incluído para atenuar o comportamento zigue-zague do robô após o desvio de uma quina, com esse novo estado o robô alterna entre a execução de um passo a 0^o no sentido do objetivo e um passo na diagonal retomando sua coluna original. A Figura 35 apresenta um exemplo dessa situação: (a) o robô executa um passo em direção à meta, (b) para retornar ao seu eixo original, o robô entra no estado Robô_Alinhado no qual rotaciona 45°para direta e executa um passo na diagonal, essa sequência de ações é repetida alternadamente de (c) até (f) e continua até que o robô atinja novamente seu eixo de movimentação inicial. O método foi simulado por meio do software Webots e testado em experimentos com robôs e-puck, tratando somente de obstáculos estáticos.

Em relação ao modelo cooperativo foi detectado um problema na estratégia de troca de posições na formação, que aumentava a probabilidade de ocorrência de pareamento de robôs (situação em que dois robôs, que estão tentando trocar de posições, ficam lado a lado por tempo indeterminado) impossibilitando que a meta fosse alcançada. A Figura 36 apresenta a situação de pareamento de robôs após a tentativa de troca de posições: (a)-(c) o robô líder identifica um obstáculo a frente e executa um manobra de desvio para esquerda, (d) os dois robôs da esquerda resolvem trocar de posição na formação, devido a distância entre os dois ter ultrapassado o limite mínimo definido, (e)-(f) os dois robôs executam passos diagonais no sentido de seus novos eixos de movimentação, (g) os robôs acabam se encontrando durante a manobra de troca de posições, (h) os robôs consideram um ao outro como obstáculo e, aplicando as regras de desvio, acabam executando passos em frente indefinidamente.

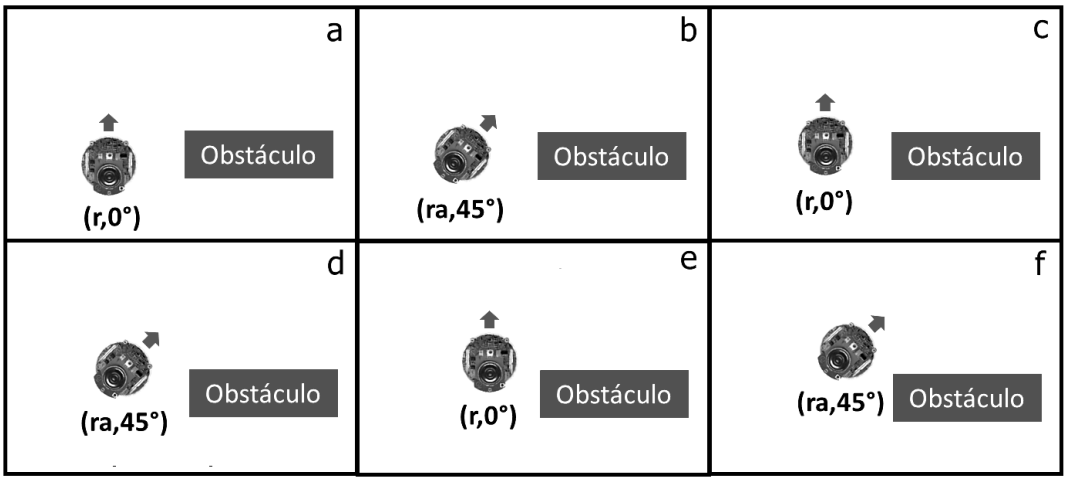

Figura 35 – Exemplo de utilização do estado Robô-Alinhado na movimentação do robô.

Os autores decidiram eliminar a estratégia de troca de posições, fazendo com que cada robô retornasse ao seu próprio eixo original. O novo modelo cooperativo apresentou um comportamento mais eficiente para os cenários testados e resolveu o problema de pareamento de robôs, como pode ser visto na Figura 37, na qual cada robô retorna para seu eixo original: (a)-(c) o robô líder identifica um obstáculo a frente e executa um manobra de desvio para esquerda, (d) o robô finaliza os passos dados para se distanciar da quina e retorna para o ângulo de 0°, (e)-(g) o robô executa passos em direção à meta enquanto detecta o obstáculo em sua lateral, (h) o robô começa a retornar ao seu eixo original alternando entre um passo em frente e um passo diagonal. Esse modelo cooperativo foi testado somente em simulação, por meio do simulador Webots.

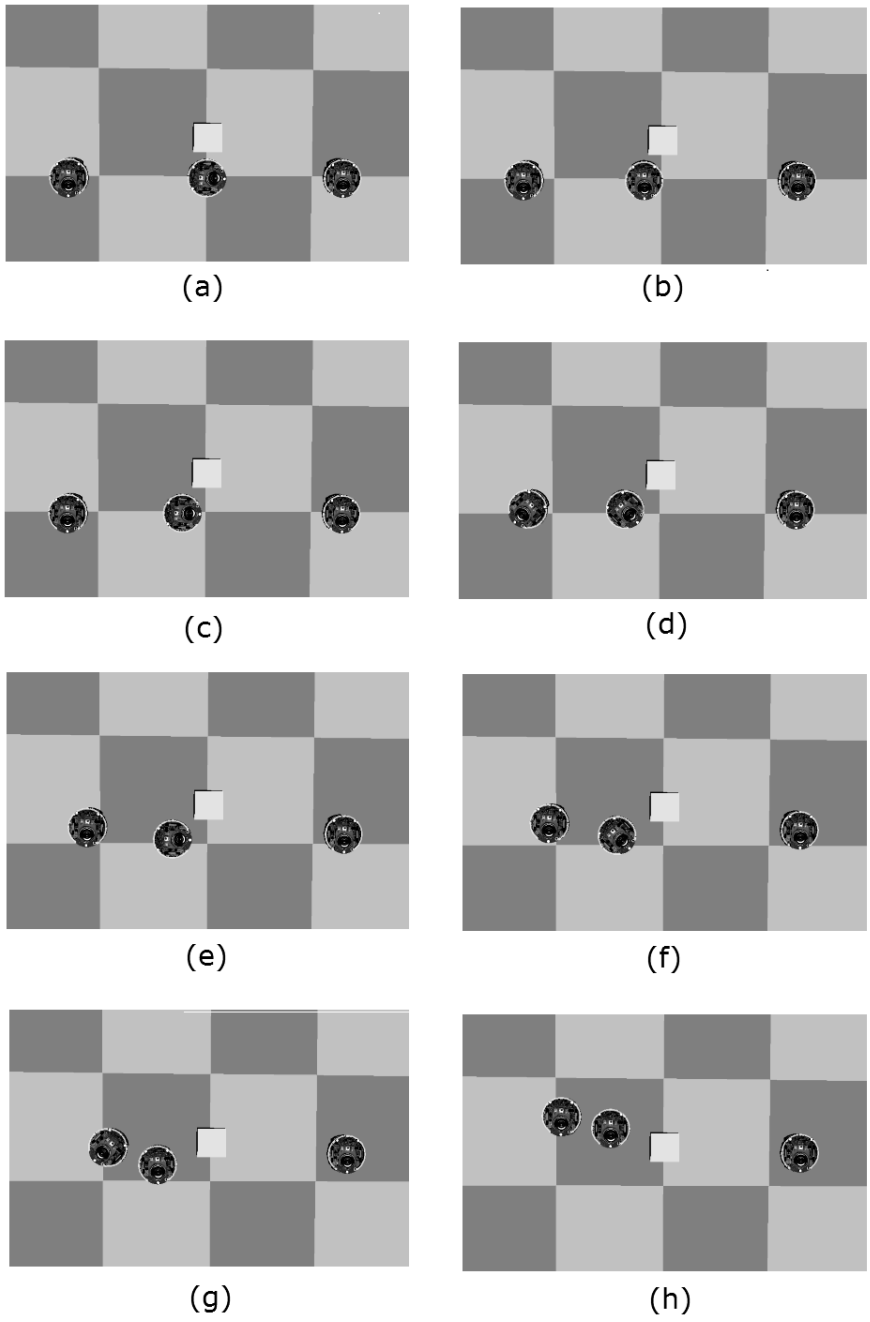

Figura 36 – Exemplo de pareamento entre robôs (FERREIRA, 2014).

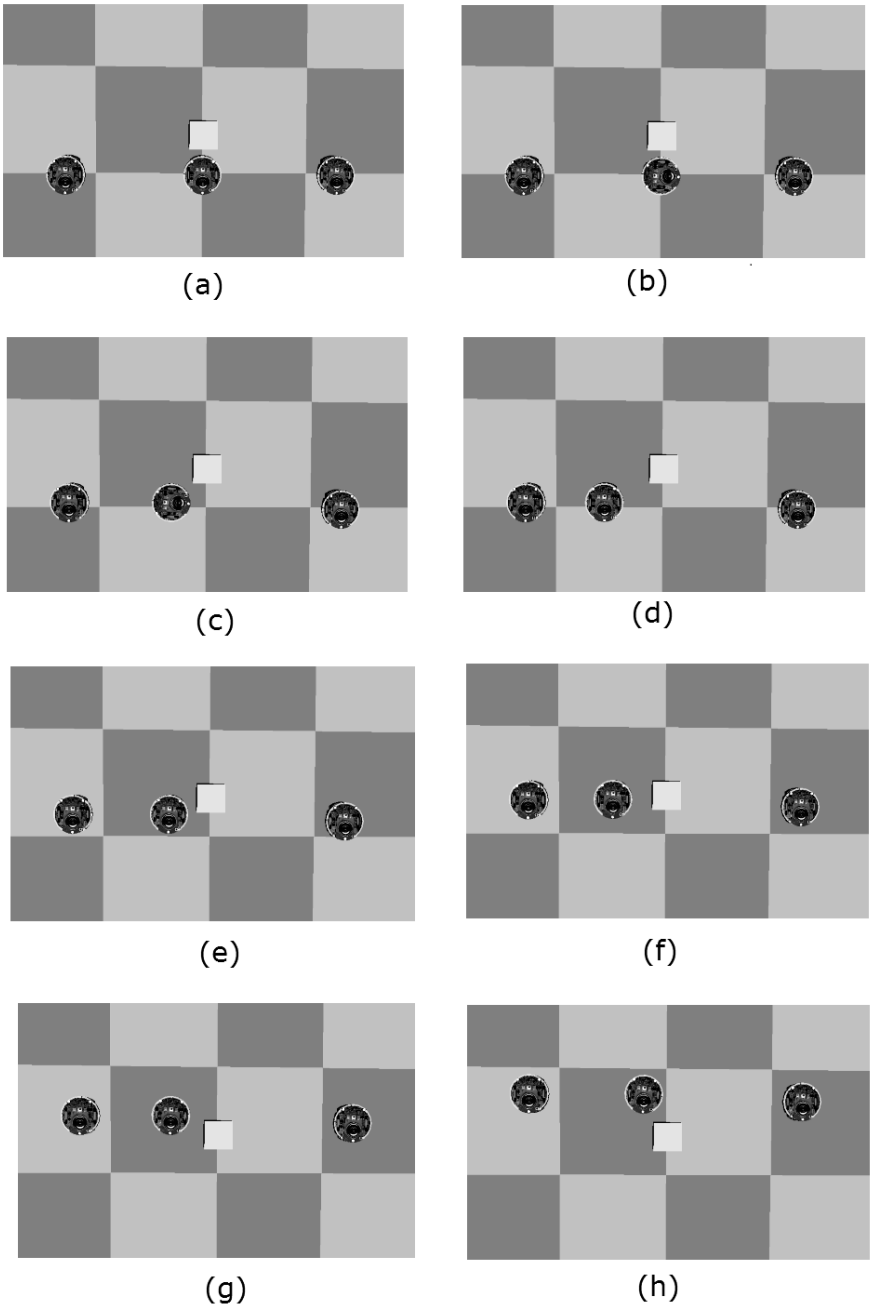

Figura 37 – Exemplo de desvio sem troca de posições (FERREIRA, 2014).

Capítulo 4

Método Evolutivo-Cooperativo

Nossa proposta envolve o desenvolvimento de um método evolutivo-cooperativo para navegação de times de robôs autônomos, utilizando a habilidade de otimização de um AG aliada às regras compactas e processamento discreto e simplificado de um autômato celular. Com a inclusão da técnica de classificação evolutiva da vizinhança e da otimização bioinspirada da reconfiguração da formação, esperamos contribuir para a melhoria do sistema de navegação dos robôs.

No problema de navegação aqui investigado, um time de robôs deve se deslocar por um ambiente, mantendo uma formação entre os agentes. Ou seja, os robôs devem seguir um padrão espacial desejado ao mesmo tempo que se locomovem pelo ambiente, seguindo eixos de deslocamento preferenciais. O padrão mais investigado neste trabalho é a formação em linha reta, mas outros padrões também são possíveis, como o padrão triangular. A partir de um ponto inicial de onde os robôs já iniciam sua movimentação no padrão desejado (definido pelas distancias horizontais e verticais que se deseja manter), o time deve se deslocar de um lado a outro do ambiente procurando manter esse padrão (ou seja, as distancias iniciais). Entretanto, obstáculos eventuais podem surgir durante essa navegação e cada membro do time deve ser capaz de se desviar autonomamente desse obstáculo, porem todo o time deve procurar manter o padrão desejado pelo maior tempo o possível, mesmo quando alguns robôs precisam realizar seus desvios. Dessa forma, o problema de navegação e desvio de obstáculos aqui investigado se diferencia do problema de controle de trajetórias com desvio de obstáculos tradicionalmente abordado por algoritmos como o BUG (HARO; TORRES, 2006) e campos potenciais (HWANG; AHUJA, 1992), nos quais o objetivo é fazer com que o robô se desloque de um ponto inicial a meta sem se preocupar com uma formação especifica. Nesses algoritmos tradicionais, quando o robô encontra um obstáculo ele apenas contorna o mesmo e depois se desloca novamente em direção à meta sem se preocupar em retornar para sua trajetória inicial nem com sua posição espacial em relação a outros robôs.

Nosso modelo teve como ponto de partida a abordagem proposta em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e aperfeiçoada em (FERREIRA; VARGAS; OLI-

VEIRA, 2014) e (OLIVEIRA; VARGAS; FERREIRA, 2015), focando principalmente na qualidade da trajetória gerada, robustez quanto à parametrização automática e estruturação da troca de mensagens entre os robôs. Primeiramente apresentaremos os conceitos básicos dessa abordagem proposta anteriormente e a seguir, apresentaremos as principais modificações efetuadas no modelo discutido em (FERREIRA, 2014). Essas modificações serão apresentadas em duas seções. Na primeira, discutimos as modificações efetuadas com respeito à navegação de um robô único. A segunda seção apresenta as modificações efetuadas com respeito a navegação do time de robôs com controle de formação. Para cada caso, vamos primeiro descrever os pontos fracos identificados nos métodos precursores e depois apresentaremos as melhorias e novas propostas desenvolvidas na presente dissertação. Ao final, o modelo resultante é apresentado.

4.1 Conceitos básicos sobre as abordagens baseadas em ACs

Tanto o modelo proposto em (FERREIRA, 2014), quanto o modelo proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) serviram de base para as investigações realizadas nesse trabalho e para o novo modelo evolutivo-cooperativo aqui proposto.

A definição do ambiente foi a mesma utilizada em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), considerando o espaço de movimentação como um reticulado bidimensional dividido igualmente em células quadradas. A decisão de movimentação do robô é realizada a cada passo de tempo e a regra de transição é escolhida com base na vizinhança do robô, que por sua vez é determinada por meio das leituras dos sensores de distância. Essa estratégia foi conceituada pelo autores como abordagem por **Regra de Atualização Local**.

O robô pode realizar diferentes movimentações no reticulado dependendo de suas células vizinhas. O primeiro tipo de movimentação é o deslocamento entre células, ou seja, o robô se locomove uma determinada distância até que ocupe outra célula do reticulado. Para isso, a distância percorrida deve equivaler ao tamanho de uma célula (c), ou em caso de movimentação diagonal a distancia será igual a raiz do valor correspondente ao tamanho de duas células ($\sqrt[2]{2c}$). O outro tipo de movimentação é simplesmente uma rotação em seu próprio eixo, ou seja, o robô fica na mesma célula porém realiza um giro em um dos 4 possíveis ângulos: -90°, -45°, 45°ou 90°.

A vizinhança utilizada foi a de Moore e o conjunto de estados que cada célula pode obter é composto por: Livre (l), Obstáculo (o), Robô (r), Robô-Rotacionado-i (rr_i) e Robô-Alinhado (ra). Os estados Livre e Obstáculo identificam a existência ou não de algum objeto a ser desviado, essa identificação é feita a partir da comparação dos valores advindos da leitura dos sensores de infravermelho com valores pré-definidos, caso a leitura

seja maior então a célula é marcada como obstáculo. Os estados Robô, Robô-Rotacionado-i e Robô-Alinhado são referentes ao estado da célula no qual o robô se encontra.

Para que cada célula tenha seu estado atualizado a cada passo de tempo é necessária a aplicação das regras de transição. O grupo de regras foi dividido em **regras para desvio de obstáculos** e **regras para controle da formação**. Quando um obstáculo é detectado os robôs agem de forma reativa e autônoma, aplicando as regras para evitar a colisão. Quando não há obstáculos na vizinhança o robô então age com o objetivo de manter sua formação original, ou seja, ele realizará ações por meio das regras de controle de formação.

O conjunto de regras leva em consideração quatro passos para o desvio da quina utilizando o estado Robô-Rotacionado-i e a alternância de movimentos diagonais com o estado Robô-Alinhado, conforme apresentado na Tabela 1. Quanto as regras que são acionadas no modo de controle de formação, os autores utilizaram os mesmos parâmetros do trabalho de (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), no qual a diferença entre a coluna original (hor) e a coluna atual (x) é considerada na hora da escolha da regra de transição (veja a Tabela 2).

4.2 Inovações aplicadas ao modelo utilizando apenas 1 robô

Nessa seção, descreveremos nossas propostas voltadas para o comportamento individual de cada robô, ou seja, todas as modificações realizadas no modelo do AC para as decisões tomadas de forma local, sem considerar a característica cooperativa do modelo para o time.

4.2.1 Limitações dos modelos anteriores

Em (FERREIRA, 2014), foi levantada uma importante limitação do modelo discutido em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c): a impossibilidade de contornar a quina do obstáculo em diversas situações. Para corrigir esse problema, os autores adicionaram 5 novos estados ao AC.

Analisamos o método proposto em (FERREIRA, 2014) quanto ao comportamento de cada robô em vários cenários e verificamos que a movimentação era realizada de forma não adequada em duas principais situações. A primeira situação ocorre durante a manobra de desvio de obstáculo: segundo as regras locais de transição, após o robô detectar um objeto à sua frente, ele deve rotacionar 90°ou -90°(dependendo dos valores obtidos por seus sensores) e se movimentar buscando encontrar o limite lateral do obstáculo. Quando a quina é detectada o estado central da vizinhança se altera para Robô_Rotacionado indicando a aplicação de 4 passos distanciando-se do obstáculo. Após realizar esses passos o robô

Tabela 1 – Conjunto de Regras para Desvio de Obstáculos

Estado	Ângulo	Vizinhança	Novo Estado	Novo Ângulo
Livre	0°	(a5 = Robô ou Robô_Rotacionado) & a3 & a7 = Livre	Robô	0°
		Caso Contrário	Livre	
	-90°	a3 = Robô & a5 = Livre & a2 = Obstáculo	Robô	-90°
		a3 = Robô_Rotacionado_0	Robô_Rotacionado_1	
		a3 = Robô_Rotacionado_1	Robô_Rotacionado_2	
		a3 = Robô_Rotacionado_2	Robô_Rotacionado_3	
		a3 = Robô_Rotacionado_3	Robô	
		Caso Contrário	Livre	
	90°	a7 = Robô & a5 = Livre & a8 = Obstáculo	Robô	90°
		a3 = Robô_Rotacionado_0	Robô_Rotacionado_1	
		a3 = Robô_Rotacionado_1	Robô_Rotacionado_2	
		a3 = Robô_Rotacionado_2	Robô_Rotacionado_3	
		a3 = Robô_Rotacionado_3	Robô	
		Caso Contrário	Livre	
Obstáculo	Qualquer	Qualquer	Obstáculo	Qualquer
Robô	0°	a1 & a2 & a8 = Livre	Livre	0°
		a1 = Obstáculo & a2 & a3 = Livre	Robô	90°
		a1 & (a2 OR a3) = Obstáculo & a7 & a8 = Livre	Robô	-90°
		a1 & a7 = Obstáculo & a3 = Livre	Robô	90°
		a1 & a3 = Obstáculo & a7 = Livre	Robô	-90°
		a8 = Obstáculo & a3 = Livre	Robô	90°
		a2 = Obstáculo & a7 = Livre	Robô	-90°
	-90°	a1 = Obstáculo & a7 & a6 = Livre	Livre	-90°
		a1 = Livre	Robô_Rotacionado_0	0°
	90°	a1 = Obstáculo & a3 & a4 = Livre	Livre	90°
		a1 = Livre	Robô_Rotacionado_0	0°
Robô_Rotacionado_i	0°	a1 & a2 & a8 = Livre	Livre	-0°
		a1 = Obstáculo & a2 & a3 = Livre	Robô_Rotacionado_i	-90°
		a1 & (a2 ou a3) = Obstáculo & a7 & a8 = Livre	Robô_Rotacionado_i	90°
		a1 & a7 = Obstáculo & a3 = Livre	Robô_Rotacionado_i	-90°
		a1 & a3 = Obstáculo & a7 = Livre	Robô_Rotacionado_i	90°
		a8 = Obstáculo & a3 = Livre	Robô_Rotacionado_i	-90°
		a2 = Obstáculo & a7 = Livre	Robô_Rotacionado_i	90°
	-90°	a7 = Livre	Livre	-90°
		Caso Contrário (Não Existe Caminho)	Robô_Rotacionado_i	-90°
	90°	a3 = Livre	Livre	90°
		Caso Contrário (Não Existe Caminho)	Robô_Rotacionado_i	90°

rotaciona novamente, retornando para o ângulo 0°(sentido da meta), e analisa o ambiente por meio de seus sensores. Caso ele identifique que não há mais obstáculos ao seu redor, a tentativa de retornar à formação é iniciada, realizando movimentos com o objetivo de voltar ao seu eixo de deslocamento original. Neste momento, o contorno da quina ainda

Tabela 2 – Conjunto de Regras para Controle de Formação

	Células no tempo t										Estado de a0 no tempo t+1	
Caso	**a0**	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**Ângulo**	**a1**	**Novo Ângulo**
hor - x = 0	Robô	Livre	Livre	Livre	Livre	Livre	Livre	Livre	Livre	0°	Livre	0°
hor - x = 0	Livre	Livre	Livre	Livre	Livre	Robô	Livre	Livre	Livre	0°	Robô	0°
hor - x >0	Robô	Livre	Livre	Livre	Livre	Livre	Livre	Livre	Livre	0°	Livre	0°
hor - x >0	Livre	Livre	Livre	Livre	Livre	Robô	Livre	Livre	Livre	0°	Robô-Alinhado	0°
hor - x <0	Robô	Livre	Livre	Livre	Livre	Livre	Livre	Livre	Livre	0°	Livre	0°
hor - x <0	Livre	Livre	Livre	Livre	Livre	Robô	Livre	Livre	Livre	0°	Robô-Alinhado	0°
hor - x >0	Robô-Alinhado	Livre	Livre	Livre	Livre	Livre	Livre	Livre	Livre	0°	Livre	0°
hor - x >0	Livre	Livre	Livre	Livre	Livre	Livre	Robô-Alinhado	Livre	Livre	0°	Robô	0°
hor - x <0	Robô-Alinhado	Livre	Livre	Livre	Livre	Livre	Livre	Livre	Livre	0°	Livre	0°
hor - x <0	Livre	Livre	Livre	Livre	Livre	Livre	Robô-Alinhado	Livre	Livre	0°	Robô	0°

não foi realizado totalmente, ou seja, o robô apenas se distanciou do obstáculo mas ainda não o ultrapassou em profundidade, o que acaba gerando um deslocamento em ziguezague. Esse comportamento se caracteriza por tentativas frustradas do robô de retornar ao seu eixo original, pois a todo momento ele encontra novamente o obstáculo em seu caminho e acaba realizando manobras de distanciamento inúmeras vezes. Os resultados dos experimentos que serão discutidos na seção 5.1 ilustram esse comportamento indesejado.

A segunda situação de ineficiência, que identificamos em (FERREIRA, 2014), ocorre quando o robô está se movimentando para frente (no sentido da meta), mas ainda não está se deslocando sobre o seu eixo de deslocamento. Como parte da estratégia para retomada do eixo, o robô sempre retorna para 0°, depois de dar um passo na diagonal, realizando rotações a todo momento. Além de aumentar o tempo necessário para que o robô volte ao seu eixo de deslocamento original, esse comportamento pode aumentar o erro de odometria, devido às constantes rotações.

4.2.2 Novos estados e regras para o modelo baseado em AC

Identificamos que o principal problema, na primeira situação de ineficiência do modelo proposto em (FERREIRA, 2014), é que, após contornar a face frontal do obstáculo, atingindo e superando sua quina, o modelo não definia um movimento do robô em direção à meta, para que a quina do obstáculo fosse superada também em profundidade. Assim, percebemos que, antes de realizar a manobra para retomada de formação, com o objetivo de retornar ao seu eixo de deslocamento original, o robô deveria efetuar alguns passos "para a frente", o suficiente para que a parte traseira do robô também superasse a quina do obstáculo. Deste modo, o modelo do AC foi alterado de forma a proporcionar essa movimentação. Quatro novos estados chamados Robô_Formação_i (rf_i) foram adicionados ao modelo proposto em (FERREIRA, 2014). O principal objetivo dos novos estados é indicar ao robô que, após se distanciar do obstáculo na manobra de desvio, ele deve primeiramente executar i passos em direção à meta. Somente após esses passos ele

deve acionar as regras de controle de formação e iniciar os movimentos para a retomada de seu eixo original de deslocamento. Um exemplo dessa situação é dado na Figura 38, como podemos ver em (a) o robô detecta o obstáculo, em (b) a manobra de desvio inicia, em (c) a quina do obstáculo é detectada, em (d) o robô executa 4 passos se distanciando do obstáculo, em (e) para contornar a quina o robô executa 4 passos em 0°(novo comportamento), e em (f) o robô finaliza a manobra de desvio da quina, posicionando-se ao lado do obstáculo e com sua direção voltada novamente para a meta (0°). A partir desse ponto, o robô pode começar a retornar ao seu eixo de deslocamento original, utilizando as regras de controle de formação, que fazem o robô se deslocar na diagonal. A quantidade i de passos é um parâmetro do modelo e deve ser definido de acordo com o tamanho da célula do AC e do tamanho do robô. Como, em nosso caso, utilizamos células medindo 1 cm e robôs e-puck cujo raio mede 3,5cm, aplicamos $i = 4$. Assim, essa quantidade de passos é suficiente para que o robô avance na profundidade do obstáculo, de forma que a parte traseira do e-puck fique acima da borda. Conforme iremos observar nos resultados do experimento com o novo modelo na seção 5.1, com a adição dos novos estados e de suas respectivas regras de transição, o robô consegue fazer o contorno completo da quina, mantendo-se a uma distância segura da borda e diminui o zigue-zague para realizar tal movimento.

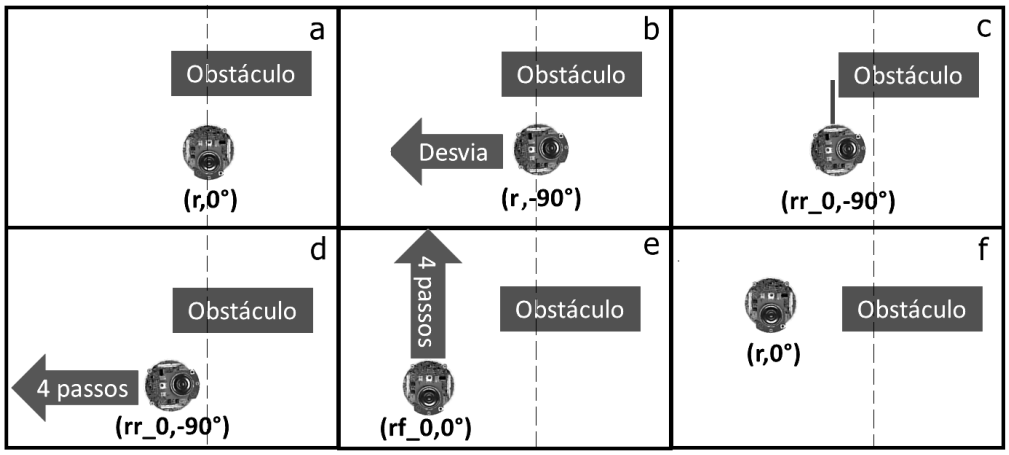

Figura 38 – Exemplo de utilização dos estados Robô_Formação_i.

Para resolver a segunda situação de ineficiência que identificamos em (FERREIRA, 2014), introduzimos mais 1 estado ao modelo, nomeado Robô_Formação_Diagonal (rfd), que indica ao robô que ele deve se movimentar somente em um ângulo de 45°(ou -45°) enquanto estiver tentando voltar para seu eixo original e não encontrar obstáculos à sua frente. Na Figura 39, podemos ver um exemplo de aplicação do estado Robô_Formação_Diagonal. Na Figura 39a, o robô se encontra deslocado do seu eixo original de deslocamento, após locomover-se à esquerda para desviar do obstáculo. Após movimentos similares aos da 38, o robô inicia seu movimento de retorno ao eixo original de deslocamento. De (a) até

(e), o robô aplicou a regra para controle de formação, se movimentando constantemente na diagonal. Em (f), o robô retorna ao seu eixo e ângulo originais.

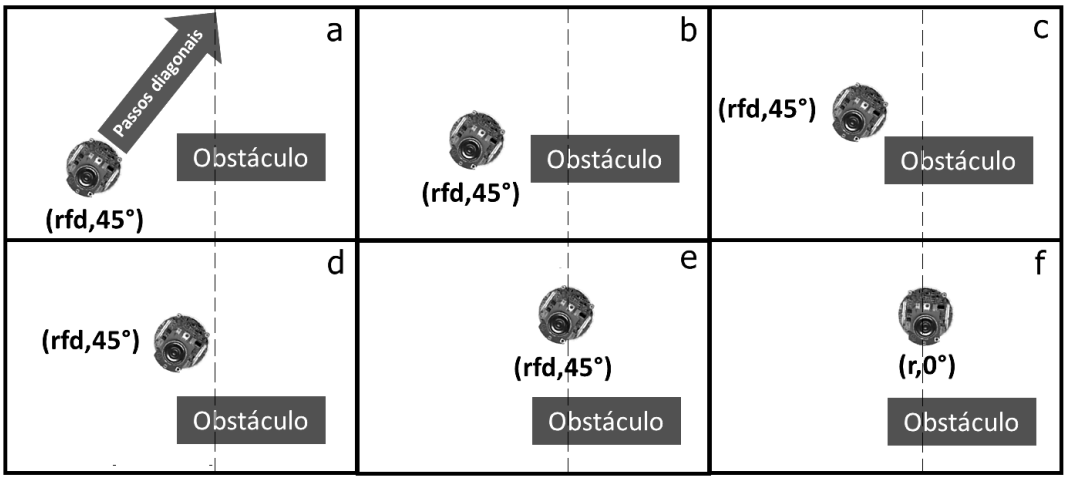

Figura 39 – Exemplo de utilização do estado Robô_Formação_Diagonal

Com a criação e inclusão dos cinco novos estados ao modelo do AC e a movimentação contínua do robô em 45°ou -45°, foram necessárias também modificações na definição da vizinhança e nas regras de transição. No trabalho de (FERREIRA, 2014) os estados das células eram atualizados somente em situações em que o robô estava em um ângulo de -90°, 0°ou 90°, ou seja, oeste (direcionado à célula *a7*), leste (direcionado à célula *a3*) e norte (direcionado à célula *a1*), como mostra a Figura 40.

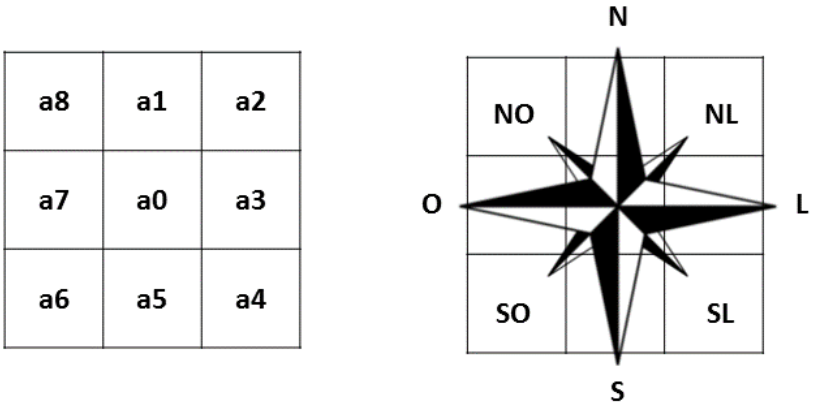

Figura 40 – Posicionamento dás células do reticulado quanto ao posicionamento cardinal.

Tendo em vista que a inserção dos novos estados permite que o robô se movimente constantemente na diagonal, incluímos dois novos cenários para definição da vizinhança (-45°e 45°), apresentados na Figura 41.

Além dos novos cenários para a atualização da vizinhança, novas regras de transição para desvio de obstáculo e controle de formação foram desenvolvidas, otimizando o

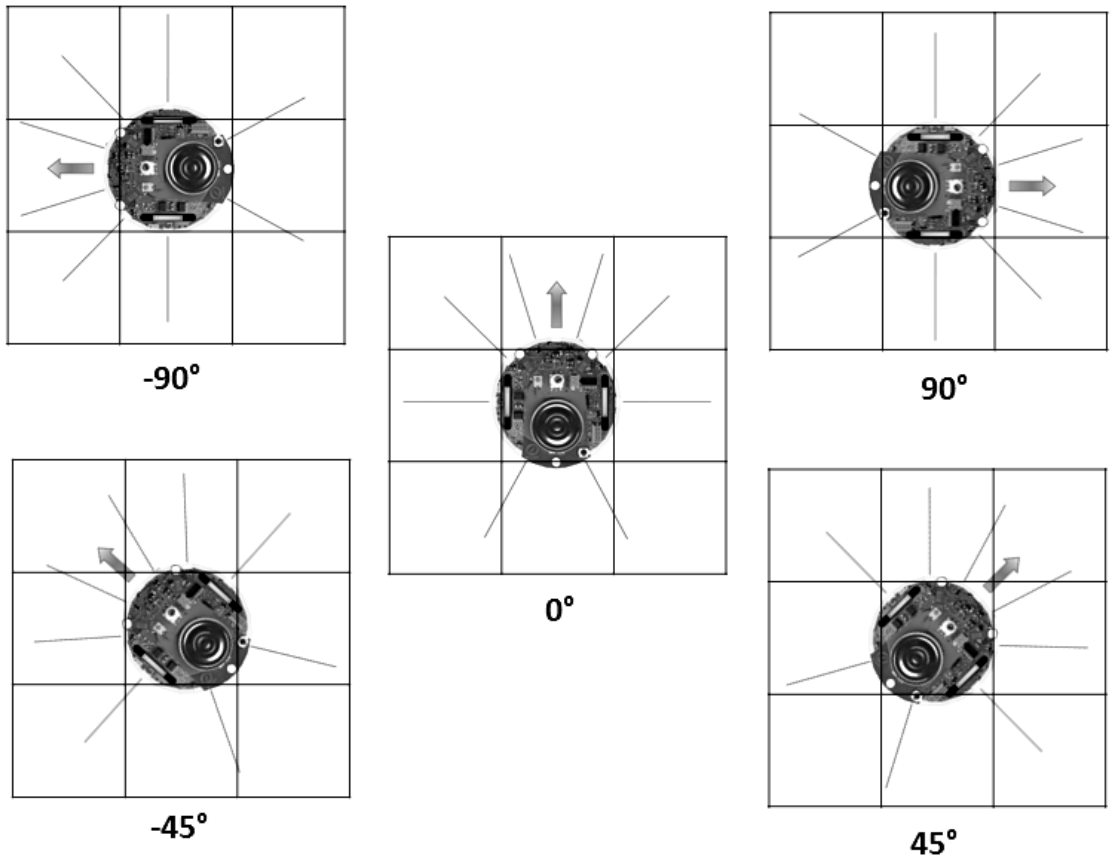

Figura 41 – Cenários de atualização da vizinhança, incluindo os cinco sentidos de movimentação do robô.

comportamento do robô quanto ao desvio de quinas de obstáculos utilizando os estados Robô_Formação e Robô_Formação_Diagonal. As novas regras já levam em consideração todas as situações de vizinhança e foram incorporadas ao modelo do AC. A Tabela 3 apresenta o novo conjunto de regras para desvio de obstáculos.

Vale destacar que a Tabela 3 apresenta regras para pares de vizinhanças, ou seja, a atualização deve ser feita de modo a representar a movimentação do robô pelo ambiente em um passo de tempo. Por exemplo, se considerarmos a regra referente ao caso em que o estado da célula centra é Robô_Formação_i, se o robô estiver em um ângulo de 0°e a célula da frente (**a1**) estiver livre, o robô executará um passo no mesmo sentido (0°) e a célula central em que ele estava passará ao estado Livre. Simultaneamente, o estado da célula **a1** deve ser atualizado considerando o deslocamento do robô para essa célula. Para isso, a regra referente ao caso em que o estado da célula central é Livre e o ângulo do robô é 0°, modifica o estado da célula central para Robô_Formação_i, sendo que o valor i depende do estado da célula central **a5** (que é célula centra da vizinhança anterior). As regras envolvidas no exemplo anterior estão destacadas em negrito na Tabela 3. Um conceito similar se aplica às regras para controle de formação, apresentadas na Tabela 4,

Tabela 3 – Novo Conjunto de Regras para Desvio de Obstáculos

Estado	Ângulo	Vizinhança	Novo Estado	Novo Ângulo
Livre	**0°**	a5 = Robô & a3 & a7 = Livre	Robô	0°
		a5 = Robô_Rotacionado_3 & a3 & a7 = Livre	Robô_Formação_0	
		a5 = Robô_Formação_0 & a3 & a7 = Livre	**Robô_Formação_1**	
		a5 = Robô_Formação_1 & a3 & a7 = Livre	**Robô_Formação_2**	
		a5 = Robô_Formação_2 & a3 & a7 = Livre	**Robô_Formação_3**	
		Caso Contrário	Livre	
	-45°	a5 = Robô_Formação_Diagonal	Robô_Formação_Diagonal	-45°
		Caso Contrário	Robô	0°
	45°	a5 = Robô_Formação_Diagonal	Robô_Formação_Diagonal	45°
		Caso Contrário	Robô	0°
	-90°	a3 = Robô & a5 = Livre & a2 = Obstáculo	Robô	-90°
		a3 = Robô_Rotacionado_0	Robô_Rotacionado_1	
		a3 = Robô_Rotacionado_1	Robô_Rotacionado_2	
		a3 = Robô_Rotacionado_2	Robô_Rotacionado_3	
		Caso Contrário	Livre	
	90°	a7 = Robô & a5 = Livre & a8 = Obstáculo	Robô	90°
		a3 = Robô_Rotacionado_0	Robô_Rotacionado_1	
		a3 = Robô_Rotacionado_1	Robô_Rotacionado_2	
		a3 = Robô_Rotacionado_2	Robô_Rotacionado_3	
		Caso Contrário	Livre	
Obstáculo	Qualquer	Qualquer	Obstáculo	Qualquer
Robô	0°	a1 & a2 & a8 = Livre	Livre	0°
		a1 = Obstáculo & a2 & a3 = Livre	Robô	90°
		a1 & (a2 OR a3) = Obstáculo & a7 & a8 = Livre	Robô	-90°
		a1 & a7 = Obstáculo & a3 = Livre	Robô	90°
		a1 & a3 = Obstáculo & a7 = Livre	Robô	-90°
		a8 = Obstáculo & a3 = Livre	Robô	90°
		a2 = Obstáculo & a7 = Livre	Robô	-90°
	-90°	a1 = Obstáculo & a7 & a6 = Livre	Livre	-90°
		a1 = Livre	Robô_Rotacionado_0	0°
	90°	a1 = Obstáculo & a3 & a4 = Livre	Livre	-90°
		a1 = Livre	Robô_Rotacionado_0	0°
Robô_Rotacionado_i	-90°	a7 = Livre	Livre	-90°
		Caso Contrário (Não Existe Caminho)	Robô_Rotacionado_0	0°
	90°	a3 = Livre	Livre	90°
		Caso Contrário (Não Existe Caminho)	Robô_Rotacionado_0	0°
Robô_Formação_i	**0°**	**a1 = Livre**	**Livre**	**0°**
		Caso Contrário	Robô	0°
Robô_Formação_Diagonal	-45°	a1 & a7 & a8 = Livre	Livre	-45°
		Caso Contrário	Robô	0°
	45°	a1 & a2 & a3 = Livre	Livre	45°
		Caso Contrário	Robô	0°

ou seja, elas sempre sevem ser aplicadas aos pares: uma para a vizinhança cuja célula central está como o robô no início da transição (estado **r** ou **rf-3** ou **rfd**) e outra para a vizinhança da céula que está no estado **livre** e que receberá o robô após a movimentação.

Tabela 4 – Novo Conjunto de Regras para Controle de Formação

Células no tempo t											Estado de a0 no tempo t+1	
Caso	**a0**	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**Ângulo**	**a1**	**Novo Ângulo**
hor - x = 0	*r*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	0°	*l*	0°
hor - x = 0	*l*	*l*	*l*	*l*	*l*	*r*	*l*	*l*	*l*	0°	*r*	0°
hor - x >0	*r* ou *rf_3*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	0°	*l*	45°
hor - x >0	*l*	*l*	*l*	*l*	*l*	*r* ou *rf_3*	*l*	*l*	*l*	0°	*rfd*	45°
hor - x <0	*r* ou *rf_3*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	0°	*l*	-45°
hor - x <0	*l*	*l*	*l*	*l*	*l*	*r* ou *rf_3*	*l*	*l*	*l*	0°	*rfd*	-45°
hor - x >0	*rfd*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	45°	*l*	45°
hor - x >0	*l*	*l*	*l*	*l*	*l*	*l*	*rfd*	*l*	*l*	45°	*rfd*	45°
hor - x <0	*rfd*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	*l*	-45°	*l*	-45°
hor - x <0	*l*	*l*	*l*	*l*	*l*	*l*	*rfd*	*l*	*l*	-45°	*rfd*	-45°

4.2.3 Classificação Evolutiva da Vizinhança

Após as melhorias efetuadas no modelo do AC descritos na subseção 4.2.1, com o objetivo de melhorar a qualidade da trajetória no desvio de obstáculos, focamos em outra limitação identificada no modelo descrito em (FERREIRA, 2014): a identificação de obstáculos a partir da leitura dos sensores infravermelhos. A definição da vizinhança é um fator determinante no sucesso do modelo de navegação baseado em AC, e influencia diretamente na qualidade da trajetória percorrida. Tal tarefa deve ser executada de forma robusta, sendo capaz de se adaptar à dinamicidade do ambiente. Com esse intuito, ao invés de utilizar parâmetros pré-fixados, como nos métodos anteriores, para definir os limites nas leituras dos sensores, desenvolvemos um AG para classificação automática das células vizinhas, levando em consideração também a luminosidade do ambiente.

O primeiro passo para o desenvolvimento do AG foi a construção de uma base de dados com valores de leituras dos sensores em diferentes situações e ambientes. Para cada cenário, além do registro dos sensores, também foi armazenada a classificação da situação da vizinhança na qual essas leituras foram realizadas. Foram definidas 6 classes, em relação à identificação de obstáculos (ou não) na vizinhança: *(C1)* vizinhança livre, *(C2)* obstáculo a norte, *(C3)* obstáculo a nordeste, *(C4)* obstáculo a noroeste, *(C5)* obstáculo a leste e *(6)* obstáculo a oeste. Criamos diversos cenários que representassem essas classes de vizinhança e, utilizando os robôs e-puck reais, extraímos 980 registros das medições dos sensores no ambiente, atribuindo a cada registro uma classe correspondente. A extração

dos valores dos sensores foi executada utilizando-se iluminação controlada e com intervalos constantes de 2 segundos entre cada medição, sendo considerados obstáculos os objetos à uma célula de distância do robô.

Para cada cenário, as medições foram realizadas em 5 ângulos diferentes quanto ao obstáculo, ou seja, o objeto utilizado foi o mesmo porém a orientação do robô foi alterada para coincidir com os ângulos de movimentação (-90°, -45°, 0°, 45°e 90°), gerando medições para cada um dos cenários nos diferentes ângulos, de forma que a base de registros ficasse independente de posicionamento. A base de dados, composta por todos os registros e suas respectivas classes foi então organizada no esquema apresentado pela Tabela 5. Os sensores traseiros s3 e s4 não foram utilizados e portanto não estão presentes na Tabela 5.

Tabela 5 – Esquema da Base de Dados dos Sensores

Medição	s0	s1	s2	s5	s6	s7	Classe
1	**179**	**231**	**210**	**328**	**160**	**134**	**C1**
2	**178**	**229**	**211**	**327**	**160**	**133**	**C1**
3	**584**	362	202	312	237	**505**	**C2**
4	**587**	365	204	312	239	**498**	**C2**
5	139	316	458	263	118	98	C5
6	138	314	452	258	114	95	C5
7	114	150	141	735	287	112	C6
...	...	...	...	...	...	...	...
980	115	147	141	730	288		C6

Os sensores infravermelho dos e-puck retornam valores inteiros. Quanto maior o valor obtido por um sensor, maior é a proximidade de um obstáculo. Como podemos observar na Tabela 5, para a classe C1 (vizinhança livre) os valores deveriam ser 0, pois representam uma situação na qual não há obstáculos próximos. Porém, os valores obtidos pelos sensores são na verdade afetados por ruídos do ambiente, de tal maneira que mesmo em um cenário sem obstáculos os sensores apresentam leituras superiores à 0. Mesmo com a presença de ruídos nas leituras dos sensores ainda é possível fazer uma classificação correta da situação da vizinhança. Na classe C2, por exemplo, os valores dos sensores s0 e s7 (frontais) foram mais altos, indicando a provável presença de um obstáculo à frente.

Após a elaboração dessa base com dados reais de leitura, utilizamos um AG para

gerar as regras automáticas de classificação da vizinhança a partir da base de dados dos sensores. A utilização de AGs para fazer a mineração de regras de classificação a partir de uma base de registros contendo dados de casos reais tem sido largamente utilizada na literatura de *data mining* (FREITAS, 2003). Essa abordagem tem sido utilizada em diversos problemas, tais como: diagnóstico de doenças de pele (FIDELIS; LOPES; FREITAS, 2000), bioinformática (AMARAL et al., 2008) e chaves criptográficas (OLIVEIRA et al., 2010). O desenvolvimento do AG implementado nessa dissertação foi realizado a partir do modelo proposto em (SILVA; RIBEIRO; AMARAL, 2013), que foi implementado pelo próprio autor dessa dissertação em um problema de bioinformática.

Cada indivíduo do AG é composto por n genes (n é o número de sensores de distância no robô) e sua classe correspondente, sendo que cada gene é subdividido em 3 campos: índice (I_i), operador (O_i) e valor (V_i). O indivíduo como um todo é a parte antecedente da regra (IF) e cada gene corresponde a uma condição. O campo I_i representa o número do sensor do robô, o campo O_i pode conter o operador $<$ ou $\geq$ e o campo V_i contém um número inteiro que pode variar entre o limite inferior e superior da leitura feita para aquele determinado sensor. A parte consequente da regra (THEN) é determinada após a execução do algoritmo, sendo uma das 6 possíveis classes de vizinhança. Essa abordagem é conhecida como Michigan (FREITAS, 2003). Nessa abordagem, a cada execução do AG, todas as regras da população representam uma única classe. Assim, para que as 6 regras sejam obtidas, é necessário executar o AG 6 vezes.

A função de aptidão, que mede a qualidade da regra como solução do problema, usa conceitos baseados em quatro tipos diferentes de resultados, dependendo da classe prevista pela regra e a verdadeira classe da amostra:

- ❏ Verdadeiro Positivo (vp) - A regra identifica a amostra como pertencente a uma determinada classe e a amostra de fato pertence a essa classe.

- ❏ Falso Positivo (fp) - A regra identifica a amostra como pertencente a uma determinada classe, mas a mesma não pertence a essa classe.

- ❏ Verdadeiro Negativo (vn) - A regra identifica a amostra como não pertencente a uma determinada classe e a amostra é de fato de outra classe.

- ❏ Falso Negativo (fn) - A regra identifica a amostra como não pertencente a uma determinada classe, mas a amostra pertence à classe em questão.

A partir desses quatro indicadores, que são calculados para todos os registros da base, considerando-se a regra que está sendo avaliada, o valor de *fitness* obtido pela Equação 1:

$$Fitness = \frac{(vp + vn)}{vp + vn + fp + fn} \tag{1}$$

Para maximizar a aptidão de cada indivíduo, o objetivo é aumentar tanto o número de registros classificados como *vp* quanto *vn*. Por meio de vários testes de configurações o conjunto de parâmetros para o AG adotado foi 50 gerações, 200 indivíduos, 6 genes, torneio estocástico de *tour* 3, *crossover* múltiplo com probabilidade de 100%, taxa de mutação de 30% e reinserção por elitismo (GOLDBERG, 1989).

Incluímos a capacidade de adequação à iluminação por meio de um parâmetro de luminosidade, que é extraído da primeira leitura dos sensores. Sendo **agV** o valor presente nas regras geradas pelo AG, **agR** a referência de luminosidade do *dataset* e (**rR**) o parâmetro de luminosidade obtido de cada sensor, em tempo real, obtemos o valor para o *threshold* de detecção de obstáculo. Como mostrado na Equação 2.

$$Threshold = \frac{agV}{agR} * rR \tag{2}$$

A regra considera a referência de luminosidade específica para cada sensor e multiplica pelo parâmetro de detecção de obstáculo. Com a aplicação dessa abordagem, o método se tornou mais independente de parâmetros fixos e passou a responder dinamicamente à diferentes iluminações do ambiente. Todas as regras geradas atingiram 100% de acurácia (*aptidão*=1 de acordo com a Equação 1) durante a evolução do AG. A Tabela 6 apresenta as regras obtidas para cada classe evoluída.

Tabela 6 – Conjunto de Regras Geradas pelo AG

Classe	Regra	Aptidão
C1:Vizinhança Livre	SE s0 < 2.68*rR1 E s5 < 1.15*rR5 E s1 < 1.47*rR1 E s2 < 1.41*rR2 E s6 < 1.62*rR6 E s7 < 2.82*rR7	1
C2: Obstáculo a Norte	SE s0 >= 2.68*rR1 OU s7 >= 2.82*rR7	1
C3: Obstáculo a Nordeste	SE s1 >= 1.47*rR1	1
C4: Obstáculo a Noroeste	SE s6 >= 1.62*rR6	1
C5: Obstáculo a Leste	SE s2 >= 1.41*rR2	1
C6: Obstáculo a Oeste	SE s5 >= 1.15*rR5	1

Após a execução do AG, as regras de classificação da vizinhança evoluídas foram adaptadas ao modelo baseado em AC, criando uma camada de inteligência quanto à definição do estado das células. Para isso, acoplamos as regras de classificação ao ciclo de execução e movimentação do robô, como mostra o Algoritmo 1.

Algoritmo 1 Classificação Evolutiva da Vizinhança

```
1: while Meta Não Atingida do
2:     procedure Definição Vizinhança(regrasAG, sensores)
3:         leitura ← ExtrairValores(sensores)
4:         vizinhança = classificaVizinhança(leitura, regrasAG)
5:     procedure Aplica AC(vizinhança)
6:         if Obstáculo Detectado then
7:             desviaObstáculo(vizinhança)
8:         else
9:             controleFormação(vizinhança)
```

Desta forma, durante a aplicação das regras de transição do AC do modelo, os dados extraídos dos sensores do e-puck são avaliados pelas regras de classificação geradas pelo AG. O estado de cada célula é atualizado e de acordo com o estado da vizinhança do robô, aplica-se as regras de desvio de obstáculo ou controle de formação.

4.3 Inovações aplicadas ao modelo utilizando o time de robôs

A seguir serão descritas as modificações realizadas no aspecto cooperativo do método, levando-se em consideração os impactos das ações de cada robô no padrão espacial do time e da comunicação para a manutenção da coesão da formação.

4.3.1 Limitação dos modelos anteriores

A estrutura líder-seguidores dos métodos propostos em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e (FERREIRA, 2014), quando aplicada em um time de robôs, demonstrou uma rigidez indesejável. Nesses modelos precursores, o líder é responsável por todo processo de decisão quanto à manutenção da formação e a comunicação entre robôs é realizada a cada passo de tempo. Utilizando essa estrutura, os escravos enviam ao mestre seu estado e seus dados de posicionamento, o mestre processa toda a formação de forma centralizada e retorna a ação que cada escravo deve executar.

Além disso, foi observado que os dois modelos anteriores (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e (FERREIRA, 2014) adotam estratégias diferentes quanto à manutenção da formação, no advento de um desvio de obstáculo por parte de um membro do time. Enquanto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), o modelo adota a inversão do eixo de deslocamento quando dois robôs se aproximam, no modelo proposto em (FERREIRA, 2014), cada robô do time deve sempre retornar ao eixo de deslocamento original, após o desvio do obstáculo. Observamos nas simulações e experimentos com robôs reais, que a adoção de uma mesma estratégia (fixa) pode não ser interessante para

diferentes situações. Em alguns casos, observamos que a inversão dos eixos pode auxiliar os robôs a retornarem mais rápido para a formação, enquanto em outros, observamos que essa inversão pode elevar o tempo necessário para a retomada da formação.

4.3.2 Nova estrutura de comunicação do time

A partir da crítica a respeito da rigidez dos modelos anteriores, uma abordagem diferente foi adotada para aumentar a independência do time e reduzir a quantidade de mensagens trocadas entre robôs.

4.3.2.1 Descentralização das decisões

Com o objetivo de alterar a estrutura de comunicação, propomos uma nova abordagem, na qual o poder de decisão centralizado do mestre foi removido. A nova função do robô central passa a ser a de um coordenador. Desta forma, os robôs vizinhos apenas enviam seus identificadores e o número de passos dados na direção desejada. Com essas informações, o robô coordenador compara a posição recebida de cada robô com os parâmetros de formação e com seus próprios dados de movimentação. Ao final, o coordenador responde com um *flag* binário indicando se o padrão espacial está mantido ou não. Os vizinhos recebem a resposta, decidem localmente qual ação devem tomar e selecionam a correspondente regra do AC. Assim, a formação do time é sincronizada e a decisão é tomada com autonomia em cada robô. Na nova abordagem, a frequência na troca de informações também foi reduzida, utilizando mensagens periódicas a cada 3 passos de tempo.

4.3.2.2 Investigação sobre os protocolos de comunicação

Para implementar a comunicação no time de robôs reais, primeiramente investigamos qual protocolo de comunicação seria mais apropriado. Em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), a comunicação entre o robô mestre e os escravos foi realizada através do protocolo Bluetooth. Em (FERREIRA, 2014), por outro lado, os experimentos com robôs reais não foram realizados. No presente trabalho, investigamos dois protocolos: Bluetooth e Wifi.

Bluetooth

Em simulação, a comunicação entre robôs foi feita utilizando as bibliotecas *emitter* e *receiver*, provenientes do pacote original do software Webots. Porém, essas funções não estão implementadas para o modo de cross-compilação (geração do código a ser embarcado no robô real), o que impossibilitaria a troca de mensagens em cenários reais. Decidimos desenvolver as funções necessárias ao Webots para que essa impossibilidade fosse

superada. Utilizando como base o trabalho (COUCEIRO; VARGAS; ROCHA, 2014), implementamos todas as funções para a troca de mensagens utilizando o protocolo Bluetooth e integramos às bibliotecas do Webots para que a cross-compilação fosse executada de forma transparente e gerasse o código a ser embarcado nos robôs e-pucks. Conforme os experimentos apresentados na seção 5.2, embora tenha sido possível implementar a comunicação através do Bluetooth, para o experimento de navegação do time, esse protocolo se mostrou inviável devido ao alto tempo de processamento e atraso na comunicação.

Wifi

O segundo protocolo analisado foi o Wifi, que apesar de oferecer maior rapidez nas trocas de mensagens, também não é suportado no processo de cross-compilação do software Webots. Assim, foi realizada a implementação de todas as funções necessárias para que a comunicação fosse realizada a partir dos robôs e-pucks reais.

A placa Overo Gumstix COM foi incorporada na parte de cima dos e-pucks, fornecendo diversas funcionalidades adicionais, entre elas a comunicação Wifi. Nesse tipo de comunicação foi necessária a utilização de um servidor, desenvolvido no trabalho de (COUCEIRO; VARGAS; ROCHA, 2014), para que houvesse a troca de mensagens. Esse servidor é executado em um computador externo e redireciona as mensagens enviadas pelos robôs para os respectivos destinatários. Apesar de centralizar as mensagens dos robôs e-pucks (que agem como clientes TCP/IPs), o servidor permite simular diversas características das redes wireless, como o número máximo de saltos. Após a realização dos experimentos detalhados na seção 5.2, o protocolo de comunicação adotado em nosso ambiente foi o Wifi.

4.3.3 Otimização Bioinspirada da Reconfiguração da Formação

No modelo proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) cada robô, para preservar a formação, deve manter uma distância pré-definida dos outros, porém, caso um dos robôs execute uma manobra de desvio de obstáculo essa distância provavelmente será violada. Quando isso acontece entra em execução a estratégia de troca de posição, fazendo com que o robô que está se aproximando troque de lugar na formação com outro robô. As funções de cada robô na formação também são alteradas, de modo que se um robô mestre realizar uma troca de posição com robô vizinho, ele perderá as funções de mestre e repassará essa tarefa para o robô que tomar sua antiga posição na formação. Todo o processo de decisão de troca de posições e de funções na formação é realizado por meio de comunicação entre os robôs. As mensagens são enviadas a todos os robôs, informando a nova configuração de posições e quem é o novo robô mestre. Desta forma, todo o time fica sincronizado sobre a estrutura do time. A decisão de iniciar essa

estratégia é tomada baseada somente no parâmetro pré-definido de distância entre robôs na formação, bastando que 2 robôs se aproximem para que a mesma seja aplicada.

Foi apontado em (FERREIRA, 2014), que essa estratégia pode apresentar um problema crítico que impossibilita o time de robôs de alcançar a meta, o pareamento de robôs. Quando essa situação ocorre, os dois robôs envolvidos na troca consideram um ao outro como obstáculos e acabam se movimentando indefinidamente em frente aplicando a regra de desvio. O pareamento ocorre devido aos dois robôs estarem realizando a manobra de troca de posição e se encontrarem no meio do caminho. Em (FERREIRA, 2014) os autores resolveram esse problema retirando completamente o processo de troca de posição do comportamento dos robôs. Ou seja, no caso de ocorrer a aproximação de 2 robôs no desvio de obstáculo, o robô que não estiver desviando do obstáculo simplesmente irá aguardar que o outro robô faça o desvio e comece a se deslocar em direção à meta novamente. O robô que estiver fazendo o desvio, por sua vez, irá contornar o obstáculo e retornar ao seu eixo original de deslocamento. Nessa estratégia não existe troca de papeis de mestre e escravo entre os robôs.

A partir da análise do problema do pareamento na estratégia de troca de eixos adotada em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), verificamos que esse problema pode ser resolvido da seguinte forma: (*i*) o robô que está desviando de um obstáculo deve comunicar sua intenção de trocar de posição na formação assim que iniciar o desvio; (*ii*) o robô que irá realizar a troca no eixo de deslocamento, após receber a mensagem, deve iniciar seu movimento em diagonal para o novo eixo imediatamente. Dessa forma, até que o robô que está contornando o obstáculo atinja a sua borda, o outro robô já realizou alguma progressão na direção da meta e os robôs não correm mais o risco de se posicionarem lado a lado ao se deslocarem para seu novos eixos.

Adotando essa abordagem, o problema do pareamento é resolvido e a troca pode ser realizada como proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c). Entretanto, ao analisar em simulação as duas estratégias (com troca e sem troca de eixos) observamos que existem cenários em que a troca provoca um retorno mais rápido à formação do time, enquanto em outros cenários a troca pode aumentar o tempo para retomada da formação.

Sendo assim, decidimos investigar se a adoção de uma estratégia mista, na qual a troca pode ou não ser aplicada dependendo da situação, traria vantagens ao modelo de navegação com controle de formação. Identificamos experimentalmente que a característica da situação onde se deveria ou não aplicar a troca, estaria relacionada à distância relativa de aproximação entre os robôs. Basicamente, nos casos em que o robô se desloca um pouco do eixo para o desvio do obstáculo, é melhor que o mesmo apenas contorne o obstáculo e os demais membros aguardem seu desvio para que reiniciem o movimento em direção à meta. Por outro lado, caso o robô que desvia se desloque significativamente do seu eixo durante o desvio, pode ser melhor que seja realizada a troca com o robô onde

ocorre a aproximação. O problema está em identificar que valor de deslocamento seria significativo a ponto de justificar a reconfiguração da formação com a troca de posições.

Com o objetivo de encontrar a melhor abordagem para solucionar esse problema, criamos um AG responsável pela otimização da reconfiguração da formação. O AG evolui o valor de dois parâmetros que representam o melhor momento para se realizar uma troca de posição no time. Esses parâmetros foram criados com o objetivo de se adaptar dinamicamente a estratégia de controle de formação. São eles:

- **fator de aproximação (fap)**, representa a distância máxima permitida entre robôs para que seja iniciada a manobra de troca de posições;
- **fator de avanço (fav)**, identifica a quantidade de passos necessários para manutenção da formação do robô que não está envolvido na manobra de troca de colunas;

Como podemos ver na Figura 42, o fator de aproximação é responsável por regular o quanto um robô pode se aproximar do eixo de deslocamento de seu vizinho. Essa restrição é utilizada sempre que algum robô desvia de eixo devido a algum obstáculo encontrado no meio do caminho. No modelo proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) esse parâmetro é definido como zero, ou seja, não pode haver nenhuma aproximação entre os robôs sem que seja iniciada uma manobra de troca de posições. No modelo proposto em (FERREIRA, 2014), o fator de aproximação é infinito, ou seja, nunca haverá a troca de eixos de deslocamento entre robôs.

Figura 42 – Fator de aproximação entre robôs.

Outro fator importante para a manutenção da formação é o comportamento do robô que não está participando da manobra de troca de posições. A Figura 43 apresenta essa situação. Podemos observar que, enquanto os dois robôs mais a esquerda estão realizando uma reconfiguração na formação, o robô mais a direita realizará uma movimentação frontal utilizando o fator de avanço, acompanhando o deslocamento dos outros robôs. Isso permite que a formação em linha seja mantida por mais tempo, mesmo que os robôs ainda estejam alternando posições.

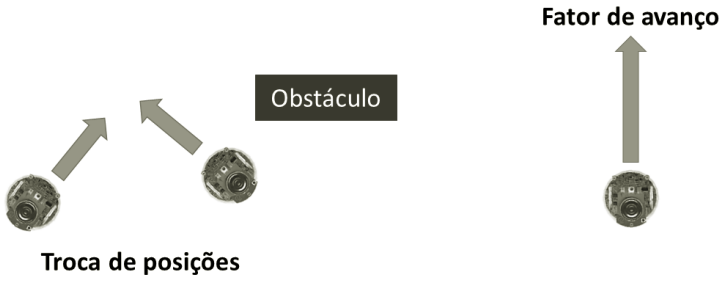

Figura 43 – Fator de avanço determinado para a manobra de troca de posições.

Além desses dois parâmetros, utilizamos o AG também para encontrar os valores das velocidades mais adequadas ao deslocamento do time. São elas:

- **velocidade de deslocamento (vd)**, representa a velocidade na qual as rodas do e-puck irão realizar o deslocamento;
- **velocidade de rotação (vr)**, refere-se à velocidade de movimentação ao redor do próprio eixo.

Dessa forma, o indivíduo evoluído pelo AG é formado por 4 parâmetros (**fav**, **fap**, **vd** e **vr**), conforme ilustrado a Figura 44. Utilizamos uma distância entre os eixos originais de deslocamento de 25 centímetros, desta forma o valor do parâmetro **fap** pode variar de 0 até 25. Para o parâmetro **fav** definimos o domínio de valores de 0 à 20 centímetros (obtido através de testes preliminares em simulação). As velocidades de movimentação não foram descritas no modelo proposto em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c); no modelo de (FERREIRA, 2014) a velocidade utilizada foi de 200 radianos por segundo, desta forma definimos o intervalo de 100 a 300 radianos por segundo para os parâmetros **vd** e **vr**.

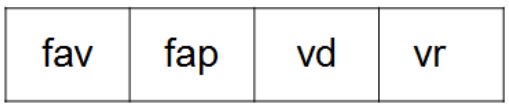

Figura 44 – Indivíduo do AG.

A função de aptidão deve refletir a qualidade dos parâmetros na navegação do time. O objetivo é avaliar qual combinação de parâmetros (indivíduo) apresenta o melhor resultado quando aplicado na estratégia de controle formação dos robôs e-puck. O melhor indivíduo é aquele que resulta na maximização do tempo de formação e do avanço em time em direção à meta, enquanto minimiza a quantidade de rotações executada por cada robô. O cálculo é realizado utilizando a Equação 3:

$$Fitness = \frac{(peso1 * formação + peso2 * distância + peso3 * rotações)}{(peso1 + peso2 + peso3)} \quad (3)$$

Na Equação, são considerados os seguintes parâmetros:

- **formação**: refere-se a porcentagem de tempo no qual o time se manteve em formação durante todo o experimento. A formação é considerada intacta caso não haja movimentação superior a 3 passos fora do eixo de deslocamento de cada robô. A cada passo de tempo, se a formação não foi desfeita é contabilizado como tempo em formação, ao final a porcentagem é dada pela divisão do tempo em formação pelo tempo total de simulação.

- **distância**: identifica a porcentagem da distância percorrida pelo time na direção da meta, o cálculo é feito considerando os passos realizados pelo time dividido pela quantidade de passos total até a meta;

- **rotações**: contabiliza a porcentagem de rotações realizadas durante o experimento, o cálculo é realizado através da divisão do número de rotações pelo número de rotações máximo naquele cenário, caso não haja nenhuma rotação o valor retornado é 1 e caso o número de rotações supere o valor máximo, é retornado o valor 0.

Como podemos observar na Figura 45, o processo se inicia com a entrada dos parâmetros genéticos e ajuste do módulo de configuração. Após essa fase, o módulo de evolução é iniciado, representando todas as etapas do AG, desde a formação da população inicial de indivíduos até o resultado final da evolução que é retornado para o usuário.

Como parte da fase de avaliação, o AG repassa o indivíduo e inicia a simulação no Webots. Utilizamos um tempo máximo para execução da simulação de dois minutos (este valor pode ser ajustado para outros tipos de cenário). Mesmo que o time não tenha completado o desvio dos obstáculos, se o tempo limite for atingido a simulação é finalizada e a o cálculo da *fitness* é realizado. Desta forma, caso o time de robôs consiga ultrapassar os obstáculos e retornar para a formação dentro do tempo permitido, sua aptidão será maior e consequentemente representará um melhor indivíduo. A aptidão gerada pela execução da simulação é então repassada ao AG, que utiliza o valor para atribuir a *fitness* ao indivíduo e continuar com o processo de otimização, ou seja, a simulação do Webots retorna ao AG o resultado da aplicação da Equação 3. Diferentes valores dos pesos (peso1, peso2 e peso3), utilizados na Equação 3, foram investigados nos experimentos descritos na seção 5.3.

4.4 Modelo Resultante

O modelo de navegação aqui proposto pode ser aplicado tanto na navegação individual de um robô sobre um eixo desejado de deslocamento quanto na navegação de um time de

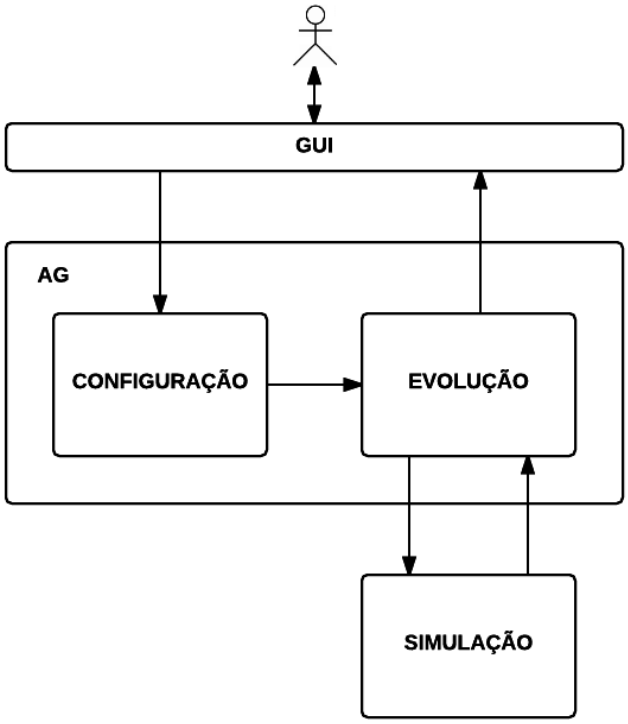

Figura 45 – Arquitetura do AG em conjunto com o simulador Webots

robôs em uma formação desejada (linear ou triangular). O ambiente deve ser discretizado em uma grade retangular de células idênticas (com tamanho de 1 centímetro). Não é necessária a informação completa do ambiente, sendo que os robôs utilizam somente os sensores para detectar e desviar de obstáculos em tempo real.

O modelo é baseado no conceito de regras de transição de ACs, sendo que a vizinhança utilizada foi a de Moore. O conjunto de estados que cada célula pode obter é composto por: Livre, Obstáculo, Robô, Robô_Rotacionado_i, Robô_Formação_i e Robô_Formação_Diagonal, com i podendo variar de 1 a 4. A vizinhança do robô deve ser determinada a cada passo de tempo por meio das leituras dos sensores de distância e da aplicação de um conjunto de regras que determina a presença ou não de um obstáculo em cada célula da vizinhança. As regras, apresentadas na Tabela 6 são utilizadas para a classificação automática das células. Essas regras foram geradas por um AG a partir de uma base com dados de leituras dos sensores em situações de presença de obstáculos. Elas também levam em consideração a luminosidade do ambiente. Para isso, o robô coordenador deve ser iniciado em uma posição livre de obstáculos para que o método faça uma leitura de referência que servirá para corrigir a luminosidade corrente do ambiente. Com a utilização dessa camada inteligente de identificação do estado das células vizinhas, o método se tornou mais independente de parâmetros fixos e passou a responder dinamicamente à diferentes iluminações do ambiente.

A decisão de movimentação do robô é realizada a cada passo de tempo por uma regra de transição com base na vizinhança identificada, na posição atual do robô estimada pela odometria e na informação recebida pelo coordenador a respeito da formação do time. A troca de informação com o robô coordenador é feita a cada três passos de tempo: cada robô do time envia sua posição estimada e o robô coordenador devolve se o padrão está ou não mantido. Basicamente, as regras do AC se dividem em dois conjuntos: (*i*) evitar colisão, que é ativado quando um obstáculo é detectado e os robôs agem de forma reativa e autônoma (3); e (*ii*) controle de formação, aplicado quando não há obstáculos na vizinhança fazendo com que o robô aja de forma cooperativa com o objetivo de manter sua formação original (4). Os conjuntos de regras de transição adotados nesse novo modelo permitiram uma trajetória mais adequada dos robôs em relação a modelos precursores, resultando no comportamento do time esperado para essa tarefa.

Como resultado da aplicação da regra de transição em determinada vizinhança, os robôs podem executar dois tipos de ações: (*i*) manter sua orientação atual e deslocar-se para uma célula vizinha, percorrendo uma distância equivalente ao tamanho de uma célula (1 cm), ou em caso de movimentação diagonal a distancia será igual a raiz do valor correspondente ao tamanho de duas células ($\sqrt[2]{2}$); ou (*ii*) realizar uma rotação sobre seu eixo (-90°, -45°, 45°ou 90°), mantendo sua posição atual.

Utilizamos um time composto por 3 robôs separados por uma distância de 25 centímetros em uma formação linear. O robô que está na posição mais interior da formação assume o papel de coordenador. O robô coordenador analisa as informações trocadas com seus vizinhos e assegura que o padrão da formação seja mantido. Assim, a formação do time é sincronizada e a decisão é tomada com autonomia em cada robô. Para efetuar o processo de troca de mensagens entre os robôs e-puck, nosso método adota o protocolo Wifi, utilizando a placa Overo.

O modelo permite a adoção de três estratégias quanto à reconfiguração da formação. A primeira abordagem, também adotada em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), sempre reconfigura a formação quando ocorre uma aproximação entre robôs; nesse caso, a troca dos eixos de referência será realizada e consequentemente os robôs irão se deslocar para um novo eixo de deslocamento. A segunda abordagem, também adotada em (FERREIRA, 2014), nunca realiza a troca de eixos, os robôs apenas contornam o obstáculo enquanto os demais membros aguardam seu desvio. A terceira abordagem é híbrida, na qual a troca de posições entre robôs pode ou não ser aplicada dependendo da situação. Nessa abordagem, o time de robôs pode apresentar os dois comportamentos, dependendo principalmente na distância de aproximação entre os robôs do time. Os parâmetros adotados para essa abordagem foram evoluídos a partir de um AG.

Capítulo 5

Experimentos e Análise dos Resultados

Nesse capítulo, vamos descrever os experimentos realizados para análise do comportamento dos robôs no modelo proposto e avaliar os resultados em diferentes cenários. Foi utilizado um laptop Sony Vaio com 6 GB de memória RAM e um processador Intel i5-3210M com 2,50 GHz. O sistema operacional utilizado foi o Windows 7 Home Premium de 64 bits. Primeiramente, nos concentramos no comportamento individual e, em seguida, avaliamos o desempenho do método aplicado ao time de robôs.

5.1 Experimentos com um único robô

Os experimentos foram realizados no ambiente de software Webots e em um ambiente físico. Grande parte dos testes prévios foram feitos no Webots, onde a maioria dos ajustes foram realizados, enquanto o desempenho final do sistema foi validado com os robôs físicos reais. O objetivo dos experimentos descritos nesta seção foi avaliar o desempenho do modelo proposto após as alterações nas regras e estados do AC, conforme descrito na subseção 4.2.1, e também a utilização das regras de classificação da vizinhança apresentadas na Tabela 6.

A fim de testar a robustez do método, criamos três cenários virtuais contendo diferentes tipos de obstáculos, variando-se de objetos poligonais até formas cilíndricas. O ambiente pode ser visto como estático, uma vez que só pode ser alterado por ações dos robôs, cujo resultado pode afetar o ambiente para os outros. Ele também é não-determinista e parcialmente observável, porque um determinado robô não pode prever o próximo passo do sistema. Portanto, é impossível para qualquer agente obter informações completas do cenário.

5.1.1 Testes em simulação

A Figura 46 apresenta os três cenários virtuais de teste e as trajetórias obtidas por um robô e-puck ao utilizar: (a) o modelo proposto em (FERREIRA, 2014) e (b) o novo

modelo. As formas dos obstáculos variam em cada cenário de simulação, proporcionando níveis distintos de dificuldade na manobra de desvio. O primeiro cenário é composto por um obstáculo único no formato retangular; o segundo contém um obstáculo no formato cilíndrico; e o terceiro é composto por dois obstáculos quadrados. O objetivo da simulação é avaliar o desempenho do robô no desvio dos obstáculos. O robô deve iniciar sua trajetória no ponto indicado por **x** e navegar sobre seu eixo de deslocamento até identificar o obstáculo a sua frente. Após o desvio do obstáculo, o robô deve retornar ao seu eixo de deslocamento original.

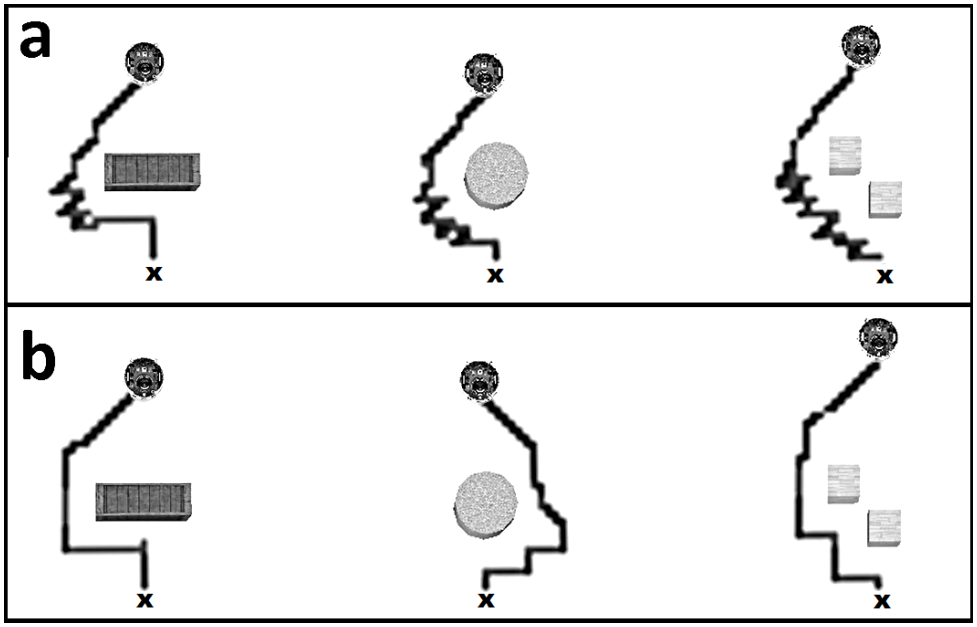

Figura 46 – Trajetória em simulação com 1 robô. Em (a) o modelo descrito em (FERREIRA, 2014), (b) modelo proposto, após as alterações nas regras e estados do AC.

Como podemos observar na Figura 46(b), o método proposto neste trabalho apresentou uma trajetória mais suave e efetiva quando comparado ao método de (FERREIRA, 2014), cuja trajetória é apresentada na Figura 46(a). Para cada cenário, dez execuções foram efetuadas e duas medidas também foram computadas: a quantidade média de rotações e o tempo médio de desvio (em segundos). As Figuras 47 e 48 apresentam graficamente os resultados médios obtidos nas simulações com cada modelo. Assim, é possível concluir que o caminho gerado pelo método proposto obteve, em média, menos rotações e diminuiu o tempo necessário para superar o obstáculo.

No cenário 1, o método proposto neste trabalho obteve uma quantidade de rotações aproximadamente 75% menor do que a obtida pelo método proposto em (FERREIRA, 2014). Neste mesmo cenário, o tempo médio para desvio foi 61% menor. No cenário 2, a quantidade de rotações realizada pelo robô utilizando nosso modelo foi aproximadamente 57% menor, enquanto o tempo para manobra de desvio completa foi aproximadamente

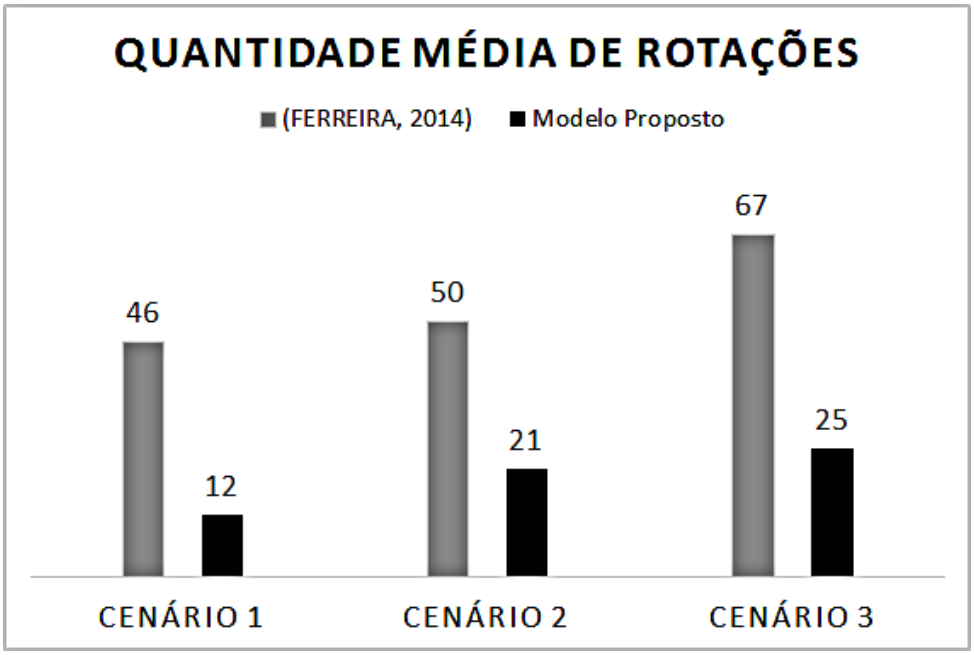

Figura 47 – Quantidade média de rotações utilizada por cada modelo para conseguir realizar o desvio completo do obstáculo e o robô retornar ao eixo original de deslocamento.

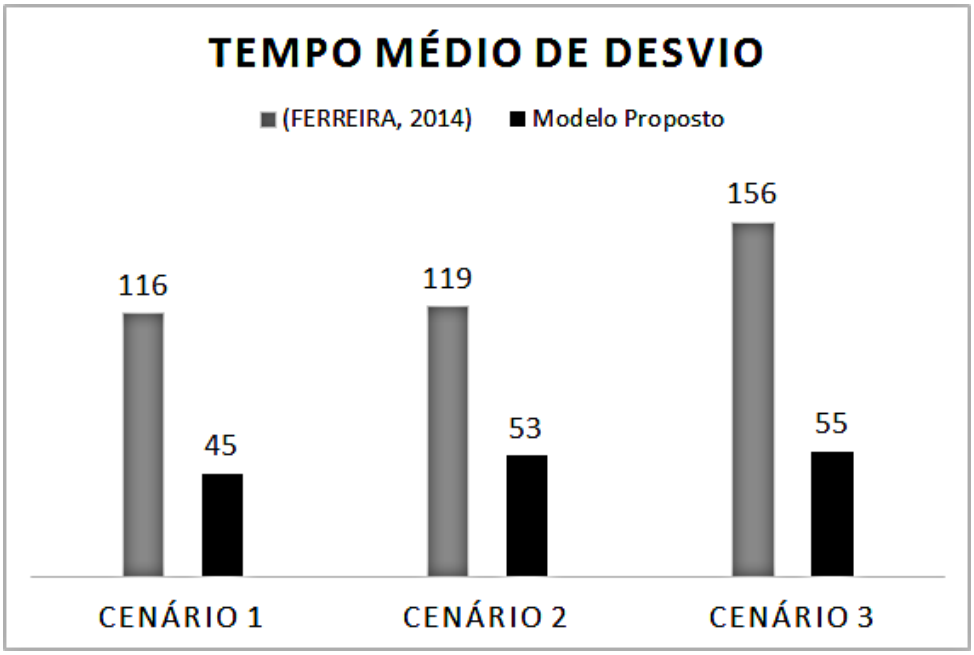

Figura 48 – Tempo médio, em segundos, necessário para que o obstáculo fosse completamente ultrapassado e o robô retornasse ao eixo original de deslocamento.

55% menor. No cenário 3, obtivemos aproximadamente 63% menos rotações e um tempo 65% menor, quando comparado ao resultado do modelo proposto em (FERREIRA, 2014).

5.1.2 Testes em cenários reais

Apesar dos experimentos em cenários reais apresentarem pequenos erros de odometria devido ao material da mesa de testes, os resultados foram bastante parecidos aos da simulação. O objetivo do robô é se deslocar sobre seu eixo de movimentação inicial até detectar o obstáculo. As regras de desvio são acionadas e o robô deve contorná-lo completamente para que possa voltar a se deslocar em seu eixo original.

A Figura 49(a) apresenta a trajetória percorrida pelos robôs utilizando o modelo de (FERREIRA, 2014) e a Figura 49(b) apresenta o resultado utilizando o nosso modelo. Nesta figura, o triângulo representa o ponto de partida, a linha indica o deslocamento do robô e cada ponto identifica o local no qual foram realizadas rotações. As trajetórias são apresentadas em três momentos do experimento, retratadas em 3 imagens: (*i*) o início da navegação, (*ii*) um instante intermediário no qual o robô já avançou aproximadamente metade da profundidade do obstáculo e (*iii*) um instante próximo ao final da navegação, no qual o robô já ultrapassou todo o obstáculo e, de acordo com as informações da odometria, o robô assume que já retornou ao seu eixo de deslocamento e orientação originais.

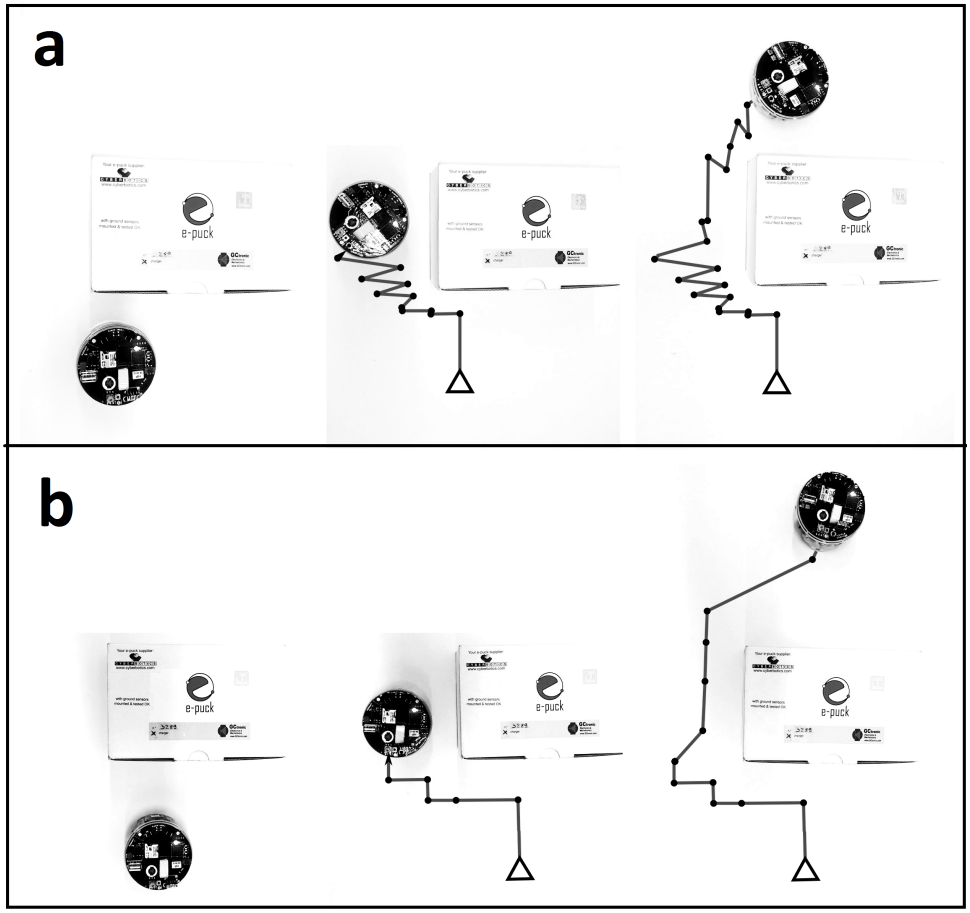

Figura 49 – Experimento com 1 robô e-puck em cenário real

Percebe-se que o método proposto demonstrou um comportamento eficiente, enquanto o apresentado em (FERREIRA, 2014) realizou um percurso com várias manobras se

afastando de seu eixo de deslocamento original. A rotação completa da roda do robô é feita em 1000 passos de revolução, a distância entre as rodas é de 52 milímetros e o raio das rodas é em torno de 21 milímetros. Com essas informações, o robô é capaz de estimar sua posição e orientação. Porém, as constantes rotações e a movimentação em zigue-zague fizeram com que o robô perdesse as referências de direcionamento utilizando o modelo proposto em (FERREIRA, 2014), como mostrado na Figura 49(a). Apesar da velocidade de movimentação ter sido a mesma para ambas as abordagens, o robô foi muito mais efetivo na tarefa de desvio de obstáculos empregando o novo modelo. O robô completou o objetivo somando, em média, 16 rotações e 80 segundos para realizar o desvio do obstáculo, no novo modelo. Utilizando o modelo proposto em (FERREIRA, 2014), foram necessárias 49 rotações e aproximadamente 300 segundos.

5.2 Experimentos com o time de robôs

Após os experimentos com um único robô, iniciamos os testes com o time de robôs, primeiramente desenvolvendo a parte de comunicação entre os e-pucks, avaliando os protocolos de comunicação Bluetooth e Wifi.

5.2.1 Experimentos preliminares com os protocolos de comunicação

A comunicação utilizando o protocolo Bluetooth requer, primeiramente, que o o robô emissor faça uma busca por dispositivos Bluetooth próximos. Após essa busca, é necessário identificar quais desses dispositivos são e-pucks, selecionando o robô receptor corretamente e estabelecendo um link através da porta serial. Os dados são enviados somente após todas essas etapas. A comunicação implementada foi testada em três fases, como mostrado na Figura 50.

Na primeira fase, foi avaliada a corretude da mensagem que chegava ao robô receptor, verificando se não havia perda de dados. Na segunda fase, foi verificada a capacidade de resposta do robô receptor, dependendo da mensagem que era recebida. Na terceira fase, houve a implementação da comunicação em time, com 2 robôs tentando se comunicar simultaneamente com o robô coordenador, que, por sua vez, deveria gerenciar corretamente a quem deveria responder. Obtivemos sucesso em todas as fases, porém, detectamos um fator que acabou tornando a comunicação Bluetooth inviável para a nossa abordagem: o tempo elevado de processamento.

Neste trabalho, optou-se por utilizar uma abordagem que maximiza a transferência do modelo, o que significa que o mesmo código utilizado na simulação é utilizado nos experimentos com robôs reais. Contudo, usando esta abordagem para a tarefa aqui investigada, a tecnologia Bluetooth foi considerada como impraticável. Para realizar a comunicação

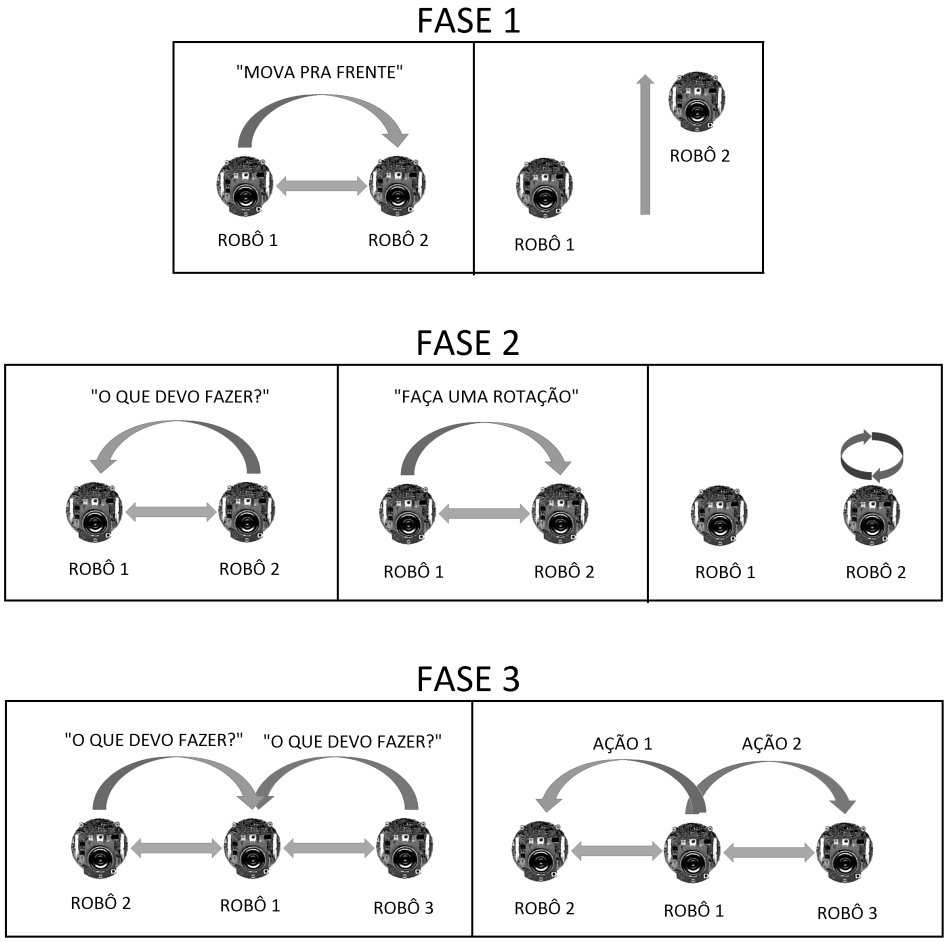

Figura 50 – Fases de teste da comunicação via Bluetooth

completa entre robôs, composta por requisição e resposta, o tempo médio necessário foi superior a 20 segundos. Esse fator também foi analisado em (COUCEIRO; VARGAS; ROCHA, 2014), chegando à conclusão que a comunicação via Bluetooth é viável somente à aplicações que requerem trocas de mensagens esporádicas. É importante ressaltar, que caso sejam utilizadas soluções dedicadas e específicas para a comunicação Bluetooth, como um algoritmo desenvolvido especialmente para lidar com a troca de mensagem nos robôs reais, o tempo de comunicação pode ser reduzido. No entanto, essa não foi a abordagem que escolhemos, pois decidimos utilizar o mesmo código da simulação, sem modificações, privilegiando a portabilidade do código.

Com o objetivo de alcançar uma comunicação robô-robô direta e rápida, nossa segunda opção foi usar uma placa de extensão na parte superior do robô, chamada Gumstix Overo COM, que fornece a capacidade de comunicação WiFi. Em relação à abordagem de comunicação, foi utilizada uma estratégia semelhante para a tecnologia Bluetooth, ou seja, o robô emissor envia uma mensagem direta para outro robô e-puck. Assim, a estrutura das mensagens foi mantida e a comunicação foi testada usando a mesma abordagem anterior.

Nessa arquitetura, a comunicação é composta por várias camadas. No envio, a primeira

etapa ocorre nos e-pucks e consiste na criação do pacote contendo as informações a serem enviadas. Esse pacote é então enviado por comunicação serial (R-232) para a placa Overo. A placa, já conectada na rede Wifi, faz a ponte entre os e-pucks e o servidor (PC) e repassa a mensagem. O servidor redireciona a mensagem até o devido receptor, o pacote chega à placa Overo do destinatário e, então, é transferida novamente via comunicação serial para o e-puck. Utilizando esse protocolo, as mensagens são trocadas instantaneamente (o tempo de redirecionamento gasto pelo servidor é mínimo), permitindo que o método seja executado no robô real de forma semelhante à simulação.

Os mesmos cenários de teste foram aplicados usando a comunicação Wifi entre os robôs e-puck reais. Como resultado, as mensagens trocadas foram todas recebidas com sucesso, com um tempo médio de menos de 1s. Embora a utilização da placa Gumstix Overo COM resultou em um aumento no custo financeiro de cada e-puck, fomos capazes de finalmente realizar a comunicação desejada e aplicar o método proposto em experimentos em time. Após esses experimentos, definimos que o protocolo de comunicação implementado em nosso método seria o Wifi.

5.2.2 Testes em simulação

Como existem três robôs e-puck físicos em nosso laboratório, criamos ambientes em simulação também com um grupo de três robôs. O objetivo do time foi manter a formação, enquanto explorava um ambiente com obstáculos. Nos experimentos descritos nessa seção, não foi empregada a estratégia de reconfiguração da formação evoluídas a partir de um método evolutivo como descrito na seção 4.3. Nesse caso, foi empregada a mesma estratégia adotada em (FERREIRA, 2014), ou seja, não existe a troca de eixos de referência no caso de aproximação entre robôs do time. A Figura 51 apresenta a formação inicial do time de robôs, definida como um padrão espacial em linha, na qual os robôs inicialmente alinhados e separados por uma distância de 25 centímetros. Além disso , a Figura 51 apresenta os obstáculos utilizados no cenário de teste; utilizamos obstáculos no formato cilíndrico e retangular.

No ambiente de simulação, não são levados em consideração o atraso provocado pelas trocas de mensagens. Portanto, a comunicação entre robôs foi realizada sem erros e a formação foi mantida durante a maior parte do percurso. Podemos perceber pela Figura 52, que os robôs apresentaram um comportamento eficiente tanto individualmente quanto em grupo. A figura apresenta 4 capturas de tela do simulador Webots em diferentes momentos da navegação: $\mathbf{t}$=0s, $\mathbf{t}$=15s, $\mathbf{t}$=30s e $\mathbf{t}$=45s.

No modo reativo (individual), o robô analisa a sua vizinhança e quando um obstáculo potencial é detectado, a regra apropriada do AC é selecionada para que o desvio seja realizado. Neste experimento, podemos perceber que todos os robôs tomaram a decisão de contornar o obstáculo pelo lado esquerdo. Entretanto, essa não é uma ação arbitrária, de forma que eles poderiam ter contornado pela direita. A manobra de desvio do obstáculo

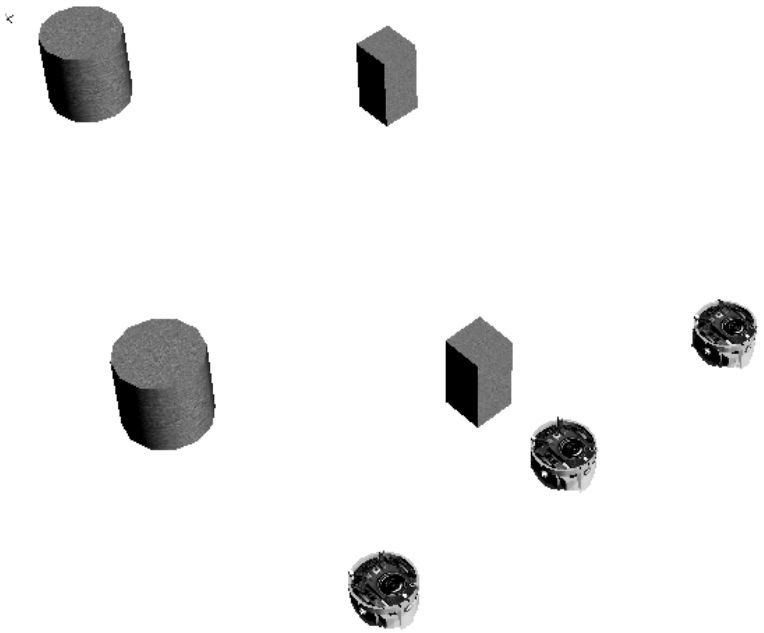

Figura 51 – Cenário de testes com o time de robôs em simulação.

depende dos valores obtidos pelos sensores. Caso o resultado da leitura apresente maiores valores nos sensores do lado direito, o robô rotacionará para esquerda; caso contrário, ele rotacionará para a direita.

Os robôs devem procurar manter a sua navegação através de seu eixo inicial de deslocamento durante todo o trajeto. Porém, para que a formação seja preservada, todos os robôs devem estar alinhados durante o percurso (para a formação em linha). Podemos observar na Figura 52 que, apesar das inúmeras manobras de desvio de obstáculo, os robôs são capazes de manter a formação linear na maior parte do percurso.

5.2.3 Testes em cenários reais

O ambiente físico utilizado no laboratório, apresentado na Figura 53, foi uma mesa de 60 cm largura, com 120 cm de comprimento e uma borda de 10 cm de altura. Utilizamos pequenos objetos (como caixas, livros e esferas), colocados no cenário, a fim de criar os obstáculos. A incidência de luz era estável, estabelecendo boas condições experimentais ao se testar diferentes comportamentos separadamente.

Os robôs e-puck possuem um seletor que permite a escolha de diferentes comportamentos quando o robô é ligado. Utilizamos o seletor na posição 10 para ativar a comunicação Wifi. Desta forma, ao ligar o robô e-puck, ele verifica internamente qual sua posição na

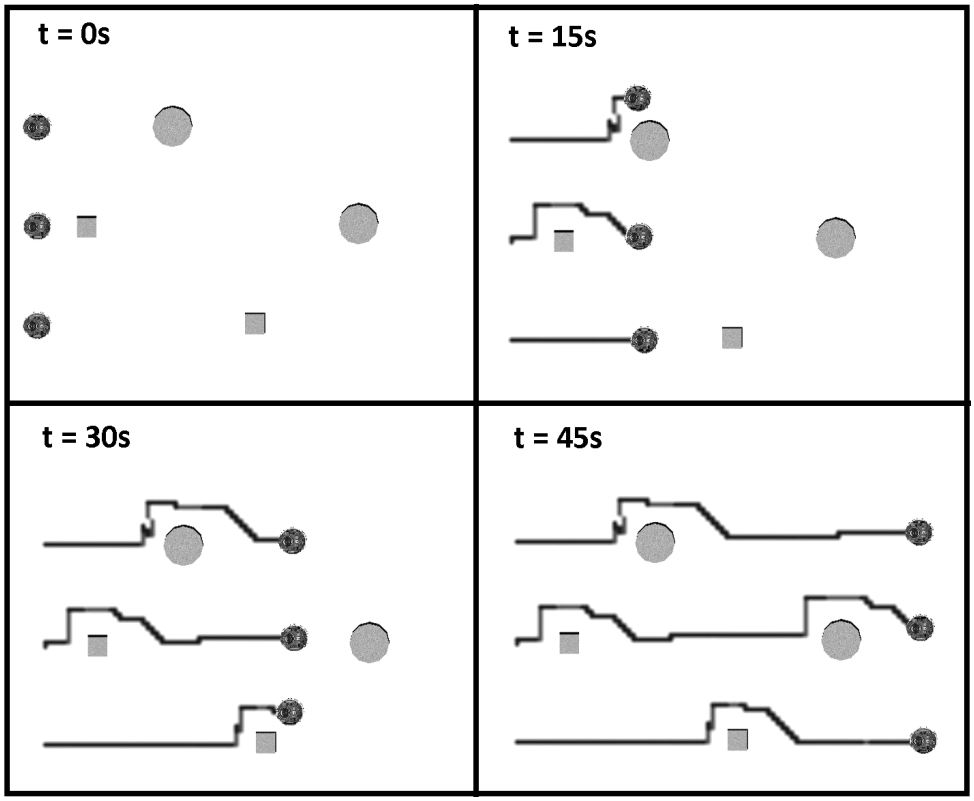

Figura 52 – Experimento de navegação em time no simulador Webots em 4 instantes de tempo: 0, 13, 30 e 45 segundos.

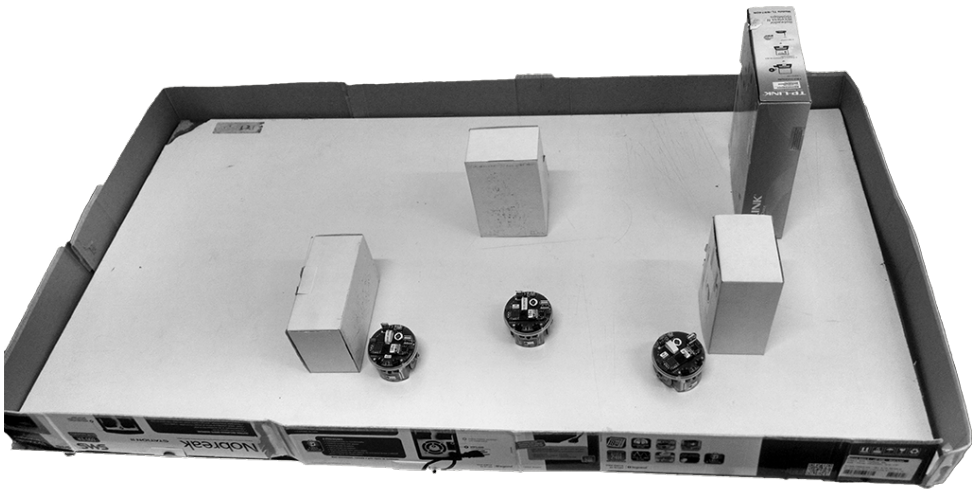

Figura 53 – Cenário de testes com o time de robôs e-puck.

formação e utiliza a placa Overo para se conectar à rede Wifi previamente configurada. Após a conexão de todos os e-pucks na rede, um computador externo conectado na mesma rede detecta as mensagens enviadas pelos robôs e começa a agir como um distribuidor de mensagens. Usando trocas de mensagens, se um robô está atrasado na formação ou à frente de sua posição, os outros robôs podem ou esperar ou se mover a fim de recuperar o padrão espacial do time. A Figura 54 mostra a trajetória (da esquerda para a direita)

realizada pelo time de e-pucks em um experimento envolvendo um cenário similar ao da Figura 53. Nesta figura, são apresentados 6 imagens capturadas do experimento em diferentes instantes de tempo: **t**=0, 15, 30, 45, 60 e 75 segundos.

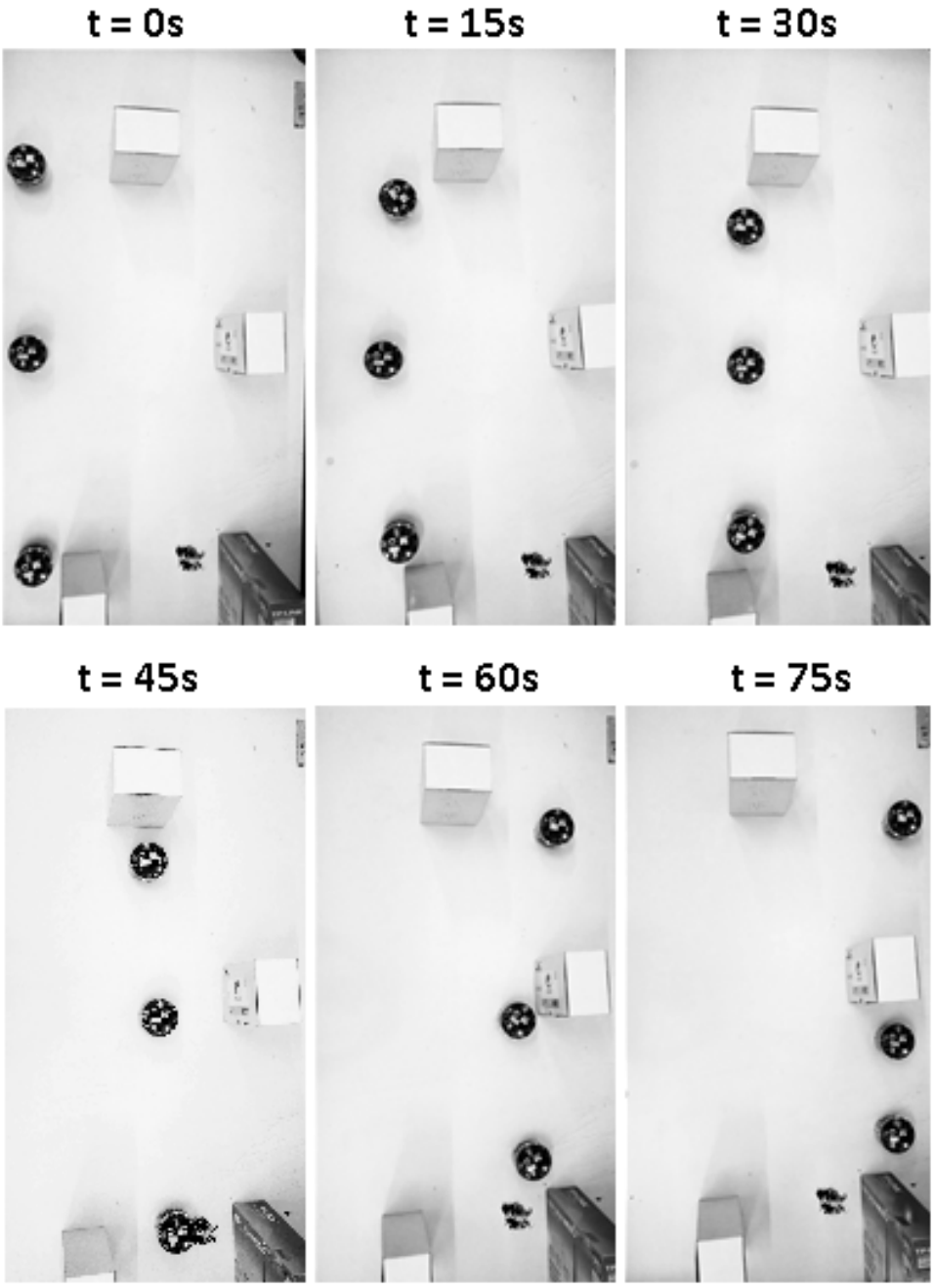

Figura 54 – Experimento em time com robôs reais.

No tempo $t = 0$s, os robôs estão alinhados em formação e já conectados na rede Wifi. Em $t = 15$s, os robôs na esquerda e na direita começam a manobra de desvio dos obstáculos. A partir de $t = 30$s até $t = 45$s, os robôs continuam a se mover em direção à meta mantendo o padrão espacial linear. No tempo $t = 60$s, o robô da esquerda volta para seu caminho original, o robô coordenador e o robô do lado direito começam a se

desviar de outros obstáculos. Em $t = 75$s todos os robôs chegam no objetivo final. O vídeo de todo o experimento está disponível em http://goo.gl/Cjz91o.

Devido ao custo e esforço necessários para executar os experimentos utilizando sistemas reais com múltiplos robôs, os testes em cenários reais geralmente apresentam grandes inconformidades com os resultados em simulação. O problema potencial é que muitas vezes as abstrações da simulação não incorporam muitos fatores que podem estar presentes em um cenário do mundo real. Os exemplos podem incluir ruído do sensor, controle de luminosidade do ambiente, deslizamento da roda, e capacidade de comunicação. Isto é especialmente verdadeiro para algoritmos altamente coordenados, em que a complexidade gera mais oportunidades para que os comportamentos observados divirjam dos esperados.

Nossa abordagem discreta baseada em ACs permitiu que o desempenho em cenários reais, como mostrado na Figura 54, fosse bastante semelhante aos obtidos em simulação. Ao final, os robôs foram capazes de agir individualmente quanto aos desvios de obstáculos e, em grupo, mantendo a formação durante todo o percurso.

É importante ressaltar que no trabalho de (FERREIRA, 2014) não foram realizados testes envolvendo times de robôs em cenários reais.

5.3 Experimentos com a reconfiguração da formação

Nos experimentos evolutivos para obtenção da estratégia de reconfiguração da formação, descrita na subseção 4.3.3, em cada geração do AG, a simulação foi realizada em cinco cenários fixos diferentes para a avaliação de cada indivíduo. Desta forma, a população é avaliada de forma menos tendenciosa (do que se tivesse sido utilizado apenas 1 cenário) e se torna mais robusta quanto a mudanças no ambiente. A Figura 55 apresenta os cinco cenários utilizados nos experimentos aqui relatados.

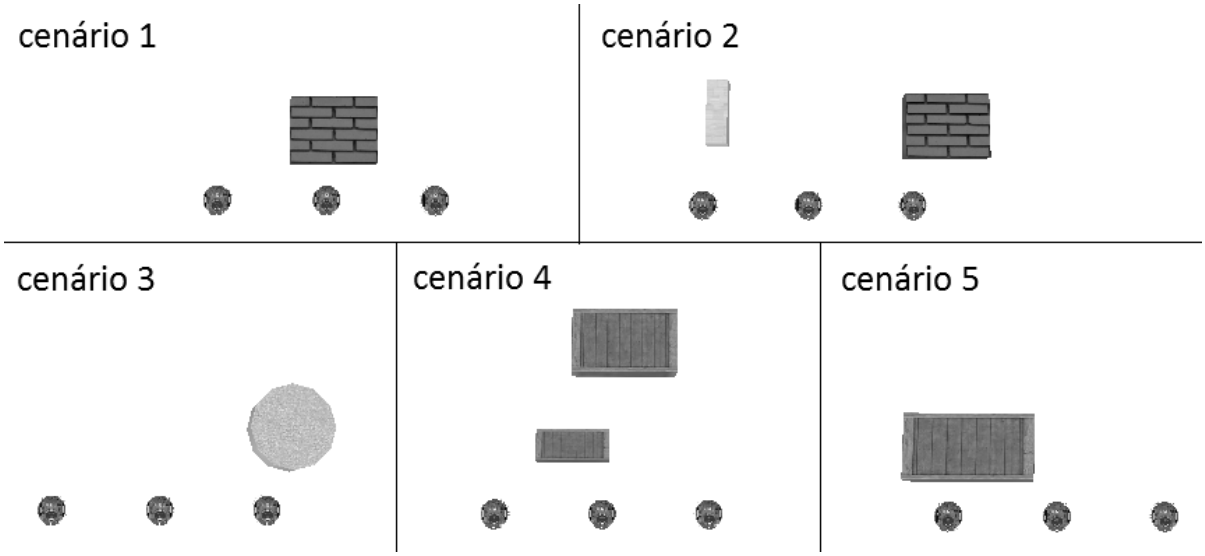

Figura 55 – Cenários utilizados durante o processo evolutivo do AG.

Cada cenário foi criado buscando explorar as características individuais e cooperativas dos robôs, de forma que pudéssemos avaliar a navegação do time pelo ambiente. A função de avaliação utilizada em todos os cenários é dada pela Equação 3, que leva em consideração o tempo em formação, a distância percorrida pelo time e a quantidade de rotações. Foram realizadas execuções do AG utilizando oito configurações de execução diferentes. Os parâmetros utilizados foram:

- **Número de indivíduos da população**: 10 ou 20;
- **Gerações**: 10;
- **Seleção**: roleta;
- **Função de avaliação**: as quatro combinações de pesos utilizadas na Equação 3 foram [1,1,1],[8,1,1],[1,8,1] e [1,1,8]. O primeiro valor refere-se peso do fator tempo em formação, o segundo refere-se ao peso aplicado à distância percorrida pelos robôs e o terceiro refere-se ao peso dado para a quantidade de rotações realizadas;
- **Recombinação**: *crossover* simples com taxa de 100%;
- **Taxa de mutação**: 10%;
- **Reprodução**: Sobrevivência dos melhores entre pais e filhos.

Desta forma, conseguimos analisar o comportamento da formação quando priorizamos diferentes fatores. Com o intuito de obter um intervalo de confiança nos resultados, também utilizamos três sementes para o AG. Ou seja, as oito configurações foram executadas três vezes, utilizando sementes diferentes.

Para calcular a aptidão, o AG executa a simulação nos cinco cenários para cada indivíduo da população, a cada geração. Ao inciar o Webots, o programa primeiramente realiza uma validação de usuário utilizando a internet, o que pode acarretar em um atraso no início da simulação. Para reduzir esse tempo, transferimos a licença para um *dongle* usb, conseguindo minimizar também os erros de autenticação. Como o tempo de execução para cada cenário foi de dois minutos, ao todo esses testes demandaram aproximadamente 25 dias. A Tabela 7 apresenta os resultados obtidos após todas as execuções.

Podemos perceber que a variação entre os valores de aptidão obtidos é pequena, ao avaliarmos o mesmo experimento nas 3 execuções do AG. Desta forma, consideramos que, embora o uso de apenas três sementes diferentes tenha sido possível devido ao elevado tempo de processamento, o AG não apresentou grandes oscilações entre as execuções. Os indivíduos gerados pelo AG apresentaram comportamentos diversificados em cada cenário, demonstrando que o comportamento em time pode ser diferente dependendo da estratégia de reconfiguração da formação. Na Figura 56, podemos observar que o time de robôs adotou trajetórias diferentes para o mesmo cenário. Na Figura 56a, a estratégia utilizada

Tabela 7 – Resultados das execuções do AG

		semente 1		semente 2		semente 3	
pesos	individuos	maior fitness	fitness média	maior fitness	fitness média	maior fitness	fitness média
iguais	10	0,800	0,800	0,800	0,800	0,800	0,800
	20	0,900	0,805	0,900	0,805	0,800	0,800
formação	10	0,800	0,800	0,900	0,900	0,800	0,800
	20	0,900	0,810	0,800	0,745	0,800	0,800
rotação	10	0,800	0,800	0,800	0,730	0,700	0,700
	20	0,800	0,800	0,800	0,710	0,700	0,700
distância	10	1,000	1,000	1,000	1,000	0,900	0,900
	20	1,000	1,000	1,000	0,915	0,900	0,900

foi a de se manter no mesmo eixo de deslocamento (similar ao modelo em (FERREIRA, 2014)), enquanto, na Figura 56b, os robôs decidiram reconfigurar a formação, realizando uma troca de posições. A estratégia evoluída utilizada em (a) foi [15 11 232 224]. Ou seja, ela utiliza um fator de aproximação 11 (indicando que apenas se os robôs se aproximarem em 11 células haverá a troca), um fator de avanço 15 (indicando que o robô que não participar da reconfiguração de posições deve acompanhar o time durante a manobra, realizando 15 passos), uma velocidade de deslocamento de 232 rad/s e uma velocidade de rotação de 170 rad/s. A estratégia evoluída utilizada em (b) foi [0 0 161 155].

No cenário 4, o AG obteve indivíduos que apresentaram comportamentos bem variados. Como podemos observar na Figura 57, quatro trajetórias diferentes foram geradas. As estratégias utilizadas em cada figura são: (a) [7 6 199 199], (b) [0 10 144 244], (c) [18 0 197 142], (d) [0 25 158 169]. Na Figura 57a podemos perceber uma abordagem híbrida: (*i*) o robô coordenador desvia do primeiro obstáculo, contornando-o pela direita, sem ativar uma manobra de troca de posição com outros robôs, (*ii*) no segundo obstáculo encontrado pelo robô coordenador foi necessária uma maior movimentação se distanciando de seu próprio eixo para realizar o desvio do obstáculo; esse deslocamento maior fez com que a reconfiguração da formação fosse considerada uma melhor escolha do que apenas retornar ao seu eixo original, (*iii*) a partir da comunicação entre os robôs, eles decidem realizar a troca de posições, fazendo com que o robô da esquerda se direcionasse para o eixo central e se tornasse o novo coordenador e o antigo robô coordenador assumisse o lugar na extremidade esquerda da formação. A Figura 57b apresenta uma abordagem similar, porém o robô coordenador decidiu contornar o segundo obstáculo pela direita, o

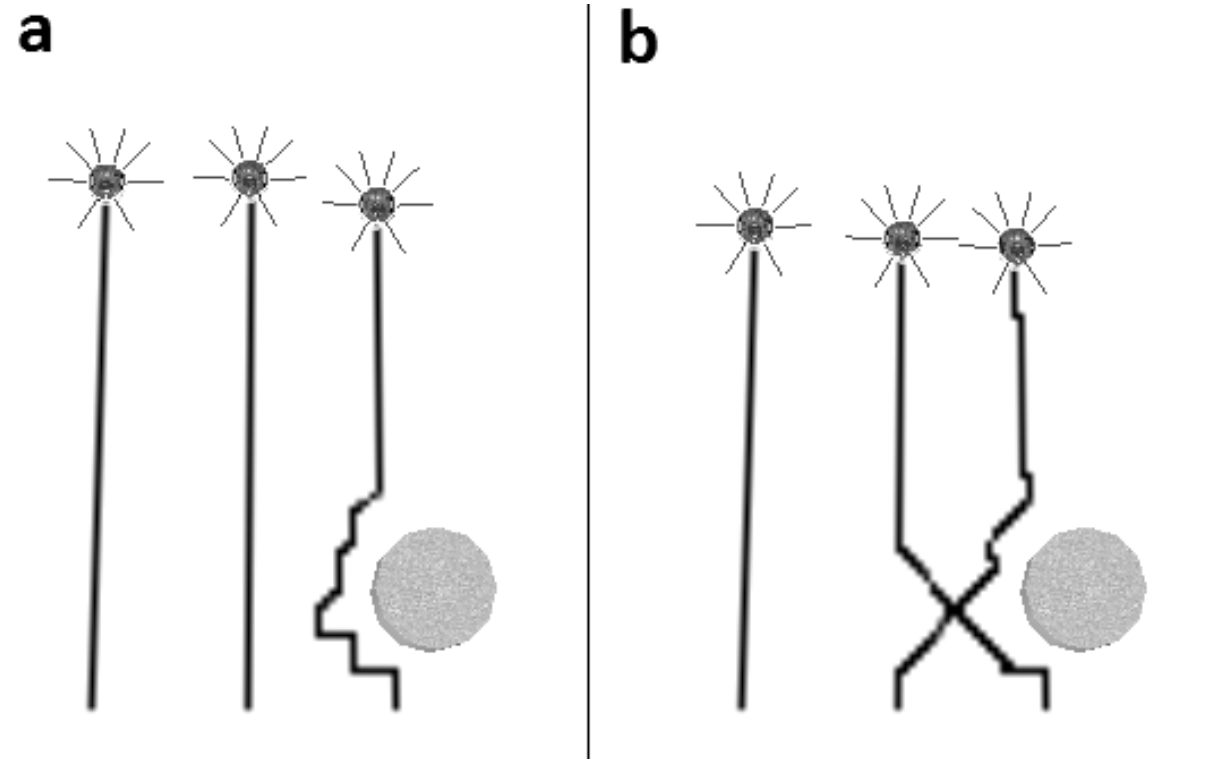

Figura 56 – Diferentes comportamentos do time de robôs para o cenário 3. Em (a) a estratégia utilizada foi [15 11 232 224]. Em (b) a estratégia utilizada foi [0 0 161 155].

que acabou gerando a troca de posições com o robô da extremidade direita da formação. As estratégias fixas são apresentadas na Figura 57c, no qual a troca de posições é ativada logo no primeiro desvio de obstáculo, e na Figura 57d, onde os robôs não utilizam a reconfiguração da formação baseada em troca de posições, simplesmente retornando para seu eixo original caso haja algum desvio de um obstáculo.

A partir de todos os indivíduos gerados pelo AG, selecionamos o melhor indivíduo na população final de cada execução (para cada configuração de teste e para cada uma das três sementes), somando no total vinte e quatro indivíduos selecionados. Com o intuito de testar esses indivíduos em cenários desconhecidos (diferentes dos utilizados no processo de evolução do AG), criamos mais vinte cenários de teste em simulação e realizamos o experimento com cada indivíduo. Todos os vinte cenários podem ser visualizados no Anexo A.

Realizamos o teste dos indivíduos em dez execuções para cada cenário, utilizando a função de aptidão com pesos iguais para os três fatores considerados em seu cálculo. Desta forma, uma melhor avaliação representa que a estratégia evoluída obteve esse resultado balanceando a maximização do tempo em formação e da distância percorrida com a minimização do número de rotações. A Tabela 8 apresenta o resultados dos experimentos para os indivíduos selecionados na primeira semente do AG.

Os indivíduos estão identificados por seus campos, o primeiro representa o valor utili-

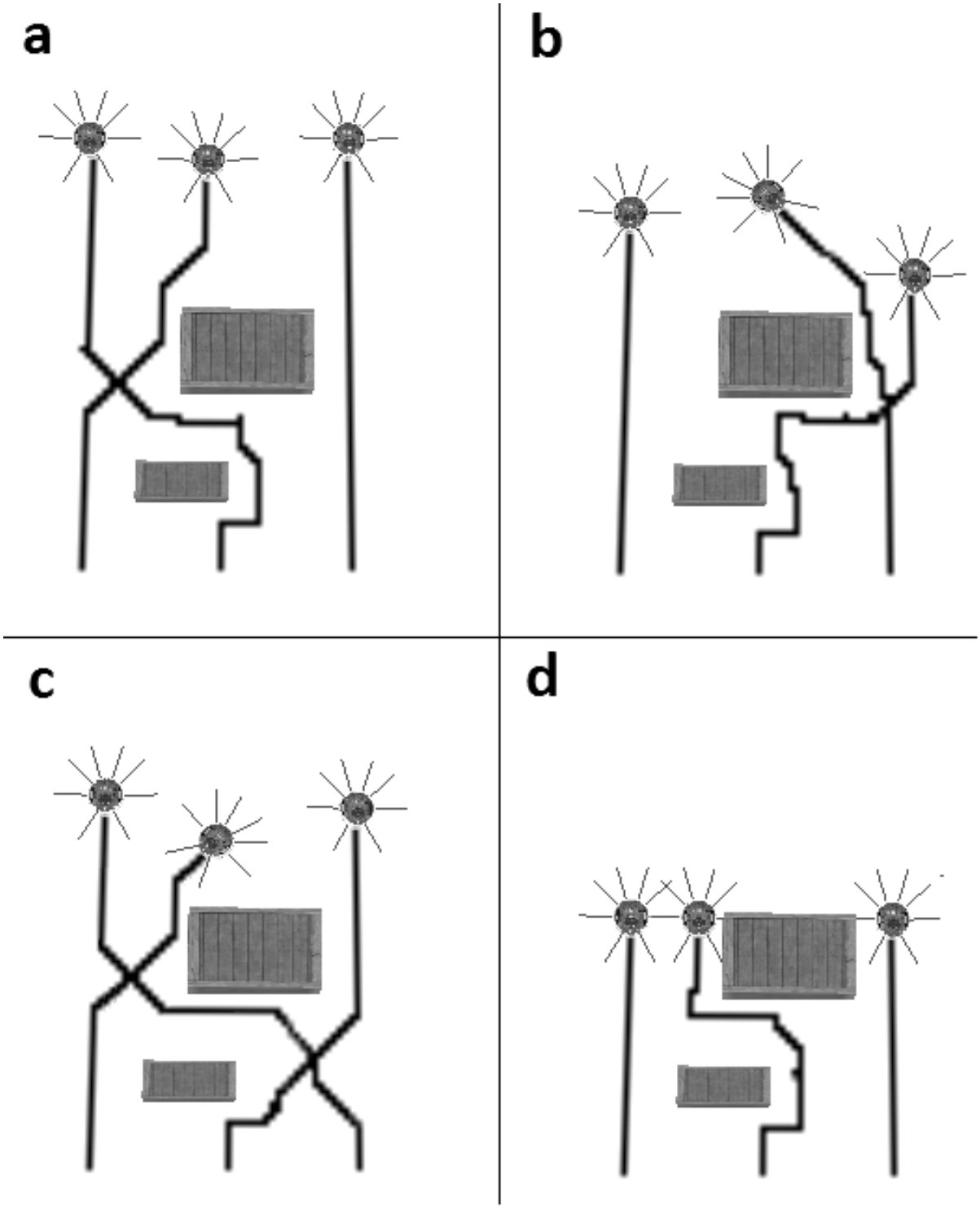

Figura 57 – Diferentes comportamentos do time de robôs para o cenário 4: (a) [7 6 199 199], (b) [0 10 144 244], (c) [18 0 197 142], (d) [0 25 158 169].

zado para o fator de avanço, o segundo é o valor para o fator de aproximação, o terceiro representa a velocidade de movimentação em frente e o quarto se refere à velocidade de rotação do robô. Podemos observar que, dentre os indivíduos evoluídos na semente 1, a estratégia que obteve a melhor avaliação foi [9 0 181 124]. Na Tabela 9 e na Tabela 10, são apresentados os resultados para os indivíduos da semente 2 e da semente 3, respec-

Tabela 8 – Resultados para os indivíduos da semente 1

	Indivíduos							
Cenários	3 0 182 214	**9 0 181 124**	10 9 151 291	10 11 182 214	13 0 186 170	19 2 141 227	19 3 142 227	20 0 180 124
1	0,8	**0,9**	0,8	0,9	0,9	0,7	0,8	0,8
2	0,8	**0,9**	0,8	0,8	0,8	0,8	0,8	0,9
3	0,5	**0,5**	0,6	0,5	0,4	0,5	0,5	0,5
4	0,5	**0,6**	0,5	0,5	0,5	0,6	0,5	0,5
5	0,7	**0,6**	0,7	0,6	0,7	0,7	0,6	0,7
6	0,4	**0,6**	0,6	0,6	0,6	0,5	0,6	0,6
7	0,7	**0,7**	0,6	0,6	0,7	0,7	0,6	0,7
8	0,4	**0,5**	0,5	0,6	0,5	0,4	0,5	0,5
9	0,6	**0,6**	0,6	0,7	0,6	0,5	0,7	0,6
10	0,7	**0,8**	0,6	0,7	0,6	0,6	0,7	0,8
11	0,4	**0,5**	0,6	0,5	0,5	0,6	0,5	0,5
12	0,6	**0,6**	0,5	0,5	0,4	0,6	0,6	0,6
13	0,5	**0,7**	0,5	0,5	0,5	0,6	0,5	0,5
14	0,6	**0,8**	0,7	0,7	0,7	0,8	0,6	0,8
15	0,6	**0,8**	0,3	0,5	0,6	0,5	0,6	0,6
16	0,5	**0,6**	0,6	0,5	0,6	0,5	0,6	0,6
17	0,7	**0,7**	0,7	0,5	0,4	0,7	0,7	0,6
18	0,6	**0,4**	0,6	0,6	0,5	0,5	0,5	0,6
19	0,7	**0,8**	0,5	0,5	0,7	0,6	0,6	0,7
20	0,7	**0,8**	0,7	0,7	0,7	0,6	0,6	0,6
Média Geométrica	0,587	**0,655**	0,588	0,590	0,580	0,591	0,598	0,625
Média Aritmética	0,600	**0,670**	0,600	0,600	0,595	0,600	0,605	0,635

tivamente. O indivíduo [0 15 249 141] foi que apresentou a melhor avaliação dentre os escolhidos para a semente 2. Quanto à semente 3 o melhor indivíduo foi [6 2 161 170].

Com o objetivo de comparar esses indivíduos gerados pelo AG com as abordagens fixas utilizadas em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e (FERREIRA, 2014), realizamos também o teste nos vinte cenários com dois indivíduos representando essas estratégias. Em ambas as estratégias adotamos o valor máximo utilizado em nossos experimentos para o fator de avanço, ou seja, 20 passos. Para as velocidades de deslocamento e rotação, aplicamos os mesmos valores utilizados em (FERREIRA, 2014), ou seja, 200 e 150 rad/s, respectivamente. Para a estratégia que sempre realiza as trocas de eixo foi definido o fator de aproximação 0, enquanto que, para a estratégia que nunca realiza as trocas foi definido um fator de aproximação 30 (maior que a distância entre os robôs). A Tabela 11 apresenta o resultado dos testes para esses indivíduos.

Analisando o resultado de todas as tabelas, percebemos que o indivíduo que apresentou a maior avaliação média para todos os cenários foi o indivíduo [0 15 249 141]. Esse indivíduo representa uma abordagem híbrida, na qual o time só realiza a reconfiguração da formação com uma manobra de troca de posições se a distância percorrida por algum robô se aproximando do eixo de deslocamento de outro for superior a quinze centímetros.

Tabela 9 – Resultados para os indivíduos da semente 2

	Indivíduos							
Cenários	**0 15 249 141**	13 1 284 207	19 1 288 104	19 2 180 173	19 6 176 135	19 9 177 135	12 13 199 199	7 6 199 199
1	**0,9**	0,8	0,9	0,8	0,8	0,8	0,9	0,9
2	**0,9**	0,8	0,9	0,9	0,8	0,9	0,8	0,9
3	**0,5**	0,4	0,4	0,5	0,5	0,4	0,5	0,5
4	**0,6**	0,5	0,5	0,4	0,4	0,5	0,4	0,4
5	**0,6**	0,6	0,7	0,7	0,6	0,5	0,5	0,7
6	**0,6**	0,5	0,6	0,5	0,6	0,6	0,4	0,6
7	**0,7**	0,6	0,6	0,7	0,7	0,6	0,6	0,7
8	**0,5**	0,6	0,4	0,5	0,5	0,5	0,5	0,5
9	**0,6**	0,6	0,6	0,6	0,6	0,7	0,5	0,6
10	**0,8**	0,7	0,8	0,8	0,5	0,6	0,5	0,8
11	**0,7**	0,5	0,4	0,6	0,6	0,7	0,7	0,4
12	**0,7**	0,7	0,8	0,6	0,4	0,6	0,5	0,5
13	**0,5**	0,5	0,5	0,5	0,6	0,5	0,5	0,5
14	**0,7**	0,5	0,7	0,6	0,7	0,6	0,6	0,7
15	**0,6**	0,5	0,7	0,6	0,6	0,6	0,5	0,5
16	**0,7**	0,6	0,5	0,5	0,6	0,4	0,8	0,6
17	**0,7**	0,6	0,3	0,7	0,7	0,7	0,7	0,5
18	**0,7**	0,4	0,4	0,5	0,5	0,6	0,6	0,6
19	**0,8**	0,7	0,8	0,7	0,6	0,6	0,5	0,5
20	**0,7**	0,6	0,7	0,7	0,5	0,5	0,6	0,5
Média Geométrica	**0,666**	0,575	0,583	0,608	0,580	0,583	0,566	0,579
Média Aritmética	**0,675**	0,585	0,610	0,620	0,590	0,595	0,580	0,595

Como utilizamos uma distância inicial entre robôs de vinte e cinco centímetros, pode-se considerar que o fator de aproximação híbrido utiliza as vantagens de ambas as estratégias fixas, de modo a realizar uma troca de eixos em um momento otimizado.

Esse comportamento foi identificado também em análises qualitativas da trajetória. Para isso, utilizamos cenários ainda mais complexos que os apresentados no Anexo A, envolvendo um número maior de obstáculos. A Figura 58 apresenta um exemplo de trajetórias com comportamentos diferentes dependendo do cenário. Utilizando os parâmetros do indivíduo [0 15 249 141], o time apresentou um comportamento adaptativo. No cenário A, a estratégia de troca de eixos entre robôs não foi utilizada em nenhum momento da navegação do time, pois embora tenham ocorrido desvios, os robôs não se deslocaram significativamente dos seus eixos. No cenário B, devido à presença de um obstáculo que demandou uma aproximação maior entre robôs durante o desvio, a reconfiguração da formação foi ativada, fazendo com que o robô coordenador trocasse de posição com seu robô vizinho direito. Desta forma, o time de robôs conseguiu extrair o melhor comportamento para cada cenário e manter a formação durante mais tempo.

Infelizmente, apesar dos resultados promissores obtidos em simulação, não foi possível avaliar o desempenho do novo modelo com estratégia adaptativa para reconfiguração

Tabela 10 – Resultados para os indivíduos da semente 3

	Indivíduos							
Cenários	1 13 232 296	4 7 232 185	4 12 232 296	5 1 213 141	**6 2 161 170**	6 2 195 263	15 13 232 296	16 2 161 170
1	0,8	0,8	0,8	0,8	**0,8**	0,7	0,8	0,8
2	0,8	0,8	0,8	0,9	**0,8**	0,8	0,8	0,8
3	0,5	0,4	0,4	0,5	**0,5**	0,4	0,5	0,5
4	0,5	0,7	0,5	0,6	**0,6**	0,6	0,5	0,6
5	0,6	0,6	0,6	0,5	**0,6**	0,6	0,5	0,6
6	0,4	0,6	0,5	0,4	**0,6**	0,5	0,4	0,6
7	0,6	0,6	0,6	0,6	**0,7**	0,6	0,6	0,7
8	0,6	0,6	0,5	0,5	**0,5**	0,4	0,6	0,4
9	0,7	0,6	0,7	0,6	**0,6**	0,6	0,7	0,6
10	0,6	0,5	0,6	0,5	**0,6**	0,4	0,6	0,6
11	0,7	0,6	0,7	0,5	**0,6**	0,6	0,7	0,6
12	0,4	0,6	0,4	0,7	**0,6**	0,4	0,4	0,5
13	0,5	0,5	0,5	0,6	**0,5**	0,5	0,5	0,5
14	0,7	0,6	0,7	0,6	**0,6**	0,8	0,7	0,6
15	0,4	0,6	0,4	0,7	**0,7**	0,5	0,5	0,7
16	0,7	0,6	0,6	0,6	**0,5**	0,5	0,6	0,6
17	0,7	0,6	0,7	0,5	**0,7**	0,5	0,7	0,7
18	0,5	0,5	0,5	0,5	**0,5**	0,5	0,5	0,5
19	0,7	0,7	0,7	0,7	**0,7**	0,6	0,7	0,7
20	0,7	0,7	0,7	0,6	**0,6**	0,5	0,7	0,6
Média Geométrica	0,591	0,603	0,581	0,584	**0,608**	0,538	0,588	0,602
Média Aritmética	0,605	0,610	0,595	0,595	**0,615**	0,550	0,600	0,610

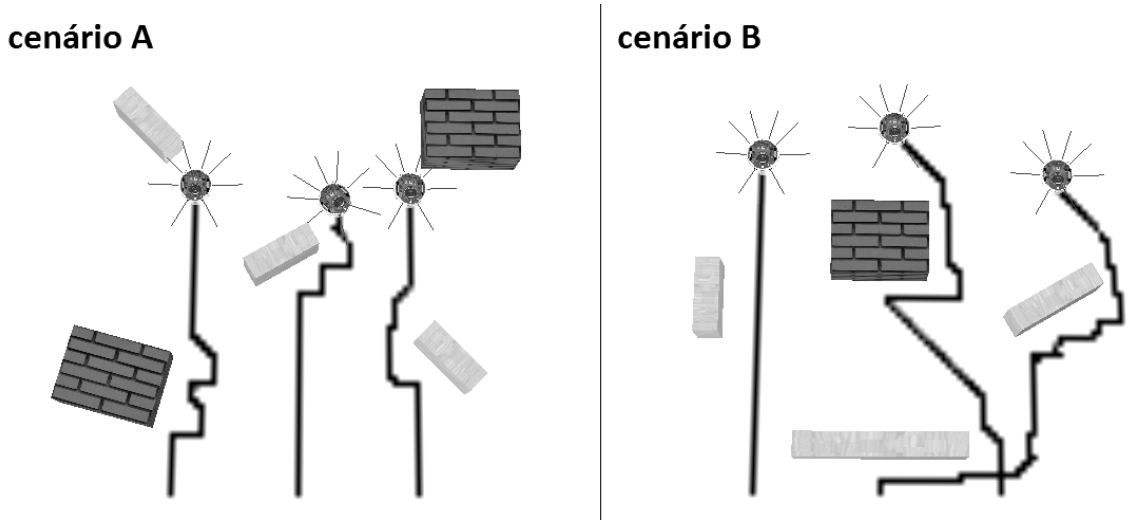

Figura 58 – Robôs apresentando comportamento adaptativo dependendo do cenário.

da formação em experimentos com robôs reais. Apos a execução dos experimentos relatados na subseção 5.2.3, os robôs e-puck existentes no laboratório de computação Bioinspirada da FACOM/UFU apresentaram uma série de problemas técnicos que impediram

Tabela 11 – Resultados para os indivíduos com estratégias fixas

	Indivíduos	
Cenários	20 0 200 150	20 30 200 150
1	0,8	0,9
2	0,8	0,9
3	0,4	0,4
4	0,6	0,5
5	0,6	0,4
6	0,4	0,4
7	0,7	0,6
8	0,4	0,6
9	0,6	0,7
10	0,6	0,6
11	0,6	0,5
12	0,6	0,4
13	0,5	0,6
14	0,6	0,8
15	0,6	0,6
16	0,6	0,6
17	0,4	0,7
18	0,5	0,5
19	0,7	0,7
20	0,4	0,7
Média Geométrica	0,556	0,587
Média Aritmética	0,570	0,605

a execução de novos experimentos. Para que esses experimentos possam ser realizados, é necessária a aquisição de um novo equipamento, inexistente hoje no laboratório: um depurador/programador MPLAB ICD3 para reprogramar o *firmware* dos e-pucks.

Entretanto, embora tais experimentos não tenham sido realizados, temos grande expectativa que os resultados em cenários reais serão bem próximos aos virtuais. A observação das similaridades dos experimentos virtuais e reais descritos na seção 5.1 e na seção 5.2 corrobora essa expectativa.

Capítulo 6

Conclusão

Por meio do desenvolvimento e aplicação de um novo método evolutivo-cooperativo na tarefa de planejamento de trajetórias e controle de formação, foi possível observar um bom desempenho do sistema de navegação, tanto na trajetória individual, quanto considerando-se um time de robôs. Além disso, o novo modelo apresentou melhorias quando comparado aos modelos anteriores discutidos em (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c) e (FERREIRA, 2014), que também são baseados em regras de autômatos celulares e que serviram de base para a nova proposição. A seguir, são descritas as considerações finais sobre nossa contribuição e possibilidades de trabalhos futuros.

6.1 Principais Contribuições

Em nosso trabalho, um novo método para planejamento de caminhos e controle de formação foi desenvolvido. Inicialmente, ressaltamos as vantagens da utilização de ACs nessa tarefa e analisamos diversos trabalhos relacionados ao problema de navegação de um time de robôs com controle de formação. A partir dessas análises encontramos alguns pontos fracos nas abordagens presentes na literatura (IOANNIDIS; SIRAKOULIS; ANDREADIS, 2011c), (FERREIRA; VARGAS; OLIVEIRA, 2014), (OLIVEIRA; VARGAS; FERREIRA, 2015) e propusemos novas abordagens para superá-las.

Primeiramente, abordamos as estratégias utilizadas pelos robôs de forma individual, sem levar em consideração a característica cooperativa. Nessa fase, criamos 5 novos estados para o modelo do AC e desenvolvemos regras mais sofisticadas para o desvio de obstáculos. Conforme os testes em simulação e em cenários reais demonstraram, o novo método obteve, em média, 64% menos rotações e diminuiu em 61% o tempo necessário para superar um obstáculo, quando comparado ao modelo proposto em (FERREIRA, 2014). Também otimizamos o modelo por meio de uma abordagem evolucionária na classificação da vizinhança, que possibilitou uma melhor definição dos estados das células vizinhas e consequente maior precisão na tomada de decisões por parte dos robôs.

Quanto ao modelo em time, implementamos uma estrutura de comunicação, que foi

testada utilizando o protocolo Bluetooth e Wifi, sendo que o segundo se mostrou adequado ao nosso método. A aplicação do método no time de robôs e-puck foi uma das fases mais desafiadoras da pesquisa, tanto no desenvolvimento da comunicação entre robôs quanto na superação de problemas de hardware, porém, a troca de mensagens permitiu que os robôs apresentassem comportamento cooperativo em cenários reais. Os resultados obtidos a partir dos experimentos com robôs individuais e em time, apresentados respectivamente na seção 5.1 e na seção 5.2, nos permitiram a elaboração de um artigo que foi submetido recentemente ao *Robotics and Autonomous Systems Journal*. Uma cópia desse artigo é apresentada no Anexo B dessa dissertação. Segundo os autores dos artigos anteriores (FERREIRA; VARGAS; OLIVEIRA, 2014) e (OLIVEIRA; VARGAS; FERREIRA, 2015), que também são co-autores do novo artigo, o maior impedimento para a publicação do trabalho anterior em um *journal* foi justamente o fato de não ter sido possível realizar experimentos em time em robôs reais. Desta forma, temos uma boa expectativa que consigamos publicar os resultados dessa dissertação em um veículo de qualidade reconhecida na área de robótica.

Após a validação do novo modelo com experimentos reais, iniciamos uma nova etapa da pesquisa, na qual analisamos a estratégia de otimização bioinspirada para a troca de posições entre robôs na formação. Para tal, fizemos uso de um algoritmo genético capaz de avaliar e evoluir uma estratégia adaptativa para a reconfiguração da formação. Esse algoritmo foi executado três vezes para cada uma das oito diferentes configurações (relacionadas ao tamanho da população e à forma de ponderação utilizada na função de avaliação). Essa execução demandou um elevado tempo de processamento (25 dias) devido a necessidade de se avaliar cada indivíduo, a cada geração, através da simulação na plataforma Webots, para a tarefa de navegação do time de robôs em cinco cenários diferentes. O melhor indivíduo obtido, após todas as análises, apresentou comportamento híbrido em diferentes cenários, o que fez com que o método ficasse mais autônomo e robusto a situações variadas no ambiente, que requeriam diferentes padrões de movimentação.

Os resultados advindos dos testes em simulação e em cenários reais demonstraram que o método proposto possibilitou a geração de trajetórias mais suaves e eficientes, apresentando menos movimentos em zigue-zague e necessitando de menos tempo para que o robô realizasse o percurso. Além dessas melhorias, verificamos que nossa abordagem demanda poucos recursos computacionais, sendo funcional até em robôs com pouca memória e processadores sem alto desempenho.

6.2 Trabalhos Futuros

Para trabalhos futuros estudamos a possibilidade de utilizar uma camada de inferência fuzzy em conjunto com uma rede neural artificial na identificação dos obstáculos. Desta forma, esperamos analisar a qualidade da trajetória no desvio de obstáculos quando

comparada à estratégia utilizando AG.

Faremos também a inclusão de mais robôs ao time, permitindo a realização de novos padrões espaciais de formação e utilização de novas estruturas de time. Tais contribuições serão testadas tanto em simulação quanto em cenários reais, o que permitirá a análise da acurácia e robustez do método em diversos ambientes.

Pretendemos também, refinar a função de avaliação utilizada no algoritmo genético responsável pela otimização da reconfiguração da formação. Apesar dos inúmeros testes, percebemos algumas situações nas quais o resultado obtido por diferentes indivíduos não reflete as diferenças observadas visualmente nos cenários de simulação. Esse refinamento pode ser feito através da inclusão de mais fatores no cálculo da aptidão dos indivíduos.

Referências

AHMED, S.; AKHTER, A.; KUNWAR, F. Cellular automata based real time path planning for mobile robots. In: **12th International Conference on Control Automation Robotics Vision (ICARCV)**. [S.l.: s.n.], 2012. p. 142–147.

AMARAL, L. R. do et al. Oncogenes classification measured by microarray using genetic algorithms. In: **Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications**. Anaheim, CA, USA: ACTA Press, 2008. (AIA '08), p. 79–84. ISBN 978-0-88986-710-9.

ASL, A. N.; MENHAJ, M. B.; SAJEDIN, A. Control of leader-follower formation and path planning of mobile robots using asexual reproduction optimization (aro). **Applied Soft Computing**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 14, p. 563–576, jan. 2014.

BALCH, T.; ARKIN, R. Behavior-based formation control for multirobot teams. **Robotics and Automation, IEEE Transactions on**, v. 14, n. 6, p. 926–939, Dec 1998. ISSN 1042-296X.

BALLARD, L.; REN, W. Dynamic formation algorithms and experiments. In: **Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on**. [S.l.: s.n.], 2009. p. 1871–1876.

BERG, J.; KARUD, C. **Swarm intelligence in bio-inspired robotics**. Dissertação (Mestrado) — Department of Computer and Information Science, The Norwegian University of Science and Technology, 2011.

BERNARDINI, F. C. **Combinação de classificadores simbólicos utilizando medidas de regras de conhecimento e algoritmos genéticos**. Dissertação (Doutorado em Ciências de Computação e Matemática Computacional) — ICMC, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2006.

CABREIRA, T.; AGUIAR, M. de; DIMURO, G. An extended evolutionary learning approach for multiple robot path planning in a multi-agent environment. In: **IEEE Congress on Evolutionary Computation (CEC), 2013**. [S.l.: s.n.], 2013. p. 3363–3370.

CAPI, G.; MOHAMED, Z. Multiple robots formation - a multiobjctive evolution approach. **Procedia Engineering**, v. 41, n. 0, p. 156–162, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

CHEN, Y.-Q.; WANG, Z. Formation control: a review and a new consideration. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005). 2005**. [S.l.: s.n.], 2005. p. 3181–3186.

COUCEIRO, M. S.; VARGAS, P. A.; ROCHA, R. P. Bridging the reality gap between the webots simulator and e-puck robots. **Robotics and Autonomous Systems**, v. 62, n. 10, p. 1549 – 1567, 2014. ISSN 0921-8890. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0921889014000967>.

CYBERBOTICS. **Webots 7: Robot Simulator**. 2015. Disponível em: <http://www.cyberbotics.com/overview>.

DARWIN, C. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life. **John Murray, London**, n. 1, p. 1–556, out. 1859.

DAS, A. et al. A vision-based formation control framework. **Robotics and Automation, IEEE Transactions on**, v. 18, n. 5, p. 813–825, Oct 2002. ISSN 1042-296X.

DASGUPTA, P.; WHIPPLE, T.; CHENG, K. Effects of multi-robot team formations on distributed area coverage. **International Journal of Swarm Intelligence Research**, IGI Global, Hershey, PA, USA, v. 2, n. 1, p. 44–69, jan. 2011.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **Evolutionary Computation, IEEE Transactions on**, v. 6, n. 2, p. 182–197, Apr 2002. ISSN 1089-778X.

DIERKS, T.; JAGANNATHAN, S. Neural network control of mobile robot formations using rise feedback. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics**, v. 39, n. 2, p. 332–347, April 2009.

DING, Y.; HE, Y. Flexible formation using two kinds of cellular automaton in obstacle environment. In: **International Conference on Intelligent Control and Information Processing (ICICIP), 2010**. [S.l.: s.n.], 2010. p. 784–787.

DUAN, H. et al. Hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration. **IEEE Computational Intelligence Magazine**, v. 8, n. 3, p. 16–27, Aug 2013.

EDWARDS, D. et al. A leader-follower algorithm for multiple auv formations. In: **Autonomous Underwater Vehicles, 2004 IEEE/OES**. [S.l.: s.n.], 2004. p. 40–46.

FERREIRA, G.; VARGAS, P.; OLIVEIRA, G. An improved cellular automata-based model for robot path-planning. In: **Advances in Autonomous Robotics Systems**. [S.l.: s.n.], 2014. p. 25–36.

FERREIRA, G. B. **Modelos Baseados em Autômatos Celulares para o Planejamento de Caminhos em Robôs Autônomos**. Dissertação (Mestrado) — FACOM, Faculdade de Computação, UFU, Uberlândia, MG, Brasil, 2014.

FIDELIS, M.; LOPES, H.; FREITAS, A. Discovering comprehensible classification rules with a genetic algorithm. In: **Evolutionary Computation, 2000. Proceedings of the 2000 Congress on**. [S.l.: s.n.], 2000. v. 1, p. 805–810 vol.1.

FREDSLUND, J.; MATARIC, M. A general algorithm for robot formations using local sensing and minimal communication. **Robotics and Automation, IEEE Transactions on**, v. 18, n. 5, p. 837–846, Oct 2002. ISSN 1042-296X.

FREITAS, A. A. Advances in evolutionary computing. In: GHOSH, A.; TSUTSUI, S. (Ed.). New York, NY, USA: Springer-Verlag New York, Inc., 2003. cap. A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery, p. 819–845. ISBN 3-540-43330-9.

FRISCH, U.; HASSLACHER, B.; POMEAU, Y. Lattice-gas automata for the navier-stokes equation. **Physical Review Letters**, American Physical Society, v. 56, p. 1505–1508, Apr 1986. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevLett.56.1505>.

FU, Y.; LI, H.; MA, Y. Path planning of cooperative robotics and robot team. In: **IEEE International Conference on Robotics and Biomimetics, 2006. ROBIO '06.** [S.l.: s.n.], 2006. p. 1250–1255.

GE, S. S.; FUA, C.-H. Queues and artificial potential trenches for multirobot formations. **IEEE Transactions on Robotics**, v. 21, n. 4, p. 646–656, Aug 2005.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization and Machine Learning**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

GONZALES, N.; ALBERTO, C. Polymorphic virus signature recognition via hybrid genetic algorithm. **Oxford Brookes University**, 2011.

GREEN, D. G.; TRIDGELL, A.; GILL, A. Interactive simulation of bushfires in heterogeneous fuels. **Mathematical and Computer Modelling**, v. 13, n. 12, p. 57 – 66, 1990. ISSN 0895-7177. Disponível em: <http://www.sciencedirect.com/science/article/pii/0895717790900999>.

GUANGHUA, W. et al. Study on formation control of multi-robot systems. In: **Third International Conference on Intelligent System Design and Engineering Applications (ISDEA), 2013**. [S.l.: s.n.], 2013. p. 1335–1339.

HAASDIJK A. E. EIBEN, A. F. T. E. **Individual, Social and Evolutionary Adaptation in Collective Systems**. Singapore: Pan Starford Publishing, 2011. (Robotics and Intelligent Systems).

HARO, F.; TORRES, M. A comparison of path planning algorithms for omni-directional robots in dynamic environments. In: **Robotics Symposium, 2006. LARS '06. IEEE 3rd Latin American**. [S.l.: s.n.], 2006. p. 18–25.

HWANG, Y.; AHUJA, N. A potential field approach to path planning. **Robotics and Automation, IEEE Transactions on**, v. 8, n. 1, p. 23–32, Feb 1992. ISSN 1042-296X.

IOANNIDIS, K.; SIRAKOULIS, G. C.; ANDREADIS, I. A cellular automaton collision-free path planner suitable for cooperative robots. In: **Panhellenic Conference on Informatics**. [S.l.: s.n.], 2008. p. 256–260.

_____. Cellular ants: A method to create collision free trajectories for a cooperative robot team. **Robotics and Autonomous Systems**, v. 59, n. 2, p. 113–127, 2011.

_____. Depicting pathways for cooperative miniature robots using cellular automata. In: **International Conference on High Performance Computing and Simulation (HPCS), 2011**. [S.l.: s.n.], 2011. p. 794–800.

_____. A path planning method based on cellular automata for cooperative robots. **Applied Artificial Intelligence**, Taylor & Francis, Inc., Bristol, PA, USA, v. 25, n. 8, p. 721–745, set. 2011.

JÚNIOR, T. A. Magalhães. **Método Criptográfico baseado em Autômatos Celulares Bidimensionais para Cifragem de Imagens**. Dissertação (Dissertação de Mestrado) — FACOM, Faculdade de Computação, UFU, 2010.

KOBAYASHI, F.; TOMITA, N.; KOJIMA, F. Re-formation of mobile robots using genetic algorithm and reinforcement learning. In: **Annual Conference SICE 2003**. [S.l.: s.n.], 2003. v. 3, p. 2902–2907 Vol.3.

KOZA, J. R. **Genetic Programming: On the Programming of Computers by Means of Natural Selection**. Cambridge, MA, USA: MIT Press, 1992.

LAUSANNE, E. P. F. de. **E-puck Education Robot @ONLINE**. 2015. Disponível em: <http://http://www.e-puck.org/>.

LIN, C.-C.; HSIAO, P.-Y.; CHEN, K.-C. A motion planning of swarm robots using genetic algorithm. In: **International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010**. [S.l.: s.n.], 2010. p. 538–543.

MASTELLONE, S.; STIPANOVIC, D.; SPONG, M. Remote formation control and collision avoidance for multi-agent nonholonomic systems. In: **Robotics and Automation, 2007 IEEE International Conference on**. [S.l.: s.n.], 2007. p. 1062–1067. ISSN 1050-4729.

MATARIĆ, M. **The Robotics Primer**. MIT Press, 2007. (Intelligent robotics and autonomous agents). ISBN 9780262633543. Disponível em: <https://books.google.com.br/books?id=WWJPjgz-jgEC>.

MEAD, R.; LONG, R.; WEINBERG, J. Fault-tolerant formations of mobile robots. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009.** [S.l.: s.n.], 2009. p. 4805–4810.

MEHRJERDI, H.; SAAD, M.; GHOMMAM, J. Multi mobile robots formation in presence of obstacles. In: **IEEE International Conference on Mechatronics (ICM), 2011**. [S.l.: s.n.], 2011. p. 510–515.

MENG, Y.; GUO, H.; JIN, Y. A morphogenetic approach to flexible and robust shape formation for swarm robotic systems. **Robotics and Autonomous Systems**, v. 61, n. 1, p. 25 – 38, 2013.

MITCHELL, M. Computation in cellular automata: A selected review. **Non-standard Computation**, p. 385–390, 1996.

MONTEIRO, S.; BICHO, E. Robot formations: Robots allocation and leader-follower pairs. In: **Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on**. [S.l.: s.n.], 2008. p. 3769–3775. ISSN 1050-4729.

NAVARRO, I.; MATIA, F. A survey of collective movement of mobile robots. **International Journal of Advanced Robotic Systems**, v. 10, n. 73, 2013.

OGREN, P.; EGERSTEDT, M.; HU, X. A control lyapunov function approach to multiagent coordination. **IEEE Transactions on Robotics and Automation**, v. 18, n. 5, p. 847–851, Oct 2002.

OLIVEIRA, G.; VARGAS, P. A.; FERREIRA, G. A local decision making cellular automata-based path-planning. **11th National Meeting on Artificial and Computational Intelligence (ENIAC2015), BDBComp, Brazilian Computer Society**, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/eniac/2014/00102.pdf>.

OLIVEIRA, G. M. B. Autômatos celulares: aspectos dinâmicos e computacionais. In: **III Jornada de Mini-cursos em Inteligência Artificial (MCIA)**. [S.l.]: Sociedade Brasileira de Computação, 2003. v. 8, p. 297–345.

OLIVEIRA, G. M. B. et al. Secret key specification for a variable-length cryptographic cellular automata model. In: SCHAEFER, R. et al. (Ed.). **Parallel Problem Solving from Nature, PPSN XI**. [S.l.]: Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, v. 6239). p. 381–390. ISBN 978-3-642-15870-4.

PUGH, J.; MARTINOLI, A. The cost of reality: Effects of real-world factors on multi-robot search. In: **Robotics and Automation, 2007 IEEE International Conference on**. [S.l.: s.n.], 2007. p. 397–404. ISSN 1050-4729.

REZAEE, H.; ABDOLLAHI, F. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. **IEEE Transactions on Industrial Electronics**, v. 61, n. 1, p. 347–354, Jan 2014.

SCHNEIDER, F.; WILDERMUTH, D. A potential field based approach to multi robot formation navigation. In: **IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003**. [S.l.: s.n.], 2003. v. 1, p. 680–685 vol.1.

SILVA, R. O.; RIBEIRO, M. de S.; AMARAL, L. Rodrigues do. Building high level knowledge from high dimensionality biological dataset (nci60) using genetic algorithms and feature selection strategies. In: **Evolutionary Computation (CEC), 2013 IEEE Congress on**. [S.l.: s.n.], 2013. p. 578–583.

SOUTO, M. C. P. e. a. Técnicas de aprendizado de máquina para problemas de biologia molecular. **Sociedade Brasileira de Computação**, 2003.

SUKHATME, G. S.; MATARIC, M. J. Robots: Intelligence, versatility, adaptivity - introduction. **Commun. ACM**, v. 45, n. 3, p. 30–32, 2002.

TIKHANOFF, V. et al. An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator. In: **Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems**. New York, NY, USA: ACM, 2008. (PerMIS '08), p. 57–61.

WAHLE, J. et al. A cellular automaton traffic flow model for online simulation of traffic. **Parallel Computing**, v. 27, n. 5, p. 719 – 735, 2001. ISSN 0167-8191. Cellular automata: From modeling to applications. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167819100000855>.

WARD, D.; MURRAY, A.; PHINN, S. A stochastically constrained cellular model of urban growth. **Computers, Environment and Urban Systems**, v. 24, n. 6, p. 539 – 558, 2000. ISSN 0198-9715. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0198971500000089>.

WOLFRAM, S. Cellular automata. **Los Alamos Science**, v. 9, p. 2–21, Jul 1983.

_____. Statistical mechanics of cellular automata. **Reviews of Modern Physics**, v. 55, p. 601–644, 1983.

_____. **A New Kind of Science**. Champaign, Ilinois, US, United States: Wolfram Media Inc., 2002.

ZUO, G.; HAN, J.; HAN, G. Multi-robot formation control using reinforcement learning method. In: **Proceedings of the First International Conference on Advances in Swarm Intelligence - Volume Part I**. Berlin, Heidelberg: Springer-Verlag, 2010. (ICSI'10), p. 667–674.

Anexos

ANEXO **A**

Vinte cenários de simulação.

A seguir estão os vinte cenários utilizados para o teste em simulação dos indivíduos gerados pelo AG.

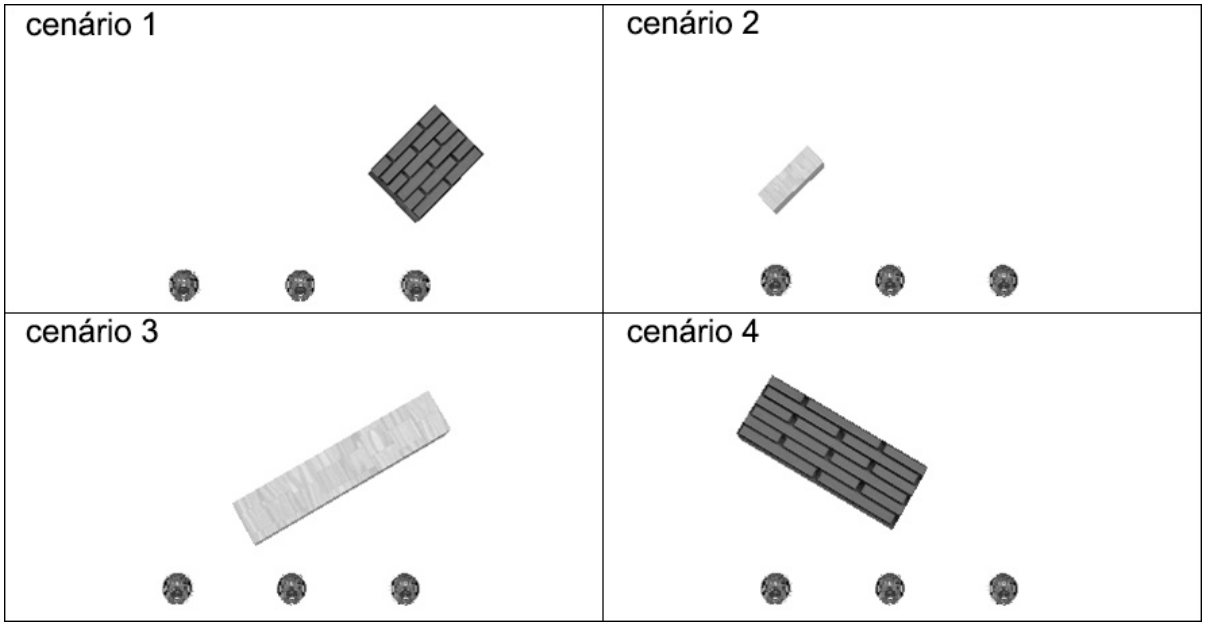

Figura 59 – Grupo de cenários 1.

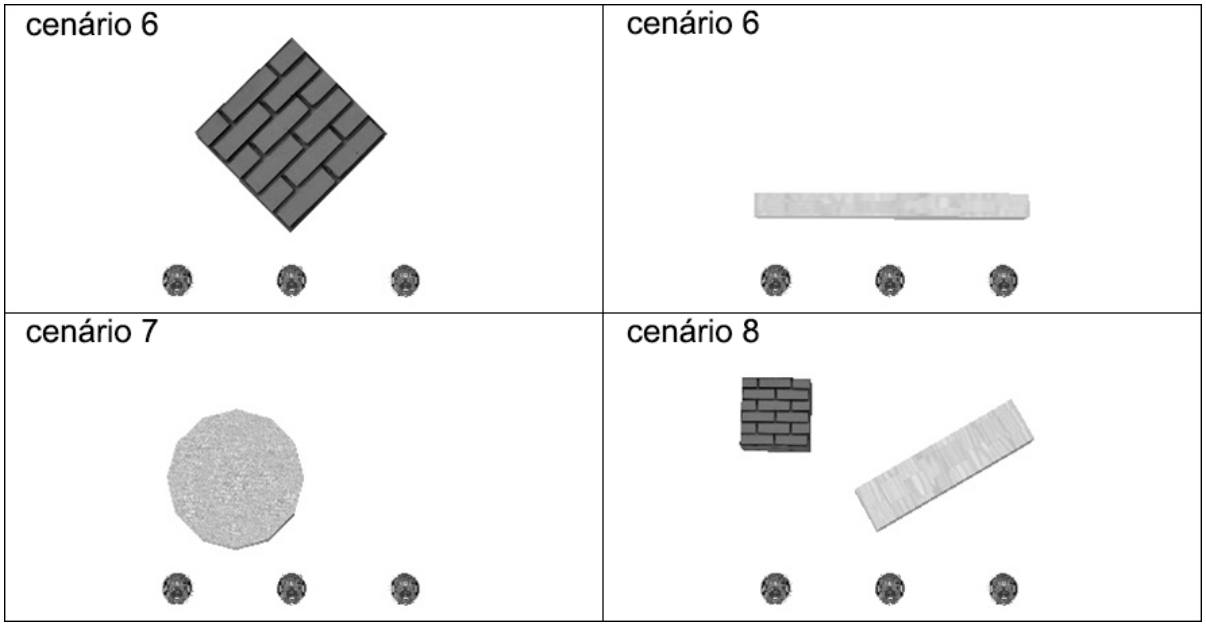

Figura 60 – Grupo de cenários 2.

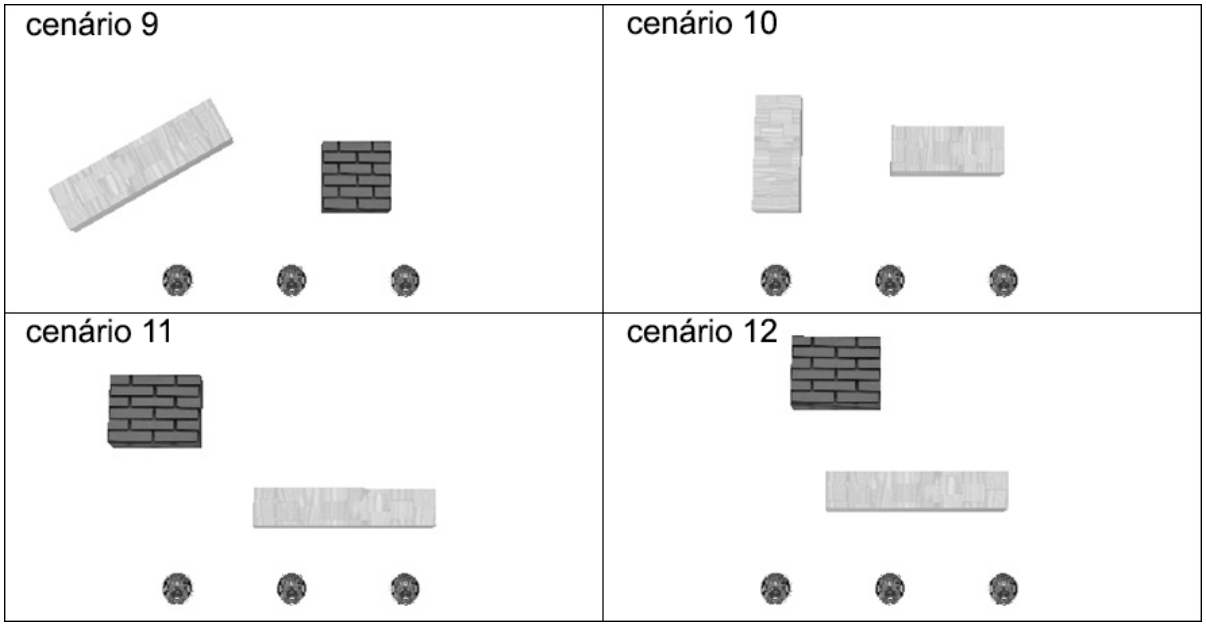

Figura 61 – Grupo de cenários 3.

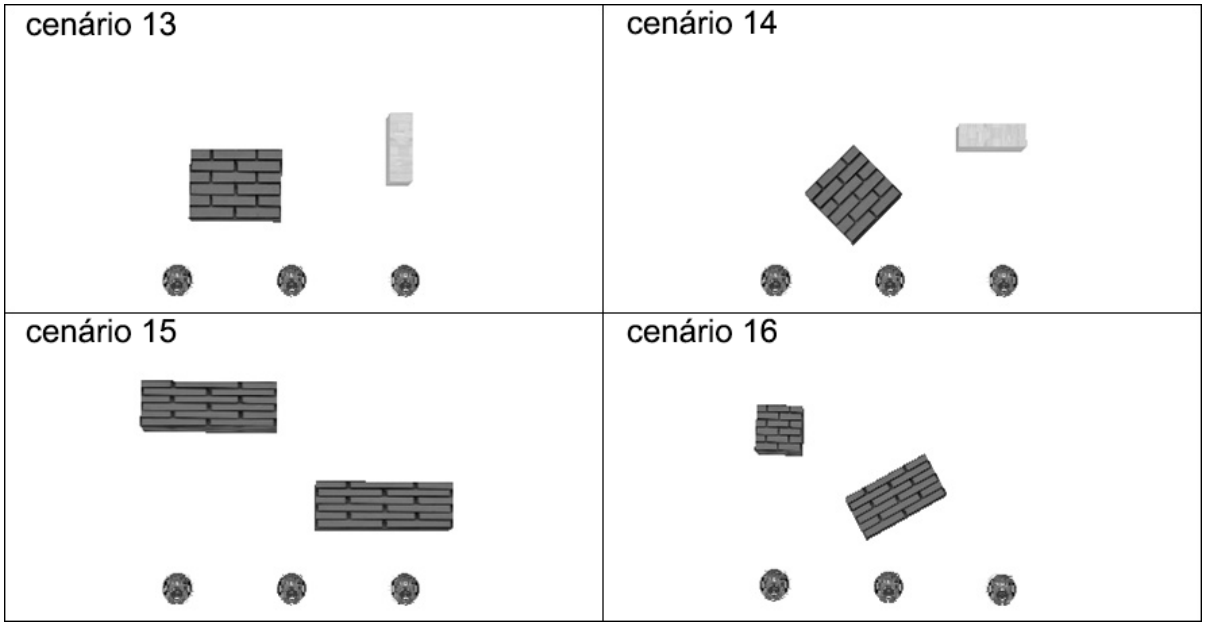

Figura 62 – Grupo de cenários 4.

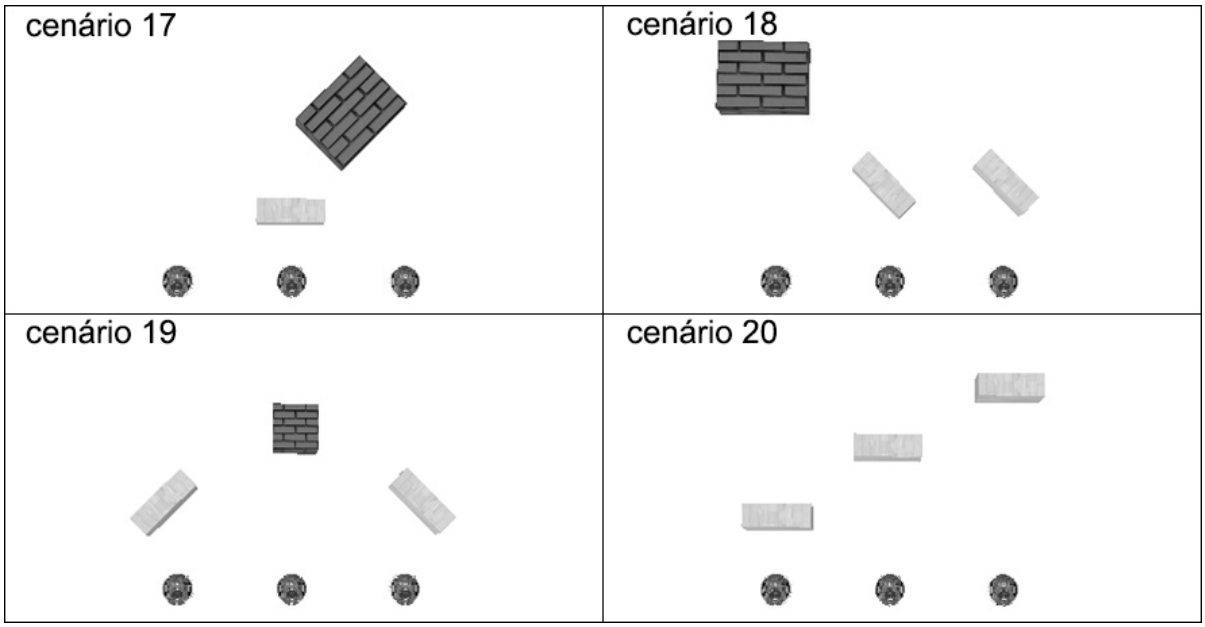

Figura 63 – Grupo de cenários 5.

ANEXO **B**

Artigo Submetido

A seguir está o artigo submetido ao *Robotics and Autonomous Systems Journal*.

A Novel Cellular Automata-Based Path-Planning for a Cooperative and Decentralized Team of Robots

Reslley Gabriel Oliveira Silva*, Giordano Bruno Santos Ferreira*, Micael Santos Couceiro†
Laurence Rodrigues do Amaral*, Patricia Amancio Vargas‡, Gina Maira Barbosa de Oliveira*

*School of Computation
Federal University of Uberlandia
Uberlandia, Minas Gerais, Brazil

†Institute of Systems and Robotics
University of Coimbra
Coimbra, Portugal

‡School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, Scotland, United Kingdom

Email: *reslleygabriel@gmail.com, giordanobsf@hotmail.com, laurence@ufu.br, gina@ufu.br,
†micaelcouceiro@isr.uc.pt, ‡p.a.vargas@hw.ac.uk

***Abstract*—In this paper, a novel cellular automata-based model is proposed to solve the path-planning and formation control problem for a team of robots, using transition rules and cellular automata (CA) states. Planning collision-free trajectories is essential for autonomous robots when moving in unknown environments. The complexity of this task increases in multi-robot systems, especially when all robots must adjust their actions in order to keep their initial team formation pattern. Here, a decentralized and autonomous robot team path-planning is proposed and tested using different communication technologies. The proposed method is implemented on a simulation environment and on real e-puck robots. The experimental results show that the proposed method can create accurate collision-free paths in real-time by improving the multi-robot system performance.**

Keywords*—path-planning, formation control, autonomous robots, cellular automata, multi-robot system.

I. Introduction

One of the major problems in robotics is planning the motion of robots from an initial point to a final goal [1]. Although robot motion-planning is widely studied, coordinating the movements of a multi-robot system (MRS) while avoiding obstacles along their trajectory is still a challenging task [2]. The strategies to achieve such coordination can be applied to a number of applications, from cooperative mapping to search and rescue, surveillance, and agriculture [3].

An MRS, composed by simple robots, offers more advantages than a large, complex, and expensive single-robot. Multiple robots, when precisely coordinated, can fulfill more complex tasks by improving efficiency and robustness, and providing parallelism (redundancy) [4], [5].

Collective coordination problems in robotics are usually classified either in formation control or flocking. Formation control is defined as the coordination of a group of robots to get into and to maintain a formation with a specific shape, while moving in the environment. On the other hand, flocking consists of moving a group of robots without considering the shape and relative positions between the robots [6].

Various techniques have been proposed in the literature to solve the formation control problem [7], [8], [9], [10]. However, real-time robot navigation requires a path-planning method to be a fast and efficient procedure, and the majority of the methods proposed in the literature either suffers from a high computational cost, or presents a constrained implementation [11].

Nonetheless, some previous approaches use the advantages of parallel computation and simplified rules of cellular automata (CA), providing a low computational cost and being functional even when working with robots that have slow processors and small memory [12]. In this paper, we introduce a method for dynamic path-planning and formation control. Our technique is based on a CA model using local decisions rules to define the next movement, solely based on the robots' neighborhood and communication.

In a preliminary phase, we used the Webots simulator [13] to reproduce the results from [12], [14], and [15]. We also applied the models to a team of real e-puck robots [16] and analyzed the robots' behavior under different environments. From this analysis, we uncovered two main shortcomings in those approaches. First, there was a lack of efficiency on the trajectory performed by the previous methods, e.g. the number of rotations and zigzag movements that were too frequent and the robot not performing a straight path. CA rules used in these previous methods do not consider a constant diagonal movement. As a result, the robots were unable to entirely negotiate the obstacle's corner before fulfilling the formation control rules. The second issue was the centralized structure of leader-slaves robots. In the previous models, the leader was responsible for all the decision-making regarding the formation maintenance and communication between robots (performed at every time step).

The major contributions of this paper can be highlighted by comparing it with the previous CA-based works [12], [14], and [15]. First of all, the obstacle's corner surpassing which was pointed as a weakness of the previous model [12] was solved here with a much smoother behavior than with the model in [14] and [15], due to the addition of new states and more complex rules; the resultant robot behavior returns better trajectories and smaller residual errors.

Although the model discussed in [14] and [15] also pre-

sented some improvements over [12], this subsequent model was developed and evaluated only in scenarios involving a single e-puck robot, while the major purpose of the model in [12] was to deal with the formation control problem in robot teams. Here, we continued the efforts started in [14] and [15], in order to finally apply the new model to the formation control problem using multi-robot scenarios. This extension forced us to deal with inherent communication issues involving the coordination of the robot team, which results in some additional improvements compared with [12]: (*i*) a faster communication protocol is employed here for the team coordination (Wifi instead of Bluetooth); (*ii*) a more decentralized communication structure is used to improve the autonomy of robotic agents; (*iii*) a previously published communication architecture is used [17], which enables a cross-compatibility between simulations and real e-pucks experiments.

The techniques proposed in this work were compared to the related literature. The results obtained through simulations and real robot experiments show improvements in the robots efficiency via a better obstacle-avoidance maneuver and robustness in different scenarios, apart from minimizing the robot-robot communication load.

The paper is organized as follows. Section II briefly defines CA concepts. Section III categorizes previous path-planning and formation control approaches. Section IV presents our proposed method, with the new set of improved rules and states. Section V describes the robot architecture and the environment details. Experiments and results are presented and analyzed in Section VI. Finally, conclusions and suggestions for future work are outlined in Section VII.

II. Cellular Automata

In the 1950s, Ulam and von Neumann introduced CA as an artificial mechanism able of self-reproduction [18]. The standard CA computational structure presents the following characteristics:

- Discrete time, space and states;
- *N*-dimensional lattice (cellular space) of identical components (cells);
- Local connectivity (concept of neighborhood);
- Deterministic and synchronous transition rule;
- Boundary conditions (used to define the neighborhood of cells belonging to lattice boundaries).

In CA, every cell is labeled with a discrete value from the set of states. Then, the transition rule is applied according to the cell state at the current time step and its neighborhood (generally composed by its nearby cells states) to find the next state of each cell of the cellular space. The entire lattice is updated simultaneously, thus, providing an inherent parallel system. Some of the key features of the CAs that led them to be considered in robotics are: *(i)* it requires low computational cost; *(ii)* it can represent high complex phenomena; *(iii)* it has simple rule-based behavior and do not need complete information of the environment.

An elementary CA has a 1D lattice, binary cell states and radius-1 neighborhood (formed by three cells). The transition rule define the output bit for each one of the eight possible three-cell binary neighborhood. As we can see in Figure 1, after the transition rule is applied to the lattice in time t0 a new lattice configuration is generated. For example, in the updating of cell 1 of lattice t0 the transition rule uses the neighborhood 110, which changes the center cell's state from 0 to 1 in the t1 lattice. A similar procedure is applied to each cell of t0, leading to the t1 configuration.

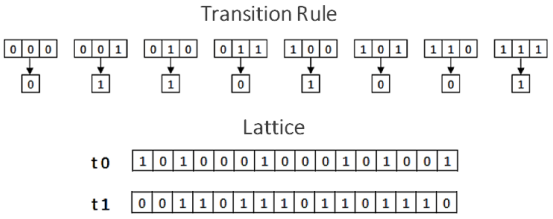

Fig. 1. Lattice configuration at the initial time (t0) and after the synchronous application of the transition rule over all cells (t1).

Considering 2D lattices, two types of neighborhood are commonly used: the von Neumann, which is formed by the center cell and its four neighbors in the cardinal directions as shown in Figure 2a, and Moore's, which also includes the diagonal cells in its neighborhood, as presented in Figure 2b.

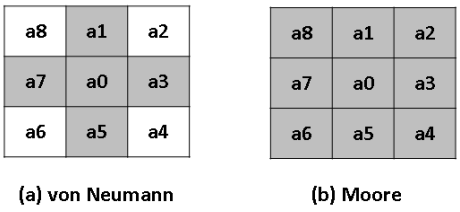

Fig. 2. In the von Neumann approach the neighborhood is composed by the a0, a1, a3, a5 and a7 cells. Using the Moore's neighborhood all the cells in a 1-radius-distance are incorporated.

Figure 3 shows an example of a two dimensional CA using the von Neumann neighborhood. The upper part of the figure shows 32 outputs related to each five-cells possible neighborhood, while the lower part shows the lattice configuration for 2 consecutive time steps. For example, considering the cell on the second row and fifth column of the t0 lattice (state 0) we can observe the change of its state in the t2 lattice. In this case, according to the neighborhood and the respective output bit of the transition rule the cell's state changed from 0 to 1.

The new model proposed in this paper uses Moore's neighborhood and it was strongly based on the results presented in [14], where the authors proposed an adaptation to [12] using scenarios with a single robot, in order to analyze the robot's behavior exclusively due to the CA rules proposed in [12]. In the next section we will present some papers in which other approaches were used to the path-planning.

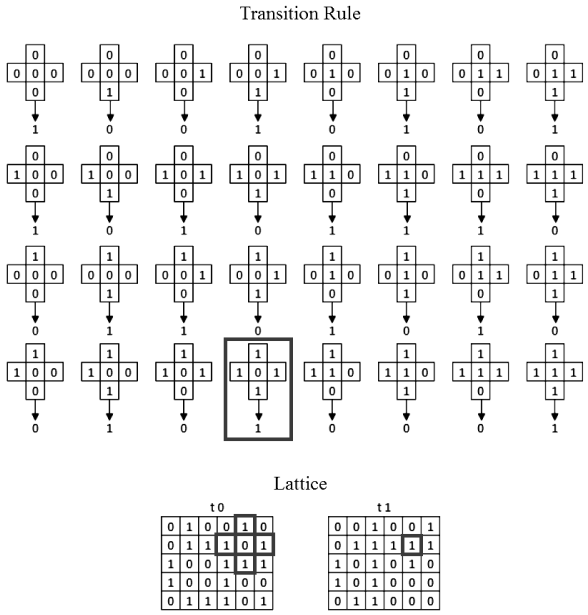

Fig. 3. Lattice configuration at the initial time (t0) and after the synchronous application of the transition rule over all cells (t1).

III. Related Work

The authors in [19] proposed a method for performing cooperative tasks between multiple robots, adapting to obstacles while keeping formation. The authors used a fuzzy controller and the concept of leader-followers, where a robot is named as leader (normally the one that is closer to the goal) and the others follow it synchronizing their actions and keeping the formation (angle and speed, relative to the leader). When a robot detects an obstacle, it can request a change in the formation structure. After the change, the other robots begin to cooperate with the new leader in its path to the goal. All the actions were taken based only on the information collected by the robot's sensors in the simulation.

In the approach proposed in [20], the goal was to establish and maintain a certain team formation, where a "friend" robot was used as a reference in a particular viewing angle. The robots work with minimal use of sensors and communication, each of which has no global knowledge of position or direction of other robots. Each robot passes its ID regularly, and a robot "driver" (that does not follow any other "friend" robot) decide its direction and passes the pattern to be held by the formation.

In [21], it was aimed to make the robots follow a path to the goal while dodging obstacles and keeping formation. Generally, the formations carried by robots have geometric shapes, and may be subdivided into multiple line segments, where each line segment can be considered as one queue. In the paper, the authors proposed a concept of potential trenches, that attracted the robots and caused them to move in the direction of the queues. Also, to avoid collision, the obstacles were treated by a repulsive force.

A method that uses an external vision system to keep the formation was proposed by [22]. In the authors approach, the path of the leader is calculated a priori and the coordination scheme is based on the distance and angle between the followers and the leader. However, the method does not guarantees treating collision between robots and obstacles in the field. The work in [23] used the leader-followers approach in order to keep a robot formation, while navigating by preset points, a water mining scanning and discovery mission. All robots knew the leader's path and adjusted their trajectory in order to keep the desired spatial pattern. The communication process was done via the sounds produced by the leader.

With the objective of following a moving target while maintaining formation, the authors in [24] used a consensus technique along with a modified Kalman filter. The robots keep a spatial pattern, but not too close to each other (to avoid collision) and not to far (to avoid loss of communication). To accomplish this goal, the authors used potential functions for cohesion and separation. The target was a moving robot, identified by a red marker. All followers detected the target using a camera and color detection techniques.

In [25] the authors proposed a cooperative algorithm based on consensus to define the path to a goal and then follow it while keeping formation. The trajectory information was passed only to a group of robots (leaders), while the other robots had to figure out the formation pattern based on the information exchanged between neighbors.

The approach proposed in [26] is divided in path-planning and formation control. The first task is obtained using Voronoy diagrams, in which the center of the spatial pattern must follow the generated path. The motion planning is done by a genetic algorithm based on potential functions of attraction (goal) and repulsion (obstacles), where the chromosome is defined by the coordinates of all the robots. The generated path is divided into various points and, at each point, the GA finds the best configuration for the robot team.

The authors in [27] proposed a formation and path planning method using Asexual Reproduction Optimization (ARO). In this technique, there is only one individual who reproduces itself, in which a new individual is generated by the parent mixture with a version of himself that is mutated in certain parts. The original individual (father) and the generated individual (son) compete (based on a fitness function) and the best individual survives. The formation control task is achieved considering a robot leader as a moving target, which the other robots should follow using potential functions. The robots must ensure a certain angle to the target so that the formation is maintained.

A multi-objective evolutionary algorithm (NSGA II), proposed by [28], evolves controllers based on neural networks for the task of controlling multiple e-puck robots in formation. The robots start at random positions on the environment, where they must get in formation with a specific spatial pattern towards reach the goal. The distance and angle to the leader are determined using a camera installed in the robot and the communication is handled using Bluetooth and a central computer.

A decentralized formation control approach was proposed by [29], where the Pioneer3Dx robots must first use consensus for setting their IDs and achieve formation (they do not have

position information or leader defined). After, they can use the leader-followers strategy, following neighbors or a hybrid approach between the previous two. They are able to perform various configurations and reconfigure the formation whenever a robot failure occurs. The method is divided into three steps: (*i*) after the agreement between the robots, the geometric formation is established, with each robot in a specific position; (*ii*) the robots move through the environment while maintaining the formation; (*iii*) a spatial pattern reconfiguration is performed according to the task or changes in the environment.

The path-planning and formation control problem was also studied in [7]. The main objective of the paper was to keep a robot team in formation while covering the environment. They designed a dual layered intelligent coordination algorithm that models each robot's path using a fifth order polynomial and has two modes of operation. The first one is the fuzzy path following and coordination mode, which enables the group to arrive at their individual destinations within the same time. The second one is the fuzzy obstacle detection and avoidance mode that recalibrates a new trajectory to move around obstacles whenever they are detected [7].

In order to achieve a time-varying formation, a translational synchronous controller for wheeled mobile robots was proposed in [8]. The method used Lagrange multipliers to achieve the nonholonomic constraints and the formation kept through the synchronization of movements between neighboring robots. With the same objective, Rezaee and Abdollahi [9] used a virtual structure in which each mobile robot was modeled by an electric charge. A virtual mobile robot was placed at the center of the desired shape and the formation was accomplished by the result of attractive and repulsive forces between robots. The obstacle avoidance method was based on a behavioral structure that used a rotational potential field to lead the robots following an optimized trajectory.

Aiming to form complex shapes with a team of robots in a distributed manner, the work in [10] proposed a morphogenetic approach, using a gene regulatory network. In this approach, each robot was seen as a cell containing a virtual DNA that encoded the target global pattern formation. Although the desired spatial formation configuration was given to each robot, they were unaware of any environment or fixed position information, therefore, they should find their position only through local processing and communication.

All the works previously cited were focused in formation control, but none of them uses CA models to perform this task, as the approach in the present work. On the other hand, different works in the literature have studied CA models in robot's path-planning.

According to [14], previous CA-based path-planning methods can be classified into six distinct approaches. **Power Diffusion** [30], in which a determined neighborhood direction is defined using the concept of power diffusion. **Goal Attraction** [31], which is based on goal attraction values that are stored in a multi-layer CA. **Voronoi's Diagram** [32], where the purpose is to find a path as distant as possible from obstacles, using the space retraction on a Voronoi's Diagram. **Distance Diffusion** [1], which employs a CA model to calculate the distance between each free cell and the goal. **Local decision making** [11], [12], [14], [15], [33], [34] and [35], in which the robot's next movement is decided based on its neighborhood, constructed at each time step using sensor's readings, without a prior knowledge of the environment. **Message sending** [36], where the search intelligence happens in the environment and each cell communicates with its neighbors.

Although all the previous approaches are based on CA-models, only the works classified into the local decision approach were focused in formation control [12] and [14]. The model proposed in [12] focused on navigating a team of robots under a specific formation from their initial positions to their final goals plus deviating from eventual obstacles. The environment was modeled as a 2D lattice divided into a simple rectangular grid of identical 1 square centimeter cell. The authors used Moore's neighborhood, considering that the robot occupies only the center cell. Every CA cell would then evolve its state according to both the neighboring cells and its own current state. To evolve, a local transition rule was applied. The CA planning algorithm then executes two different sets of rules: collision avoidance and formation control.

- **Collision avoidance**: A reactive mode in which the robot scans its neighborhood at every time step in order to detect a potential obstacle. This is accomplished by comparing all the values from the sensors' readings (with pre-established thresholds). After updating the status of the adjacent cells, the appropriate CA rule is selected depending on the robot's direction in the center cell.
- **Formation control**: The robot has to keep navigating over its initial column during the entire route, i.e., it has to maintain its trajectory in the axis formed by the initial position and the goal whenever possible. If the occurrence of an obstacle makes the robot to deviate, the robot must return to this axis as soon as possible. When the neighborhood is free of obstacles, a set of rules is applied, thus moving the robot back to its original axis.

All cells update their state simultaneously by applying the appropriate local transition rule at each time step, which can also result in a given robot action. There are three possible cell states: Free (*F*), Obstacle (*O*) and Robot (*R*), and two eligible actions: (*i*) to move to an adjacent cell keeping the current orientation; or (*ii*) to make a rotation staying in the same cell. The robot's rotation angle, as we can see in Figure 4, can take five possible values and interpretations: $-90°$ (it points to a7), $-45°$ (it points to a8), $0°$ (the robot points to a1), $45°$ (it points to a2), $90°$ (it points to a3).

Unfortunately, the infrared sensors on the e-puck are not equally distributed at its circumference, thus, sometimes, obstacle detection cannot be successfully achieved. Aware of this fact, the authors in [14] found a **corner deadlock problem** in the model proposed in [12], observed both in simulation and real world experiments. The problem led the robot to remain rotating in the same axis indefinitely. As shown in Figure 5, (a) the robot identifies an obstacle, (b) it turns left to deviate from the obstacle, (c) it continues to deviate until the obstacle's corner, where the infrared sensor is not able to identify it, (d) the robot returns to its original direction and its frontal sensors detects the obstacle, (e) it turns left and again its sensors do

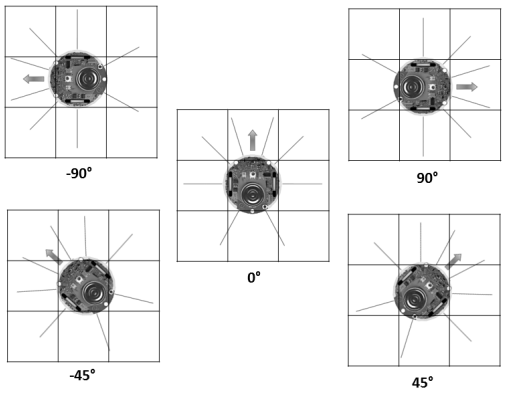

Fig. 4. Five angles of movement of the robot.

not detect the obstacle, (f) the robot turns right and identifies the obstacle again.

In order to solve this problem, the authors included a new state that identifies if the robot has come from a previous rotation due to some obstacle identification in an early step. With the Rotated-Robot (*RR*) state, the robot recognizes a "corner scenario" and makes a new step to the left to deviate from the corner, even if its side sensors are not detecting the obstacle. This new state and consequent modifications in the deviation rules made the robot to be able to surpass the corner. However, experiments with real e-puck robots showed that the robot behavior was still not satisfactory.

On a later paper, the authors in [15] added new states to the CA model in order to improve the robot's movements when surpassing the obstacle's corner. Whenever the robot identifies a possible corner, it moves four consecutive steps. As such, four states Rotated-Robot-i (*RR-i*) were used, where variable *i* represents the number of steps. Another state, called Formation-Robot (*FR*), was also incorporated to make the robot return to its original crossing axis, through diagonal movement steps. As a result, the robot alternates between vertical and diagonal steps until it returns to the correct position (or until a new obstacle is detected). These modifications solved the corner deadlock problem and made the robots surpass obstacles.

Figure 6 shows an example of the e-puck simulation when the model in [15] is applied to a simple scenario with just one obstacle: (a) the robot moves towards its goal, (b) it identifies an obstacle, (c) it turns left and start to deviate from the obstacle, (d) the robot's sensor do not detect the obstacle anymore, (e) it turns right, returning to the goal's direction and identify the obstacle again, (f)-(g) it turns left and execute the four consecutive steps maneuver, (h) after the four steps, the robot turns right, (i)-(k) the robot tries to return to its original axis, alternating between frontal and diagonal steps, (l) the robot identifies the obstacle and starts the collision avoidance maneuver again.

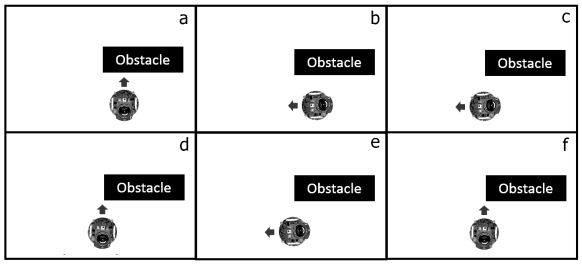

Fig. 5. In the corner deadlock situation, found using the model in [12], the robot can not surpass an obstacle due to its misplaced sensors and incomplete set of rules.

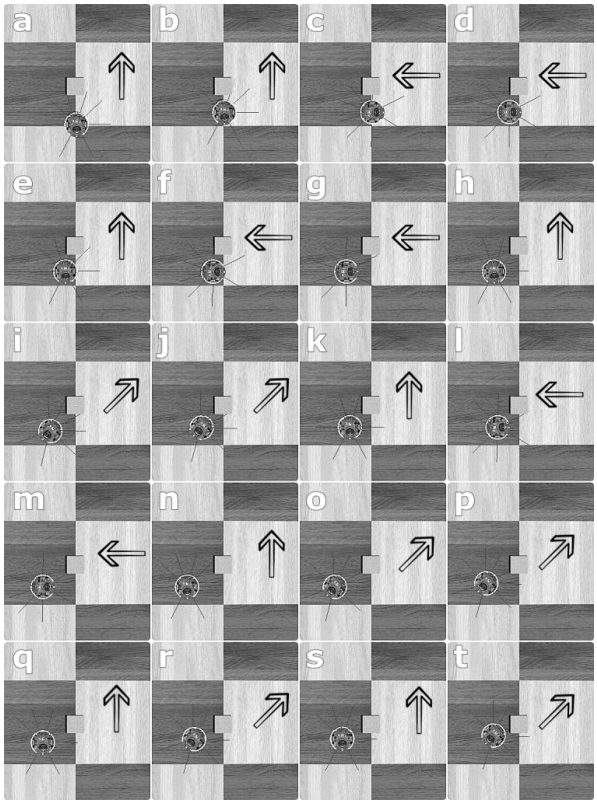

Fig. 6. Robot behavior when surpassing a corner in simulation of the model in [15].

IV. A Novel Cellular Automata-Based Path-Planning Method for a Team of Robots

Although the modifications and improvements performed in the CA model in [14] and [15] returned a collision-free navigation planner, there were still some shortcomings. For instance, an expressive number of rotations and zigzag movements were still observable when the robot was negotiating an obstacle.

Moreover, the CA methods described in [14] and [15] were based on the previous model proposed in [12] to a multi-robot scenario. Nonetheless, they were applied only to single-robot

scenarios, both in simulation and with a real robot. Therefore, new shortcomings in the robot's behavior were spotted when we started to investigate the application of the CA model to a team of robots. In such a scenario, the excessive rotation and zigzag movements of the robots deviating from an obstacle jeopardized the maintenance of a desirable team formation.

We observed that those movements were caused by two major facts: (*i*) the robots were not able to entirely negotiate the obstacle's corner before applying the formation control rules, and (*ii*) the CA rules did not consider a constant diagonal movement (instead of the alternation between forward and diagonal shifts).

A. Additional CA States and Improved Set of Rules

Based in our observations on the experiments with real robots using previous models [12], [14] and [15], in this paper we propose a novel improved CA model able to deal with the navigation of a robot team, capable of maintaining a pre-specified team formation while performing an efficient and smooth trajectory. The major modifications related to the CA rules in the model were: (*i*) the inclusion of five new CA states; and (*ii*) the inclusion of fourteen new rules to deal with the new states and to provide a more efficient behavior.

We solved the problem of completely negotiating an obstacle's corner by adding four new Formation-Robot (*FR-i*) states. Each robot identifies an obstacle using its frontal sensors and decides whether to turn $90°$ or $-90°$ based on the values from its sensor readings. Next, it starts its motion towards the obstacle's corner, where it applies the four Rotated-Robot states (*RR-i* states, as proposed in [14]). After these four steps in the orthogonal direction, it is expected that the robot is far enough from the corner, it then applies four consecutive movements towards the goal. Thus, the robot will try to use the new Formation-Robot states to perform four steps towards the goal, unless a new obstacle is identified. With this new scheme, it is possible to guarantee that the formation control phase begins after the avoidance of the obstacle's edge. Figure 7 shows an example of the new model: (a) the robot detects the obstacle, (b) the obstacle avoidance maneuver starts, (c) the robot detects the edge of the obstacle, (d) it avoids the edge by giving four steps in $-90°$, (e) the robot completes the edge avoidance with four steps in $0°$, (f) the robot is now ready to apply formation control rules.

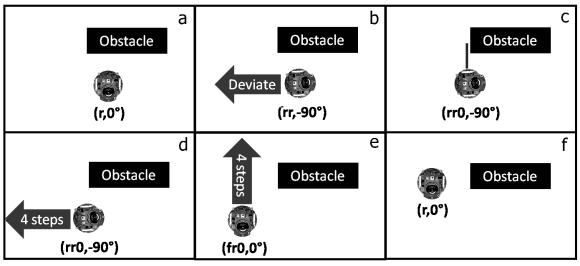

Fig. 7. Example of application of the new model using the *FR-i*) states.

The shifts between forward and diagonal steps implemented in [15] generated undesired zigzag movements and, as the number of rotations increased, the accumulated error in the odometry measurements affected the control system. In order to enhance the robot's performance, we designed a new state called Formation-Robot-Diagonal (*FRD*). This new state indicates to the robot can move constantly in $45°$ or $-45°$, while trying to resume the formation, thus improving the path quality and reducing the localization error. Figure 8 shows a situation in which this state is used. These snapshots can be seen as the continuation of the snapshot (f) in Figure 7, after the obstacle avoidance maneuver. In Figure 8: (a)-(e) the robot applied the formation control rules, moving in constant diagonal steps with the *FRD* state, (f) the robot returned to its original axis and angle. After this, the robot will restart to move over its original axis, unless a new obstacle is found.

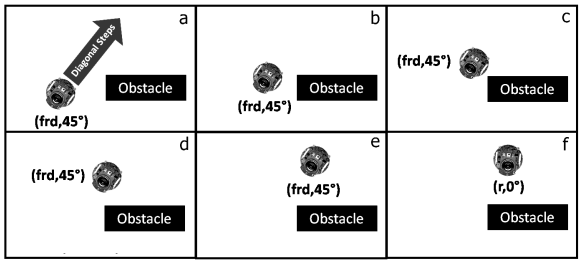

Fig. 8. Example of application of the new model using the *FRD*) states.

Considering the new states and a diagonal neighborhood update, we generated a new set of improved rules for collision avoidance and formation control. Table I presents the detailed set of the collision avoidance rules in the novel model. The fourteen new rules proposed in this model are highlighted. We can see in the second row of the table, for example, that when the cell a5 is in state *RR-3* (Rotated-Robot-3) and there is no obstacle in the neighborhood, the robot has already moved from the obstacle and is now going towards the goal (angle $0°$). Hence, it starts the four-steps-maneuver to avoid the obstacle's corner and changes to the state *FR-0* (Formation-Robot-0). After achieving the state *FR-3*, the robot will try to regain the team topology, using the new formation control rules, showed in Table II. The c-axis parameter represents the current axis in which the robot is moving, and the o-axis parameter represents the original axis, which the robot should pursue.

B. Team Communication and Structure

The method proposed in [14] and refined in [15] was only tested using single robot environments. However, it was designed to work equally using various robots in formation. Thus, we applied it to a team of real robots and analyzed its performance. We noticed that the static structure of leader-slaves robots (also used in [12]) showed an undesirable inflexibility. In these models, the leader is responsible for all the decision making regarding the team formation and the communication between robots at every time step. The slaves have to send their state, horizontal and vertical positions to the master, which has to process all the information in a centralized way and then return the action for each slave, which contradicts the idea of distributed CA-based architectures.

Aiming to improve the independence of the team components, we used a new approach, which gives more autonomy

TABLE II. NEW SET OF FORMATION CONTROL RULES

Cell state at time t											a0 state at time t+1	
If	**a0**	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**Angle**	**a1**	**New Angle**
c-axis - o-axis = 0	*R*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	0°	*F*	0°
c-axis - o-axis = 0	*F*	*F*	*F*	*F*	*F*	*R*	*F*	*F*	*F*	0°	*R*	0°
c-axis - o-axis >0	*R* or *FR3*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	0°	*F*	45°
c-axis - o-axis >0	*F*	*F*	*F*	*F*	*F*	*R* or *FR3*	*F*	*F*	*F*	0°	*FRD*	45°
c-axis - o-axis <0	*R* or *FR3*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	0°	*F*	-45°
c-axis - o-axis <0	*F*	*F*	*F*	*F*	*F*	*R* or *FR3*	*F*	*F*	*F*	0°	*FRD*	-45°
c-axis - o-axis >0	*FRD*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	45°	*F*	45°
c-axis - o-axis >0	*F*	*F*	*F*	*F*	*F*	*F*	*FRD*	*F*	*F*	45°	*FRD*	45°
c-axis - o-axis <0	*FRD*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	*F*	-45°	*F*	-45°
c-axis - o-axis <0	*F*	*F*	*F*	*F*	*F*	*F*	*FRD*	*F*	*F*	-45°	*FRD*	-45°

TABLE I. NEW SET OF IMPROVED OBSTACLE AVOIDANCE RULES

State	Angle	neighborhood	New State	New Angle
F	**0°**	a5 = R & a3 & a7 = F	R	**0°**
		a5 = RR-3 & a3 & a7 = F	**FR-0**	
		a5 = FR-0 & a3 & a7 = F	**FR-1**	
		a5 = FR-1 & a3 & a7 = F	**FR-2**	
		a5 = FR-2 & a3 & a7 = F	**FR-3**	
		Otherwise	F	
	-45°	**a5 = FRD**	**FRD**	**-45°**
		Otherwise	**R**	**0°**
	45°	**a5 = FRD**	**FRD**	**45°**
		Otherwise	**R**	**0°**
	-90°	a3 = R & a5 = F & a2 = O	R	-90°
		a3 = RR-0	RR-1	
		a3 = RR-1	RR-2	
		a3 = RR-2	RR-3	
		Otherwise	F	
	90°	a7 = R & a5 = F & a8 = O	R	90°
		a3 = RR-0	RR-1	
		a3 = RR-1	RR-2	
		a3 = RR-2	RR-3	
		Otherwise	F	
O	Any	Any	O	Any
R	0°	a1 & a2 & a8 = F	F	0°
		a1 = O & a2 & a3 = F	R	90°
		a1 & (a2 or a3) = O & a7 & a8 = F	R	-90°
		a1 & a7 = O & a3 = F	R	90°
		a1 & a3 = O & a7 = F	R	-90°
		a8 = O & a3 = F	R	90°
		a2 = O & a7 = F	R	-90°
	-90°	a1 = O & a7 & a6 = F	F	-90°
		a1 = F	RR-0	0°
	90°	a1 = O & a3 & a4 = F	F	-90°
		a1 = F	RR-0	0°
RR-i	-90°	a7 = F	F	-90°
		Otherwise (There is no path)	RR-0	0°
	90°	a3 = F	F	90°
		Otherwise (There is no path)	RR-0	0°
FR-i	**0°**	**a1 = F**	**F**	**0°**
		Otherwise	**R**	**0°**
FRD	**-45°**	**a1 & a7 & a8 = F**	**F**	**-45°**
		Otherwise	**R**	**0°**
	45°	**a1 & a2 & a3 = F**	**F**	**45°**
		Otherwise	**R**	**0°**

for each robot and requires less team communication. The modifications that we performed in the team structure decrease the master responsibilities of the leader robot and turn it into a coordinator robot. In the proposed model, during the communication process, the neighboring robots only exchange the number of steps and their ID, while the coordinator compares the information received with the formation parameters and with its own data to flag if the spatial pattern of the team is correct. The neighbor robot will receive the answer and decide locally which action it has to take and select the corresponding CA rule. Thus, the spatial pattern of the team is synchronized and the decision is taken with more autonomy by each robot.

Instead of using a step by step communication, we reduced the number of information exchanged to a periodic message at every p steps. Therefore, the robots were able to perform more actions independently and still synchronize their positions regarding the formation maintenance. Although p is a parameter of the model that can be adjusted for each scenario and application, we used $p = 3$ in all simulations and real experiments performed. The reason to use $p = 3$ is because this value presented a good trade-off between the amount of time holding formation and avoiding obstacles, regarding communication needs.

In order to implement the team communication in the real robots, we had to choose which communication technology would be the most appropriate for the e-puck robots. Despite the significant compatibility between the Webots simulator environment and e-puck platforms, the simulator does not provide cross-compatibility for robot-to-robot communication. Using the architecture developed in [17], we built the messages to be exchanged between the e-puck robots and integrated them to the Webots library. Performing this step in the simulation experiments, we could decrease the reality gap [37], since the same code developed in the simulator could be cross-compiled to the real robots.

Our first choice of communication technology was Bluetooth, as employed in the experiments reported in [12], using the Webots' libraries *emitter* and *receiver*. Bluetooth interface requires the emitter e-puck to first search for Bluetooth devices, then to identify the ones that are e-pucks, select the desired one and establish a Serial Port Profile (SPP) link. Only after all these steps are successful, the emitter e-puck can send the data. Although this is an efficient paradigm in simulation, it can not be compared to real Bluetooth communication involving e-pucks. Therefore, as a first step to analyze the robustness of this approach in real world experiments, we implemented two basic test scenarios: (*i*) the robot 1 would send a message,

and the robot 2 would execute an respective action, (*ii*) the robot 2 would send a question message, the robot 1 would send a response message and then the robot 2 would execute the action sent by robot 1(see Figure 9).

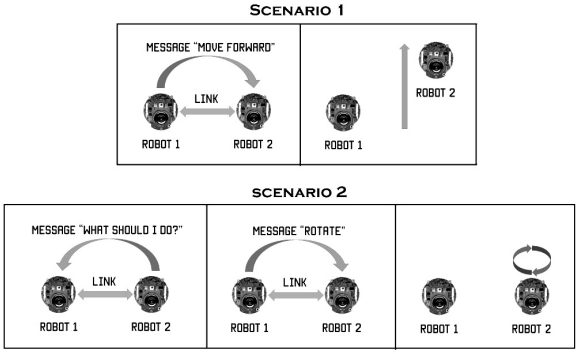

Fig. 9. Test scenarios for exchanging Bluetooth messages.

We succeeded in exchanging information in both scenarios (see video[1] and video[2]), but after a series of experiments we perceived that, despite our reduction in the frequency of messages exchanged between robots, the e-pucks Bluetooth solution may only be employed on multi-robot applications that only require sporadic communication, since exchanging a single message were taking, in average, more than 20s.

In this work we chose to use an approach that maximizes the transferability of the model, which means that the same code used in simulation is also employed in the experiments with real robots. Hence, if one develops an external Bluetooth solution and use it only in the e-puck robot, it may considerably reduce this time. However, using this reality gap approach for the task investigated here, i.e., the navigation of a robot team with a formation pattern, the standard Bluetooth was considered as unfeasible.

Aiming to achieve a direct and fast robot-robot communication, our second choice was to use an extension board on the top of the robot, called Gumstix Overo COM, which provides the typical WiFi 802.11 communication capability. Regarding the communication approach, a similar strategy to the Bluetooth technology was used, i.e., the emitter robot directly sent a message to the receiver buffer of another e-puck. Thus, the messages did not need any modification.

The same test scenarios used in simulations were applied using Wifi communication between real e-pucks. As a result, the messages exchanged were all successful received, with an average time of less than 1s. Although the use of the Gumstix Overo COM turret increased the standard e-pucks financial cost, we were able to finally accomplish the desired team communication and apply the proposed method in real world experiments.

V. METHODS

This section is intended to give an overview of the tools involved in the development of our proposed method. In our simulations, we used a Sony Vaio laptop with 6 GB of RAM memory and a Intel i5-3210M processor with 2.50 GHz. The operating system used was the 64-bit Windows 7 Home Premium.

A. The e-puck Robot

The robot we used is called e-puck and is a miniature mobile robot developed by EPFL [16] made for educational purpose. It is a open source project with low level access to electronic devices and has been used in many research papers due to its unlimited possibilities of extensions [38]. Our choice to work with the e-puck robot, was due to its many features relating to bio-inspired robotics and multi-agent experimentation.

The e-puck has an integrated dsPIC microcontroller processor with 8kB RAM, 144kB flash memory, and is supported by a custom GCC C compiler. Besides the processor, it is equipped with several sensors and actuators which enables flexible ways to interact and behave with the environment. Also, its hardware structure is considered to be simple, allowing us to include extensions. As described in the previous section, we used the Gumstix Overo COM board in order to perform the Wifi communication process. Figure 10 shows the details of the e-puck structure and the Gumstix Overo COM turret components.

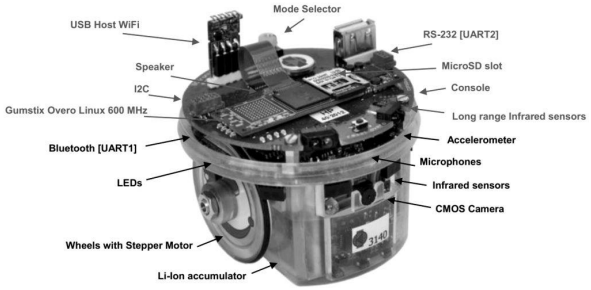

Fig. 10. System architecture of the e-puck robot equipped with Gumstix Overo COM (extension board on the top of the robot).

The speaker can emit several types of sound, the three microphones can capture and localize a sound source, the 3D accelerometer can detect position and rotation changes. The camera has a resolution of 640 x 480 and can capture images at 4 frames per second.

An important feature for robot navigation is the ability to perceive close obstacles or intensity from IR light. On the e-pucks this is done using its eight infrared (IR) sensors. However, unfortunately, as shown in Figure 11, the IR sensors are unevenly distributed around the robot, which can decrease the accuracy of the identification of nearby objects. There is a concentration on the front of the robot (sensors ps0, ps1, ps6 and ps7), which improves obstacle detection in the area. When it comes to the robot's side, the e-puck structure has only one sensor (ps2 to the right and ps5 to the left). In the rear, only two sensors are present (ps3 and ps4), thereby detecting side and rear diagonal obstacles is difficult. This hardware limitation had to be surpassed using software techniques for collision detection.

[1]http://goo.gl/YKulDm
[2]http://goo.gl/wNxYsC

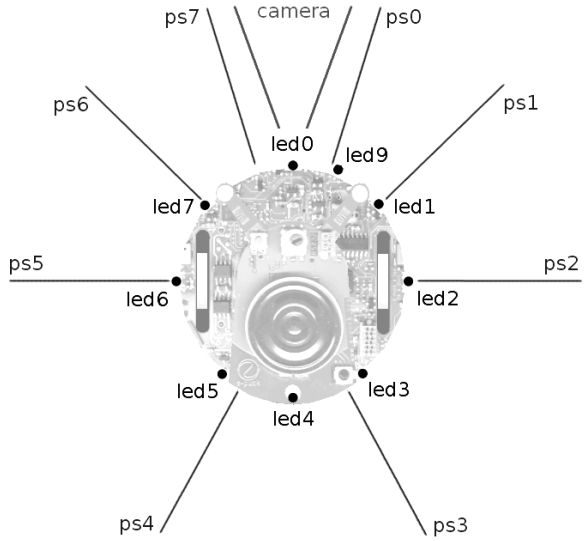

Fig. 11. IR sensors, camera and LEDs of the e-puck robot

B. Webots

The Webots simulator, shown in Figure 12, is a powerful robotic simulation platform that offers fast prototyping of mobile robots, flexible environment simulation and realistic modeling [5]. The cross-compilation option enabled us to to build a model which can be later transferred to a real robot.

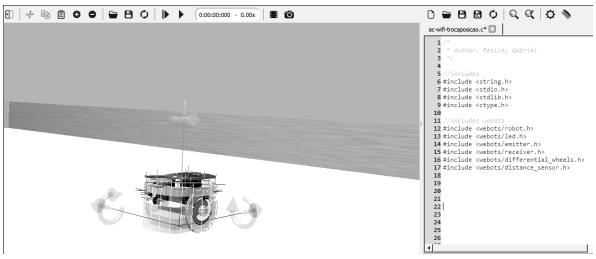

Fig. 12. Webots simulator

We used the EDU 7.4.3 version of Webots, but despite the high accuracy and simulation detail levels, the code developed in the simulator can display different behaviors when running in the real world experiments, especially when it comes to coordination algorithms for cooperative teams of robots. Therefore, it was essential to use techniques that reduced this nonconformity and validated the method with robots in different real scenarios.

C. Environments

Our experiments took place in the software environment Webots and in a physical environment. Much of the prior testing was done in Webots, while the end performance of the system was tested on the real physical robots.

1) Simulation Environment: Webots provides the flexibility of adjusting many parameters and physics configuration of the environment, like mass distribution to objects and friction. The idea is that switching from software to hardware requires little or no changes to the system.

In order to test the robustness of the method, we created scenarios containing different types of obstacles, varying from polygonal to cylindrical shapes. Also, as we have three physical e-puck robots in our laboratory, the team of robots used in simulation was composed with three robots initially aligned and separated by a 25 centimeter distance. As we can see in Figure 13 the team of robots starts in formation and must keep the line pattern while moving.

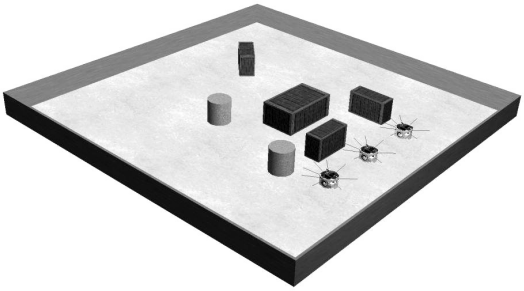

Fig. 13. Simulation scenario in the Webots simulator.

2) Physical Environment: The physical environment in the lab, presented in the Figure 14, was a 60 x 120 cm white board with 10 cm high brown walls. We also have small objects (like boxes, books and spheres) which were placed in the scenario in order to create the obstacles. The light incidence was stable, establishing good experimental conditions when testing different behaviors separately.

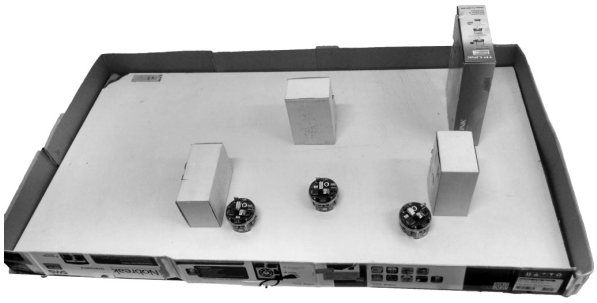

Fig. 14. Real world scenario in the our lab containing the obstacles and the team of e-pucks.

The environment can be seen as static, since it can only be changed by the robots' actions and its outcome can affect the environment for others. It is also non-deterministic and partly observable, because a given robot cant predict the next step of the system, hence is impossible for any given agent to obtain accurate, complete, up-to-date information about the environment.

VI. Experiments and Results

This section presents the simulation and real world experiments. First, we focused on individual behavior and then on the entire system performance. The obtained results are highlighted and discussed based on our expectations and previous approaches.

A. Single Robot

When it comes to testing the robot's behavior under the improved set of rules and states, first we verified the efficiency of the proposed approach in a step by step simulation, shown in Figure 15: (a) the robot moves towards the goal, (b) it identifies an obstacle, (c) it turns left to deviate, (d) the robot's lateral sensor stops detecting the obstacle, (e) it turns right, returning to the goal direction, and the obstacle is detected again by the robot's frontal sensors, (f)-(g) it applies the *RR-i* states, (h)-(j) the robots moves towards the goal using the *FR-i* states, (k) it starts the formation control maneuver to return to its original axis, (l)-(o) it moves ahead while the obstacle is detected in its side, (p)-(q) the robot uses the *FRD* state to move in diagonal steps, (r) it detected the obstacle and performed a -45°rotation, (s) the robot executes some steps ahead in order to surpass the obstacle edge, (t) it resumes the diagonal steps moving towards its original axis.

We can see that, in comparison with Figure 6, this new behavior allowed the robot to negotiate the obstacle and to perform a smoother trajectory around it, which could decrease the time needed to achieve the final goal.

After obtaining the desired outcome in the first experiment, we needed to test the robustness of the method, so we set up three different scenarios and carried out ten executions in each of them using a single robot. The obstacles' shape was designed to vary in each simulation scenario, thus providing distinct levels of difficulty in the avoidance maneuver. The goal of the robot was to avoid the obstacle and return as soon as possible to its original axis.

As we can observe in Figure 16, the trajectory performed by the proposed method presented fewer zigzag movements and rotations than the method presented in [15] (which in turn is an improvement of the methods in [12] and [14]). On average our approach had 64% less rotations and decreased in 61% the time needed to avoid the obstacle and complete the mission.

As expected, experiments performed in real world environments showed some odometry errors due to the slippery floor. However, as it can be observed in Figure 17, the results obtained were very similar to those in simulations, as depicted in Figure 16. The triangle represents the starting point where the robot initiated its trajectory and each black point in the path means that a rotation was performed. The previous method [15] made the robot almost loose its original direction, causing an increase in the number of steps and a high number of zigzag movements. In comparison, our proposed approach kept a smoother trajectory throughout the entire detour and was able to completely surpass the obstacle and to return to the original axis of navigation.

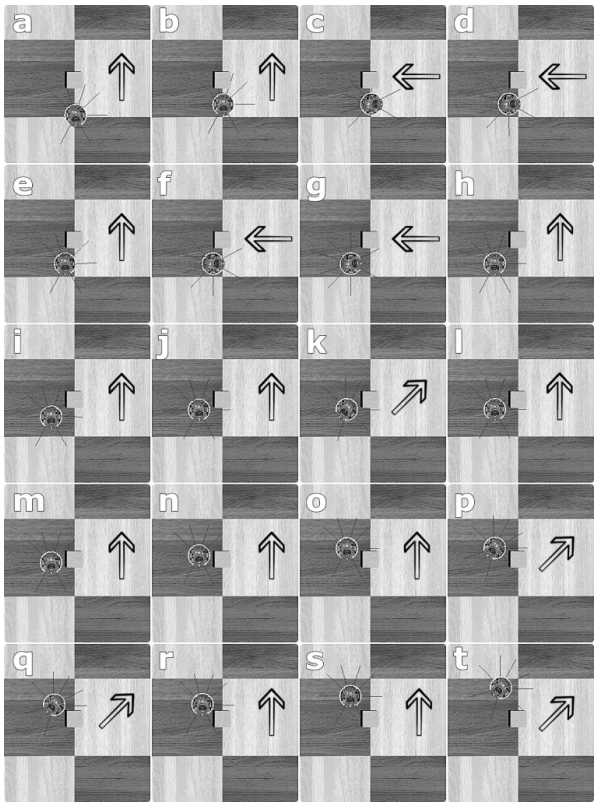

Fig. 15. The improved behavior of the robot when surpassing a corner in simulation, using the new CA states.

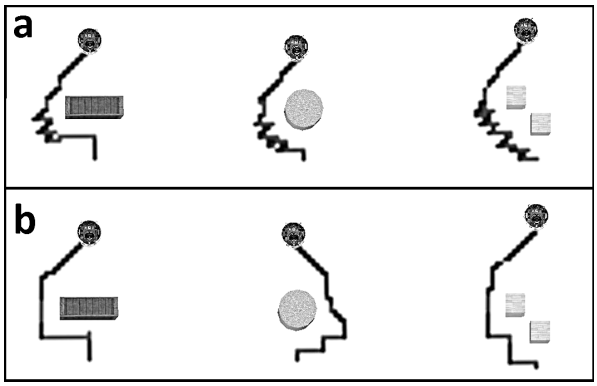

Fig. 16. Trajectory in simulation - (a) approach used by [15] and (b) proposed method.

B. Robot Team

After the experiments using a single robot, we analyzed the proposed team structure and communication using a team of three e-puck robots. It is worthy noting that in [14] and [15] there were no experiments using a team of real robots. In such experiments, the robots had to perform a strict line formation while moving towards a goal in the environment.

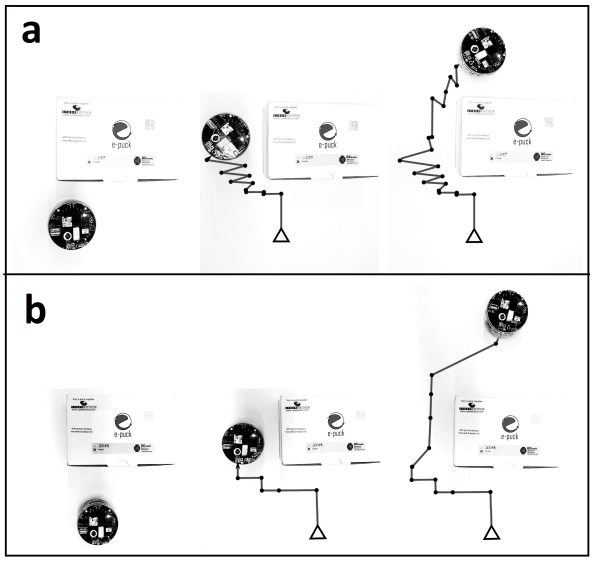

Fig. 17. Real experiments - (a) approach used by [15] and (b) proposed method.

It is noteworthy that Webots simulator does not consider any kind of delays regarding the communication process. Thus, in simulation, the avoidance and regaining of the pattern formation were achieved flawlessly in terms of communication.

As we can see in Figure 18 the robot team was able to navigate in formation, even in environments with plenty of obstacles. Using message exchanges, if a robot was considered to be late or ahead of its position, the others robots could either wait or move in order to regain the team spatial pattern.

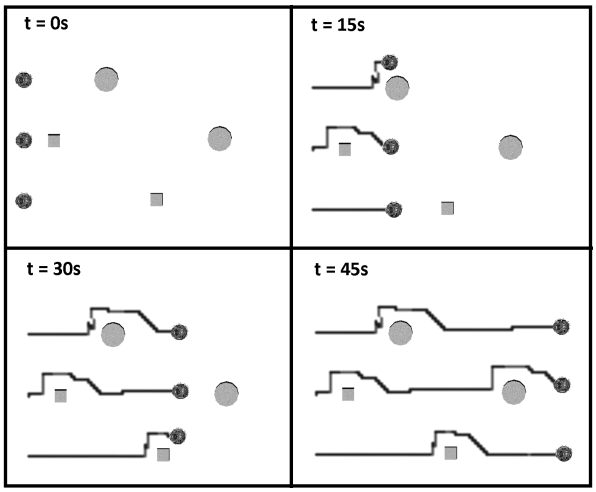

Fig. 18. Simulation results with a team of three robots.

The experimental results from the real-time system, proved the effectiveness and the robustness of the proposed CA method in creating collision-free paths. Figure 19 shows the trajectory (from left to right) performed by the team of e-pucks. At t = 0s the robots are aligned in formation. At t = 15s the two robots on the left and on the right avoid the obstacles. From t = 30s to t = 45s the robots continue to move towards the goal. At t = 60s the left robot returns to its original path, the coordinator robot and the right robot start deviating from the other obstacles. At t = 75s all the robots arrive at the final goal. The robots were able to act individually, regarding the avoidance of obstacle, and as a group, in order to keep the team formation. (The video of the whole experiment is available at [3]).

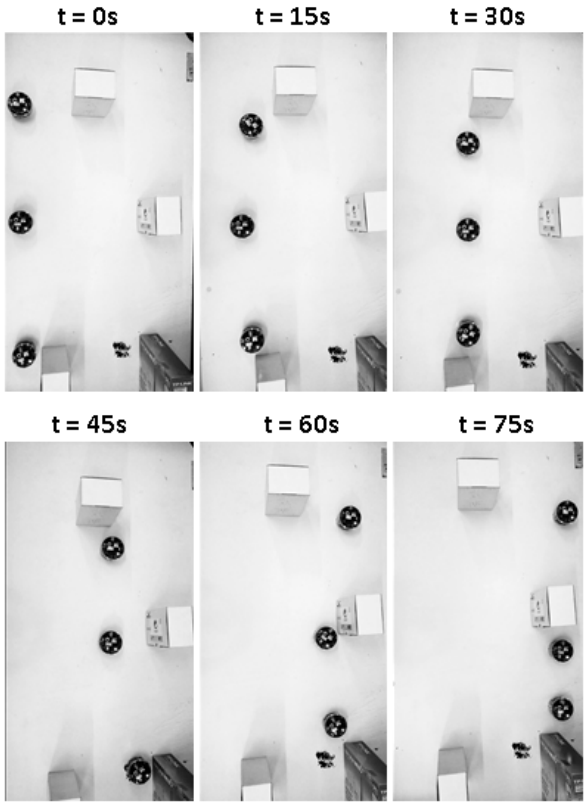

Fig. 19. Real experiments with the e-puck communicating and maintaining formation.

VII. Discussion

Although the simulation is a valuable technique for speeding up the design process, it is not enough to validade a navigation model and it is crucial to verify the behavior of the robots in physical environments. Due to the cost and effort required to perform the experiments using multiple robots, the tests in real scenarios usually presents a lack of consistency when compared to the virtual-world results. The problem is that often the simulation abstractions do not incorporate the factors that may be present in a real-world scenario. Examples may include sensor noise, ambient light control, wheel slip and limited communication infrastructure. These factors may have a significant impact on the system performance, especially for

[3]http://goo.gl/Cjz91o

highly coordinated algorithms where the complexity creates more opportunities to errors. Despite of this intrinsic difficulty, our discrete approach based on CAs allowed the performance in real-world scenarios to be quite similar to those obtained in simulation. The robots were able to act individually when performing obstacles deviations, and as group while keeping the formation pattern.

The new CA model with the addition of new states and improved set of rules, enable the robots to exhibit a more adequate behavior when performing a obstacle avoidance task. Besides, due to the more natural and smooth characteristics of the generated trajectory, the proposed method achieved a smaller odometry error, which led the robots to calculate their current positions in a more accurate way. Although the proposed improvements have increased the complexity of the model (number of CA states and transition rules), the resulting method is still simple enough to be used in robots equipped with minimal computation capabilities, such as the e-pucks.

The development of multi-robot systems, where the robots must share information, is considerably more challenging than single robot solutions. Although the Webots platform provided the communication functionally in simulation, the absence of a portable real-world library for robot-to-robot communication increased the gap between model development and its implementation in real robots. This fact was one of the obstacles to the execution of experiments in a real robot team encountered in [14] and [15]. Such deficiency was overcome in this work with the adoption of a library developed in [17]. The use of this library extended the cross-compatibility between the Webots simulator and the e-puck robotic platform, providing the communication capabilities needed to carry out the formation control experiments in a physical scenario.

Starting from several experiments employing the proposed model in the navigation of individual and e-puck teams, we could observe that a weakness of our model and other CA-based path-planning is related to the process of neighborhood construction based on the sensor readings. The interpretation of the presence or absence of an obstacle in a specific neighbor cell is based on a set of crisp rules which employs some fixed thresholds. We believe that a new process to construct the neighborhood in a more adaptive way, considering some characteristics of a noisy and dynamic environment (as the current ambient light) may be an important improvement of such kind of model.

VIII. Conclusion and Further Work

This paper proposes a novel cellular automata-based method for robot path-planning and formation control. We started by outlining the use of a CA for solving the path-planning problem and its main advantages when compared to similar approaches. We then described the new states and more elaborated rules that were used to provide efficient robot behavior, in order to accomplish the desired task. We also proposed and tested a more decentralized team structure with different communication technologies.

The results from the simulation environment and from the experiments with real robots established the effectiveness and the robustness of the new proposed method. The trajectory of the robots was optimized, and the time necessary to execute the task was decreased. The proposed approach generated more fast and accurate paths and its requirements for computational resources were minimal. Hence, it was functional even in robots equipped with small on-board memory and slow processors.

As one might expect, working with real robots presents more challenges and inconsistencies when compared to simulated environments. Although the Webots simulator provided a helpful environment and several development shortcuts, a lot of time and effort was devoted in order to oversome the reality-gap, in particular with regards to the communication between the e-pucks. Nonetheless, the implementation and test on the real robots allowed us to validate the proposed system.

As future work, we intend to design a co-evolutionary method [39], in which a genetic algorithm may be used to determine the best moment to perform a rearrangement on the team formation structure. Moreover, a new neighborhood classification system will be used to determine the adjacent cells' states based on the sensors readings on a noisy and dynamic environment.

Acknowledgment

The authors would like to thank the Robotics Lab at the School of Mathematical and Computer Sciences at Heriot-Watt University for their support in providing access to the e-puck robots used in the real experiments. CNPq, CAPES and Fapemig for funding.

References

[1] F. Soofiyani, A. Rahmani, and M. Mohsenzadeh, "A straight moving path planner for mobile robots in static environments using cellular automata," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on*, July 2010, pp. 67–71.

[2] W. Guanghua, L. Deyi, G. Wenyan, and J. Peng, "Study on formation control of multi-robot systems," in *Third International Conference on Intelligent System Design and Engineering Applications (ISDEA), 2013*, Jan 2013, pp. 1335–1339.

[3] K. Ioannidis, G. Sirakoulis, and I. Andreadis, "Depicting pathways for cooperative miniature robots using cellular automata," in *International Conference on High Performance Computing and Simulation (HPCS), 2011*, July 2011, pp. 794–800.

[4] Y.-Q. Chen and Z. Wang, "Formation control: a review and a new consideration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005). 2005*, Aug 2005, pp. 3181–3186.

[5] P. Dasgupta, T. Whipple, and K. Cheng, "Effects of multi-robot team formations on distributed area coverage," *International Journal of Swarm Intelligence Research*, vol. 2, no. 1, pp. 44–69, Jan. 2011.

[6] I. Navarro and F. Matia, "A survey of collective movement of mobile robots," *International Journal of Advanced Robotic Systems*, vol. 10, no. 73, 2013.

[7] H. Mehrjerdi, M. Saad, and J. Ghommam, "Multi mobile robots formation in presence of obstacles," in *IEEE International Conference on Mechatronics (ICM), 2011*, April 2011, pp. 510–515.

[8] I. M. Sanhoury, S. H. Amin, and A. R. Husain, "Multiple nonholonomic wheeled mobile robots trajectory tracking while maintaining time-varying formation via synchronous controller," *Procedia Engineering*, vol. 41, no. 0, pp. 1044–1050, 2012.

[9] H. Rezaee and F. Abdollahi, "A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 347–354, Jan 2014.

[10] Y. Meng, H. Guo, and Y. Jin, "A morphogenetic approach to flexible and robust shape formation for swarm robotic systems," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 25 – 38, 2013.

[11] K. Ioannidis, G. Sirakoulis, and I. Andreadis, "Cellular ants: A method to create collision free trajectories for a cooperative robot team," *Robotics and Autonomous Systems*, vol. 59, no. 2, pp. 113–127, Feb. 2011.

[12] K. Ioannidis, G. Sirakoulis, and I. Andreadis, "A path planning method based on cellular automata for cooperative robots," *Applied Artificial Intelligence*, vol. 25, no. 8, pp. 721–745, Sep. 2011.

[13] Cyberbotics. (2014) Webots 7: Robot simulator. [Online]. Available: http://www.cyberbotics.com/overview

[14] G. Ferreira, P. A. Vargas, and G. Oliveira, "An improved cellular automata-based model for robot path-planning," in *Advances in Autonomous Robotics Systems*, ser. Lecture Notes in Computer Science, M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, Eds. Springer International Publishing, 2014, vol. 8717, pp. 25–36.

[15] G. Oliveira, P. A. Vargas, and G. Ferreira, "A local decision making cellular automata-based path-planning," *11th National Meeting on Artificial and Computational Intelligence (ENIAC2015), BDBComp, Brazilian Computer Society*, 2015. [Online]. Available: http://www.lbd.dcc.ufmg.br/colecoes/eniac/2014/00102.pdf

[16] Ecole Polytechnique Federale de Lausanne. (2014) E-puck education robot. [Online]. Available: http://www.e-puck.org

[17] M. S. Couceiro, P. A. Vargas, and R. P. Rocha, "Bridging the reality gap between the webots simulator and e-puck robots," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1549 – 1567, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889014000967

[18] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of Modern Physics*, vol. 55, pp. 601–644, Jul 1983. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.55.601

[19] Y. Fu, H. Li, and Y. Ma, "Path planning of cooperative robotics and robot team," in *IEEE International Conference on Robotics and Biomimetics, 2006. ROBIO '06.*, Dec 2006, pp. 1250–1255.

[20] J. Fredslund and M. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 837–846, Oct 2002.

[21] S. S. Ge and C.-H. Fua, "Queues and artificial potential trenches for multirobot formations," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 646–656, Aug 2005.

[22] M. Defoort, T. Floquet, A. Kokosy, and W. Perruquetti, "Sliding-mode formation control for cooperative autonomous mobile robots," *Industrial Electronics, IEEE Transactions on*, vol. 55, no. 11, pp. 3944–3953, Nov 2008.

[23] D. Edwards, T. Bean, D. Odell, and M. Anderson, "A leader-follower algorithm for multiple auv formations," in *Autonomous Underwater Vehicles, 2004 IEEE/OES*, June 2004, pp. 40–46.

[24] Z. Wang and D. Gu, "Cooperative target tracking control of multiple robots," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 8, pp. 3232–3240, Aug 2012.

[25] J. Cook and G. Hu, "Experimental verification and algorithm of a multi-robot cooperative control method," in *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, July 2010, pp. 109–114.

[26] C.-C. Lin, P.-Y. Hsiao, and K.-C. Chen, "A motion planning of swarm robots using genetic algorithm," in *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, Nov 2010, pp. 538–543.

[27] A. N. Asl, M. B. Menhaj, and A. Sajedin, "Control of leaderfollower formation and path planning of mobile robots using asexual reproduction optimization (aro)," *Applied Soft Computing*, vol. 14, Part C, pp. 563 – 576, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613002883

[28] G. Capi and Z. Mohamed, "Multiple robots formationa multiobjctive evolution approach," *Procedia Engineering*, vol. 41, pp. 156 – 162, 2012, international Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705812025428

[29] G. Lee and N. Y. Chong, "Decentralized formation control for small-scale robot teams with anonymity," *Mechatronics*, vol. 19, no. 1, pp. 85 – 105, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957415808000871

[30] C. Shu and H. Buxton, "Parallel path planning on the distributed array processor," *Parallel Computing*, vol. 21, no. 11, pp. 1749 – 1767, 1995.

[31] F. Marchese, "A reactive planner for mobile robots with generic shapes and kinematics on variable terrains," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, July 2005, pp. 23–30.

[32] F. Marchese, "Time-invariant motion planner in discretized c-spacetime for mrs," *Multi-Robot Systems, Trends and Development, Toshiyuki Yasuda (Ed.)*, 2011.

[33] P. Tzionas, A. Thanailakis, and P. Tsalides, "Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 2, pp. 237–250, Apr 1997.

[34] A. Akbarimajd and C. Lucas, "A new architecture to execute cas-based path-planning algorithm in mobile robots," in *Mechatronics, 2006 IEEE International Conference on*, July 2006, pp. 478–482.

[35] A. Akbarimajd and A. H. Zadeh, "A novel cellular automata based real time path planning method for mobile robots," 2011.

[36] A. Rosenberg, "Cellular antomata," *Parallel and Distributed Processing and Applications*, pp. 78–90, 2007.

[37] S. Koos, J.-B. Mouret, and S. Doncieux, "The transferability approach: Crossing the reality gap in evolutionary robotics," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 1, pp. 122–145, Feb 2013.

[38] J. Berg and C. Karud, "Swarm intelligence in bio-inspired robotics," Master's thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, 2011.

[39] P. A. Vargas, E. A. Di Paolo, I. Harvey, and P. Husbands, *The Horizons of Evolutionary Robotics*, 1st ed. The MIT Press, 2014.