CIÊNCIA DA COMPUTAÇÃO

CAMINHOS DE COBERTURA ADAPTÁVEIS PARA ROBÔS ASPIRADORES

VICTOR HEIDI OLIMPIO OTTO

Rio Verde, GO
2021

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO – CAMPUS RIO VERDE

CIÊNCIA DA COMPUTAÇÃO

CAMINHOS DE COBERTURA ADAPTÁVEIS PARA ROBÔS ASPIRADORES

VICTOR HEIDI OLIMPIO OTTO

Trabalho de Curso apresentado ao Instituto Federal Goiano – Campus Rio Verde, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Márcio Antônio Ferreira Belo Filho

Rio Verde – GO

Julho, 2021

RESUMO

OLIMPIO OTTO, Victor Heidi. **Caminhos de cobertura adaptáveis para robôs aspiradores**. 2021. 66p Monografia (Curso de Ciência da Computação). Instituto Federal de Educação, Ciência e Tecnologia Goiano – Campus Rio Verde, Rio Verde, GO, 2021

Um robô aspirador pode apresentar muitas vantagens para a rotina de pessoas usualmente ocupadas. Para a sua funcionalidade, é necessário que o caminho percorrido pelo robô cubra toda a superfície disponível para limpeza, conhecido na literatura como problema de cobertura de caminhos. Neste trabalho, o robô aspirador não possui o conhecimento prévio da área, deve se adaptar à arquitetura e mobília da casa por meio de sensores e busca um menor tempo de percurso para limpar todo o espaço. Propomos algoritmos capazes de fazer uma varredura completa com resposta eficiente a mobílias e obstáculos, por meio de uma estratégia de mini caminhos onde o robô gera diversos caminhos possíveis a sua volta, promovendo adaptabilidade ao percurso do robô aspirador. Posteriormente, com o conhecimento da área completa, os percursos podem ser aperfeiçoados através de um algoritmo genético ou *Fix and optimize*. Os algoritmos propostos são comparados com os algoritmos de cobertura por caminhos da literatura para um conjunto de instâncias. A estratégia proposta teve boa performance em comparação a outras abordagens da literatura.

Palavras-chave:Robô aspirador, Inteligência Artificial , Problema de cobertura por caminhos.

ABSTRACT

A robot vacuum cleaner can have many advantages for the routine of people who are usually busy. For its functionality, it is necessary that the path taken by the robot covers the entire surface available for cleaning, known in the literature as path coverage problem. In this work, the robot vacuum cleaner does not have prior knowledge of the area, it must adapt to the architecture and furniture of the house through sensors and it aims a shorter time to clean the entire space. We propose algorithms capable of performing a coverage path with efficient response to furniture and obstacles, through minipaths, where the robot generates several possible paths around it according to its sensors, promoting adaptability to the path of the robot vacuum cleaner. Later, with the knowledge of the complete area, paths are improved by genetic algorithms and Fix and optimize approaches. The proposed algorithms are compared to the path coverage algorithms from the literature for a set of instances. The proposed strategy performed well compared to literature approaches.

Keywords:Robot vacuum cleaner, Artificial Intelligence, Path coverage problem.

Lista de ilustrações

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SÍMBOLOS

CCP	Complete Coverage Path
STC	Spanning Tree Coverage
GRASP	Greedy Randomized Adaptive Search Procedure
BSA	Back Tracking Algorith
TASP	Turn-away Starting-point
GAYA	Genetic Algorith Yakoubi
NNY	Neural Network Yakoubi
PTZ	Path Transform Zelinsky
STC	Spanning Tree Coverage
WES	Wavefront Survey
YCM	Young-ho Choi Matrix

SUMÁRIO

1 INTRODUÇÃO

A presença de robôs aspiradores em casa está se tornando cada vez mais frequente, e sua demanda aumenta a cada dia. Segundo VELOSO (2020), a venda de robôs aspiradores aumentou em 802% em abril de 2020, em relação ao ano anterior, gerando uma demanda muito grande em limpeza automatizada nas casas. Deste modo, diversas pesquisas estão sendo desenvolvidas para criar um robô capaz de executar toda limpeza de uma casa, de modo que limpe a maior área possível e com rapidez, para todo tipo de residência. Um grande desafio é desenvolver uma inteligência artificial que faça a limpeza com estas características.

Para originar uma inteligência artificial capaz de fazer a limpeza da casa, deve-se implementar um planejamento de cobertura por caminhos. O planejamento é o que define o comportamento do robô durante a limpeza da casa. Nos casos mais simples, o robô pode ser instruído a fazer os movimentos/estratégias: seguindo a parede, espiral, zigue-zague e aleatórios. No entanto, utilizar apenas uma das estratégias pode não apresentar boas soluções para o problema. São necessárias as informações do ambiente obtidas dinamicamente para se estabelecer uma boa rota de varredura. Deste modo, existem modelos de robôs mais avançados que apresentam meios de reconhecimento da área, para planejar os movimentos, evitar repetições e impedir colisões com a parede ou móveis.

Com o objetivo de o robô aspirador ser mais eficiente na limpeza da residência, ele deve apresentar um método de lidar com os diversos tipos de mobílias que servirão como obstáculos durante seu percurso. Deste modo, como normalmente o robô não tem o conhecimento prévio da área, é preciso diversos sensores para ter a percepção dos objetos ao seu redor e fazer o mapeamento do local durante o percurso. Um método inteligente é fazer esse reconhecimento enquanto faz uma varredura eficiente nos locais já reconhecidos, um ritmo que dever ser seguido até o fim da limpeza.

No entanto, uma aplicação que faça total limpeza de uma sala com mobílias não é o único problema de se resolver, o planejamento também deve considerar o tempo de limpeza, criando trajetórias que sejam relativamente rápidas de serem realizadas, ou seja, evitar fazer muitas curvas ou retornos desnecessários. Por conta disso, diversos trabalhos da área de Planejamento de cobertura completa (PCC) demonstraram métodos para fazer uma varredura completa, porém, muitos apresentaram problemas para encontrar o percurso mínimo para a cobertura completa, e/ou evitar repetir alguns caminhos.

O intuito deste trabalho é implementar um algoritmo de PCC que tenha a capacidade de adaptar seu caminho durante o percurso. Inicialmente o método de

cobertura será feito com uso de mini caminhos, com base em 3 algoritmos: Guloso, GRASP (*Greedy Randomized Adaptive Search Procedure*, (FESTA; RESENDE, 2002)) e Árvore, em que a diferença é a forma de escolher os mini caminhos. Posteriormente, após a primeira cobertura completa, também terá o aperfeiçoamento com base de parâmetros e das abordagens: algoritmo genético e *Fix and Optimize*, para ajustar os caminhos, a fim de tentar chegar ao melhor tempo de cobertura completa. Por fim, os algoritmos implementados serão abordados em um conjunto de instâncias, com o propósito de observar sua eficácia em relação aos algoritmos da literatura.

Este trabalho está dividido em diversas seções para mostrar uma coletânea de inteligências dos robôs aspiradores, e propor um novo método de realizar a cobertura completa. A Seção de Revisão da Literatura tem diversos trabalhos que atuam na área de cobertura e mostrar quais inovações eles trouxeram para a área, além da distinção entre algoritmos online e offline. A Seção Algoritmos da Literatura, já possui mais aprofundamentos em alguns dos algoritmos e qual sua metodologia para realizarem as varreduras. Em seguida, a Seção Proposta contém toda explicação de como o algoritmo proposto deve funcionar e quais suas qualidades. Na seção de resultados, são apresentados e discutidos todas as tabelas com os dados e saídas dos algoritmos implementados. Por fim, a conclusão, onde destacamos alguns pontos do trabalho e discorremos algumas ideias futuras para o problema e os métodos desenvolvidos.

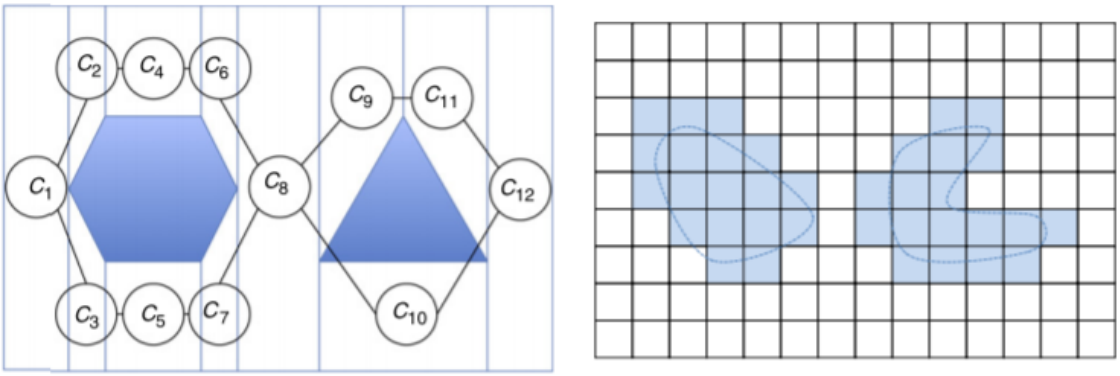
2 REVISÃO DE LITERATURA

Para compreender o planejamento de um robô aspirador, diversas metodologias de cobertura por caminhos foram pesquisadas a fim de conhecer os procedimentos necessários para fazer a cobertura completa da área de modo eficiente.

A fim de estudar os algoritmos de cobertura por caminhos, dois artigos de revisão sobre complete coverage path (CCP) para robôs: os trabalhos de GALCERAN (2013) e CHOSET (2001) demonstram uma coletânea dos desenvolvimentos desta área, com diversas abordagens em técnicas e mapeamento da área.

Para o planejamento de caminhos do robô é preciso passar o conhecimento da área para o robô, com uso de sensores, e/ou conhecimento prévio da área em questão. No melhor caso seriam áreas retangulares e sem obstáculos presentes, porém este caso é minoria nas residências domésticas, onde existem diversas mobílias e outros obstáculos. Por conta disso, existem diversas metodologias para considerar esses obstáculos enquanto o local é mapeado, onde o robô transforma esta área em grafos ou matriz, com o propósito de dividir para conquistar, o ambiente da casa é dividido em áreas menores, livre de obstáculos, tornando mais fácil a limpeza.

Figura 1 – Ambientes tratados em grafo e em matriz.



Fonte: (GALCERAN, 2013)

Com o propósito de efetuar a transformação em grafos, utiliza-se um procedimento de decomposição celular, onde divide-se uma área em regiões menores com vértices e arestas. O robô com base de sensores, capta pontos de referências dos obstáculos, criando um grafo, e suas áreas formadas são chamadas de células. Essas células normalmente são áreas sem obstáculos, apenas com arestas que servem para delimitar a região. Este método forma polígonos ao redor de obstáculos que seriam trabalhosos fazer um percurso para abrangê-lo.

Para a transformação em matriz, o robô passa pelo processo de discretização espacial em matriz, onde com auxílio de sensores são estipulados diversos quadrados com tamanho fixo ao seu redor. O robô pode determinar um valor para cada

quadrado dependendo se a área tem um obstáculo ou não. Ao fim, conforme a varredura acontece, é construída uma matriz composta de linhas e colunas destes quadrados, e na conclusão da cobertura, o robô tem o mapa completo com as áreas limpas e objetos da sala.

O modelo de matriz é muito utilizado em áreas planas e fácil de ser manipulado pelo algoritmo, além de trazer resultados a curto prazo. No entanto, conforme aumenta o tamanho da matriz trabalhada, consequentemente aumenta o tamanho do percurso para cobertura completa, o que pode ocasionar em problemas de odometria, onde a precisão decai conforme os acúmulos de pequenos desvios durante o trajeto, causando erros de percurso.

Para este trabalho, foi utilizado o modelo de discretização espacial em matriz para o desenvolvimento do algoritmo proposto. Ao lidar com residências, não há a preocupação de tratar áreas muito grandes, e dificilmente chegará a um ponto de problema de odometria.

Com o mapeamento de uma área, o algoritmo precisa ter um planejamento de cobertura por caminhos para que limpe toda a sala em maior área possível com um tempo aceitável. Um planejamento simples deve utilizar um padrão de movimento que consiga cobrir uma grande área utilizando movimentos repetidos, como caminho em zigue-zague, e aplicar um algoritmo correto, para que o robô faça este padrão de movimento gere uma varredura completa.

No entanto, para um planejamento mais eficiente, deve incluir mais padrões de movimentos, no caso de uma residência, necessitaria de um movimento que faça a limpeza no entorno dos objetos. Desta forma, existe um método com uso de sensores para auxiliar a ficar o mais próximo do objeto sem ter o contato, enquanto faz a limpeza ao seu redor (HERT, 1996). Assim, é possível expandir a área de cobertura, e chegar a um resultado mais eficiente.

Utilizar mais de um padrão de movimento é bastante comum nas metodologias de planejamento de caminhos. O caminho mais simples, o zigue-zague, tem a maior eficiência em limpar em relação aos outros movimentos simples como: espiral, movimentos aleatórios e o percurso de seguir a parede. No entanto, o trabalho de HASAN (2014) já demonstrou que mesmo levando maior tempo de limpeza, obteve uma maior área limpa quando combinou estes 4 estilos de movimentos em uma sala.

Para o desenvolvimento dos padrões de movimentos, existem 2 grupos de algoritmos principais que definem o comportamento do robô, nos quais diferenciam como são aplicados os movimentos. GALCERAN (2013) classifica os algoritmos em dois grupos: online e offline. Ambos tem a função de fazer a cobertura completa por caminhos, mas possuem tratamentos de dados diferentes para chegar a esse objetivo. Suas características são:

1. Offline: O mais comum entre as metodologias, possui o conhecimento prévio da área que será trabalhada, e normalmente não tem adaptação no planejamento dos caminhos durante o percurso, ou seja, se algo inesperado acontecer durante o percurso, o robô não tem meio de replanejar o caminho já decidido antes da partida. No entanto, ainda pode ter métodos de correção para que ele ainda siga o caminho com precisão, com uso de sensores e ajustes nos movimentos.

   Apresentam os modelos mais eficientes, são mais fáceis de serem desenvolvidos, porém, em caso de distorção do ambiente, eles são menos confiáveis. Normalmente são a base de um projeto inicial, para assim, passar para o método online.

2. Online: Mais difícil de implementar e serem eficientes, envolve sensores e algoritmos mais complexos, não possui um conhecimento prévio da área, tem a capacidade de reconhecimento e adaptação de caminho, ou seja, o robô pode construir o caminho conforme faz o seu percurso, e tem a possibilidade de sempre replanejar o caminho em caso de obstáculos inesperados.

   São mais dependentes de sensores para um mapeamento eficiente, porém, possuem os melhores resultados em instâncias mais próximas à realidade.

Estes algoritmos também apresentam distinções nas áreas que são abordados, portanto, é preciso conhecer as características como o seu terreno, dimensões e obstáculos. Deste modo, é preciso formar algoritmos para cada área de atuação, a fim de não entrar em conflito com metodologia em áreas distintas. Para este trabalho, serão abordadas principalmente as técnicas de planejamento de caminho para limpeza de residências urbanas, ou seja, áreas planas, com mobílias como obstáculos.

Para os seguintes tópicos, serão abordados os algoritmos de outros atores que serão a base para o desenvolvimento do algoritmo proposto. Inicialmente será apresentada a ideia principal de como cada um faz a cobertura completa, mais detalhes sobre suas metodologias estarão na Seção 3.

O modelo de *wavefront* proposto por ZELINSKY et al. (1993) é um algoritmo offline de CCP que faz a cobertura mesmo que tenha obstáculos, com o uso de matriz e transformação de distância. Neste algoritmo o usuário pode escolher o ponto de saída e o de chegada para o robô, pois o *wavefront* adiciona valores nas casas adjacentes à chegada na matriz, e para as casas adjacentes seguintes, vai incrementando valores maiores até o preenchimento total da matriz. O robô começa sempre escolhendo o número maior ao redor, até que faça toda a varredura e termine na chegada. No entanto, este modelo em maioria das instâncias, não traz um caminho ótimo, já que pode fazer repetições de passos durante seu percurso.

Como o desenvolvimento do algoritmo proposto será online, serão abordados diversos algoritmos online para apresentar metodologias de cobertura por caminhos sem o conhecimento prévio da área, visto que são os modelos mais confiáveis atualmente. Deste modo, o robô é coberto de sensores infravermelhos ou ultrassônicos, que determinam uma região de conhecimento para ser trabalhado, que é a região de alcance do sensor. Definida esta região, existem diversos algoritmos que determinam regras para o comportamento do robô dependendo do que é captado pelos sensores.

Um algoritmo online eficiente que não repete passos é o modelo de cobertura baseado em abrangência da árvore, algoritmo Spanning Tree Coverage (STC) (GABRIELY; RIMON, 2002). O robô faz discretização espacial em matriz da área de trabalho em células quadradas do tamanho de dois diâmetros do robô, e considera toda célula parcialmente preenchida com mobília como obstáculo, ao fim, é construído uma matriz composta de células. O tamanho de dois diâmetros se torna vantajoso para fazer o planejamento com nenhuma repetição de caminho, modelo eficiente para trabalhos que não pode repassar por um mesmo lugar. Porém, tem dificuldade em lidar com células parcialmente preenchidas, ou seja, quando há obstáculo nas células, e por mais que autor tenha proposto uma versão do algoritmo para a cobertura destas células, o procedimento é obrigado a repetir alguns passos para uma varredura completa.

Um modelo online que apresenta uma grande capacidade de adaptação, é o uso de algoritmo genético para o planejamento de caminho, onde utiliza a base de mutação e *crossover* (YAKOUBI; LASKRI, 2016). Esta técnica envolve melhorar a eficiência do caminho durante o percurso, onde na área de conhecimento do robô, dentro do alcance dos sensores, são selecionados diversos caminhos para visualizar a eficiência de cada um com base de uma função de avaliação, e percorrer aquele com maior valor. Além dos caminhos escolhidos, também são feitas as variações com base de mutação e *crossover*, gerando caminhos semelhantes, os quais também são comparados para tentar encontrar um valor ainda melhor na fórmula. Portanto, mesmo que os objetos da área mudem de posição durante o percurso, o robô sempre adaptará seus caminhos até cobertura completa, e entregar um resultado satisfatório.

Existem situações em que a repetição acaba sendo inevitável, seja acabando em um beco sem saída, ou existirem múltiplas salas e um único corredor, para este tipo de problema, o robô deve conter um método de voltar para as áreas ainda não limpas, chamado de *Backtracking*. Este método consiste em encontrar e criar um caminho curto até a área não limpa, um exemplo eficaz é fazer o robô armazenar as posições ainda não limpa na memória, percorrer o caminho mais curto até um destes espaços, e continuar com a varredura (GONZALEZ et al., 2005). Desta forma, o robô sempre vai buscar todas as áreas ainda não limpas que são conhecidas, e

completando a varredura mesmo que tenha que repetir alguns passos.

No desenvolvimento do algoritmo é importante otimizar o tempo do percurso, para ser o mínimo possível. Desta forma, técnica para aperfeiçoar o movimento, como o giro do robô, pode poupar muito tempo, pois o giro exige a parada do robô para fazer as viradas durante o percurso. O trabalho de HUANG (2001) propôs uma forma de evitar essas paradas, que é replanejar a área de trabalho para fazer curvas, retirando a necessidade de girar e consequentemente parar, deste modo, o robô fica sempre em movimento e com o tempo otimizado.

Outro modelo que busca otimizar o tempo, é sempre buscar fazer o robô fazer menos curvas possíveis, realizando a cobertura com caminhos retos o máximo possível, como já proposto pelo trabalho de MITSCHKE; UCHIYAMA; SAWODNY (2018), o qual propõe um método que faz diversas verificações para sempre buscar os caminhos retos mais longos possíveis durante o seu percurso, buscando sempre realizar a cobertura completa com movimentos mais rápidos. O trabalho também demonstra uma tabela com os tempos e energias necessárias para realizar os movimentos do robô, sendo eles: aceleração, giro e desaceleração.

Uma última observação, é a orientação inicial do robô, pois em diversos trabalhos de CCP utiliza como padrão de movimento, o zigue-zague, portanto, o robô deve estar alinhado corretamente para que siga as paredes horizontais de modo, paralelas a elas. Desta forma, é importante que o robô faça esse alinhamento antes de iniciar o percurso, pelo trabalho de CHOI et al. (2009) ele demonstrou um método fácil e eficaz deste alinhamento, apenas seguindo uma parede próxima e conseguindo ter o alinhamento horizontal, para que a cobertura em zigue-zague fique correta em relação à sala e maximize a área de limpeza.

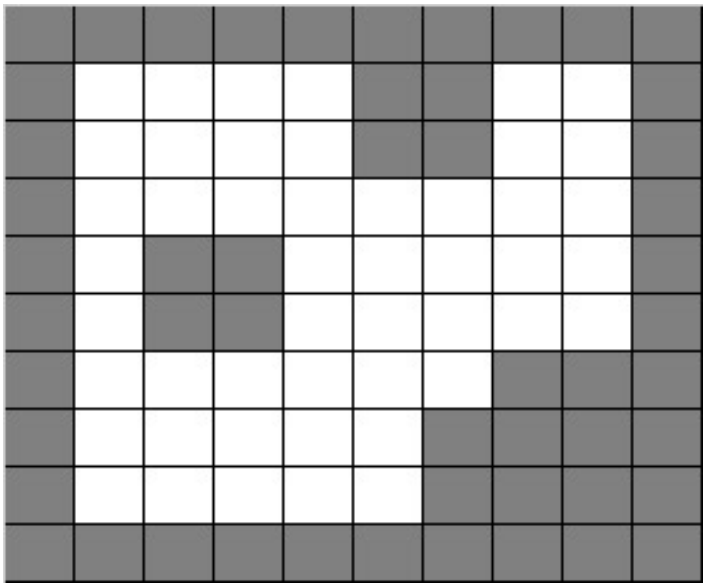
Para melhor visualização dos trabalhos estudados, foi construída uma tabela, que contém o método que é abordado, autor, algoritmo online/offline e ambientes tratados que é como o robô armazena os dados em relação a área de atuação.

Trabalhos de caminho de cobertura completo pesquisados			
Método	**Referência**	**On/Off-line**	**Ambientes tratados**
Wavefront	(ZELINSKY et al., 1993)	Off-line	Discretização espacial em matriz
Algoritmo STC	(GABRIELY; RIMON, 2002)	On-line	Discretização espacial em matriz
Cobertura com algoritmo genético	(YAKOUBI; LASKRI, 2016)	On-line	Discretização espacial em matriz
Cobertura espiral com *backtracking*	(GONZALEZ et al., 2005)	On-line	Discretização espacial em matriz
Cobertura espiral com correção de alinhamento	(CHOI et al., 2009)	On-line	Discretização espacial em matriz
Cobertura com busca de caminhos longos	(MITSCHKE; UCHIYAMA; SAWODNY, 2018)	On-line	Discretização espacial em matriz
Otimização de percurso	(HUANG, 2001)	On-line	Poligonal
Cobertura com sensor de contato	(HASAN, 2014)	On-line	Sensorial

3 ALGORITMOS DA LITERATURA

Essa seção demonstra o planejamento de caminho dos algoritmos que serão abordados neste trabalho em uma instância padronizada. A instância escolhida foi de uma sala (Figura 2) com alguns cômodos que servirão como obstáculos durante o percurso.

Figura 2 – Instância criada.



Fonte: Próprio autor

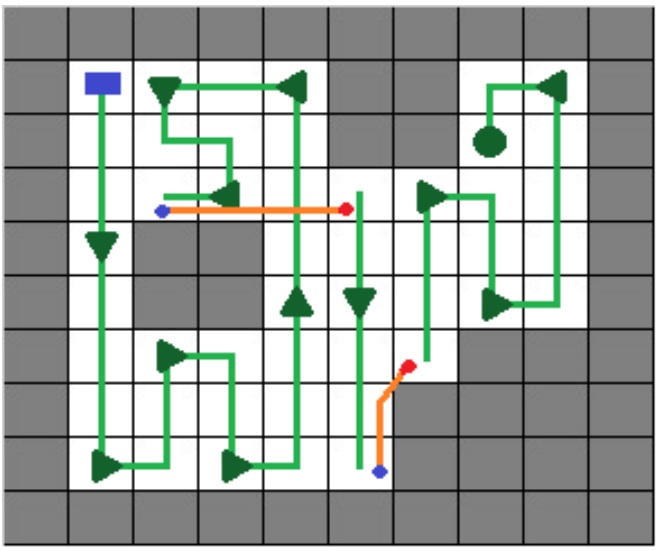
Para a demonstração dos caminhos construídos, alguns símbolos devem ser explicados antes. O início do percurso é marcado com um retângulo azul, o final com um círculo verde e caminhos em verde são o percurso principal do algoritmo. O caminho laranja é feito pelo *backtracking* começando pelo ponto azul e terminando no ponto vermelho. A técnica de *backtracking* será aprofundada em um tópico futuro, e consiste em buscar sempre o espaço não limpo mais próximo e direcionar o robô até este espaço.

3.1 Algoritmos online

O padrão de movimento mais comum entre os algoritmos online e offline é o de zigue-zague, deste modo, será o primeiro implementado pois serve de base para diversos algoritmos de planejamento. Neste algoritmo, inicialmente escolhe alguma direção para seguir a varredura, e prioriza sempre o caminho mais longo no começo do percurso, com intenção de fazer o zigue-zague mais longo possível e diminuir o

número de curvas. Este procedimento inicial é repetido toda vez que não é possível continuar em zigue-zague na direção escolhida, ou quando termina um *backtraking*, até que conclua a limpeza completa.

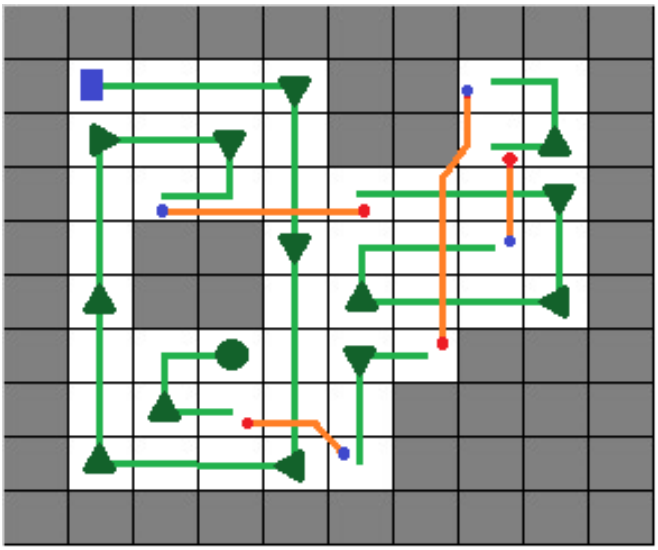
Figura 3 – Planejamento de caminho com zigue-zagues.



Fonte: Próprio autor

Outro modelo bastante utilizado é o planejamento de cobertura em formato espiral. O algoritmo tem o intuito de fazer seu caminho em espiral no sentido horário, onde as regras se resumem em sempre tentar virar para a direita quando tiver espaço livre ao colidir com um obstáculo ou parede, se não ele vira para esquerda ou faz *backtraking*, caso ambos lados já estejam limpos ou com obstáculos.

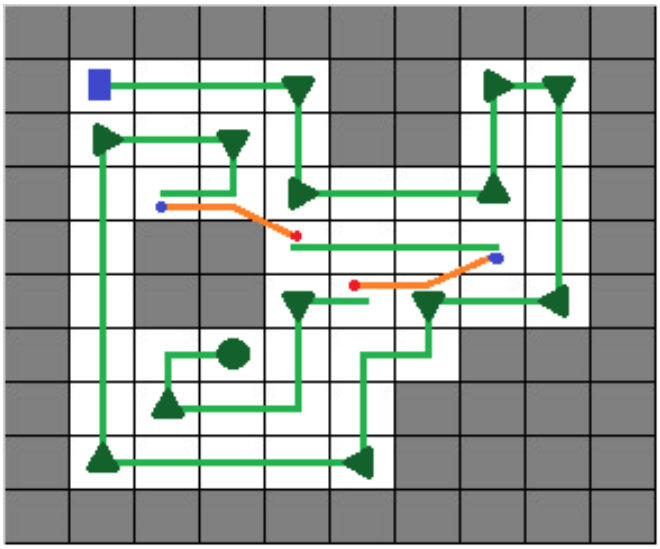
Figura 4 – Planejamento de caminho com espirais.



Fonte: Próprio autor

O algoritmo de Gonzalez (2005) segue uma regra de sempre seguir a parede da esquerda, nomeado como LeftHand, e vai acompanhando até que não exista mais espaços vazios para limpar, e também apresenta em seu algoritmo o *backtraking* para o local mais próximo ainda não preenchido, e traçar o caminho mais curto até ele.

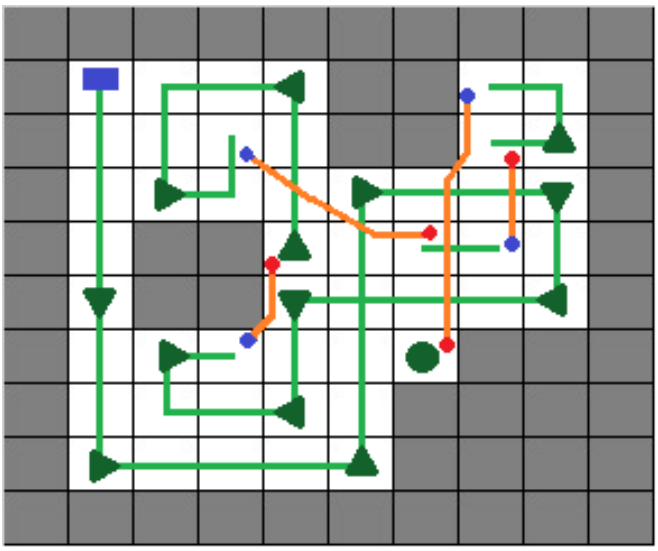
Figura 5 – Planejamento de caminho LeftHand de GONZALEZ et al. (2005).



Fonte: Próprio autor

O algoritmo de Mitschke (2018) tem o princípio de sempre escolher o caminho reto que leve o mais longe do ponto inicial, nomeado como LongestPath, e caso tenha uma colisão com a parede, a decisão de virar para algum sentido é feito com uso de sensores para medir qual caminho terá maior distância do ponto inicial do percurso com o fim do trajeto daquela virada. Também possui algumas regras para evitar curvas desnecessárias, como: se for possível atravessar até 1 local já limpo até um próximo não limpo, ele atravessa, evitando fazer curvas e passar para outras áreas.

Figura 6 – Planejamento de caminho LongestPath de MITSCHKE; UCHIYAMA; SAWODNY (2018).



Fonte: Próprio autor

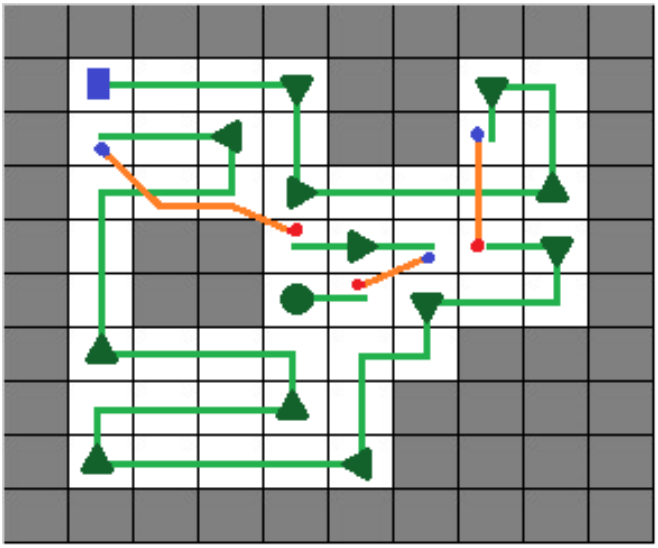
3.2 Algoritmos offline

O modelo *wavefront* utiliza por meio de ondas que vai adicionando valores ao redor do ponto de chegada, e esses valores aumentam a cada onda até completar a matriz. Neste algoritmo, as casas adjacentes são adicionados um valor 3 e para as diagonais um valor 4, para cada onda esses valores passa adiante e incrementa as próximas casas, sempre em 3 ou 4 para cada onda.

Figura 7 – Demonstração das ondas formadas pelo *wavefront*, distintas pelas cores verde e azul.

O caminho é feito seguindo sempre o maior valor ao redor disponível, até que faça a varredura completa do local, o local que o "0" é escolhido pelo usuário com a tendência de ser um dos últimos lugares que o robô vai passar durante seu percurso, mas não garante que ao chegar nele terá uma varredura completa. O algoritmo também traz a opção de encontrar o caminho mais curto de qualquer local até chegar no "0", onde será tratado mais tarde na seção *backtracking*.

Figura 8 – Planejamento de caminho com *wavefront* de ZELINSKY et al. (1993).

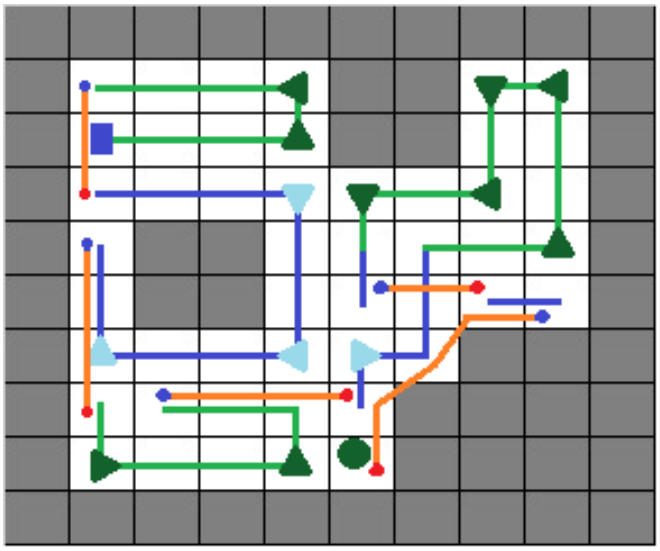


Fonte: Próprio autor

O planejamento de spanning tree coverage (STC) offline existe uma condição de sempre exigir 4 espaços vazios para formar um quadrado maior para que seja feita a abrangência da árvore, deste modo, esse algoritmo tem complicações para lidar com salas pequenas ou com muito obstáculos. Normalmente é utilizado com outros algoritmos para completar a varredura da área, neste exemplo, para completar a varredura, utilizamos o algoritmo de espiral para passar e varrer os cantos que o STC não alcança, onde é expresso em azul. Sua posição inicial também é alterada, por conta do algoritmo buscar sempre um caminho reto para iniciar e evitar fazer curva na árvore inicial.

A abrangência da árvore é feita com planejamento para o robô ter espaço para um caminho de ida e de volta, sem a necessidade de repetir algum espaço e conseguir voltar ao ponto de partida após a cobertura completa da área. No entanto, em locais apertados ou com muitas mobílias, pode ocasionar em diversas árvores, necessitando de um *backtracking* que o leve de uma árvore a outra.

Figura 9 – Planejamento de caminho com GABRIELY; RIMON (2002).



Fonte: Próprio autor

3.3 *Backtracking*

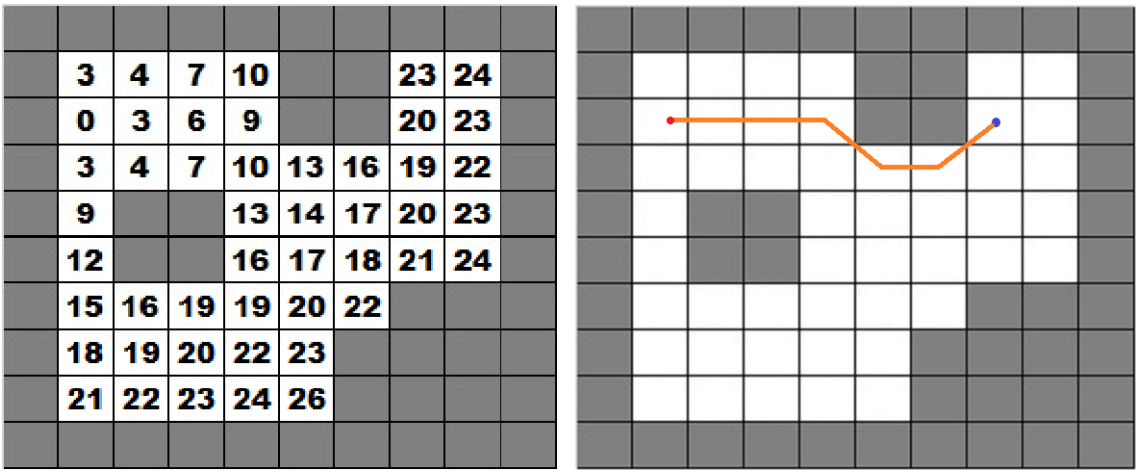
O método de *backtracking* tem a função de retornar o robô quando se encontra em um lugar sem opções de caminhos não limpos ou em um beco sem saída, necessitando realocar o robô para outra posição.

Primeiro, o algoritmo deve buscar em sua memória o local não limpo mais próximo, assim, para realizar esta escolha, é feito o cálculo da distância euclidiana em todos os quadrados conhecidos que não foram varridos, aquele que apresentar a menor distância, é o ponto de chegada para o *backtracking*.

Também é preciso realizar o caminho mais curto possível até a chegada, um modelo confiável para fazer essa busca, é o algoritmo *Wavefront* do trabalho de ZELINSKY et al., com transformação de distância aplicada em matriz. Com a instrução de seguir sempre os números menores até o local desejado, também possibilita caminhos diagonais na matriz, então o robô sempre faz passos eficientes evitando muitas curvas.

Na Figura 10 o quadrado "0" define o local de chegada, enquanto o robô pode estar em qualquer lugar da matriz que sempre chegará onde deseja desde que haja um caminho entre os dois. O algoritmo sempre instrui a seguir pelo menor valor ao seu redor, incluindo as diagonais, para executar o menor caminho.

Figura 10 – Backtracking com *wavefront*, partindo do quadrado 20 até o 0.



Fonte: Próprio autor

4 PROPOSTA

4.1 Introdução ao método

O algoritmo proposto deve executar a varredura completa de uma área sem o conhecimento prévio dela, ou seja, um algoritmo online, além de conseguir propor a limpeza em um tempo aceitável e com intenção de ter boa performance em relação aos algoritmos da literatura.

O algoritmo mapeia a área da sala em formato de matriz, a construção do caminho será feita por mini caminhos até a varredura completa, como proposto pelo YAKOUBI; LASKRI (2016), desta forma, o algoritmo escolhe o caminho que é o melhor para a situação.

Para realizar a escolha do caminho ideal no local, cada caminho gerado é pontuado pelo algoritmo, e é escolhido aquele com melhor pontuação. A pontuação do caminho tem base em seu formato e se está próximo ou não de local já limpo ou parede.

Para a geração e escolha dos caminhos, foram proposto 3 algoritmos: GRASP (*Greedy Randomized Adaptive Search Procedure*, (FESTA; RESENDE, 2002)), Guloso e Árvore. Cada um apresenta uma abordagem distinta na criação de caminho, com suas vantagens e desvantagens.

1. Algoritmo GRASP: Apresenta um método de geração mais diversificado com base de sorteio, trazendo mais possibilidades de caminhos, porém, pode trazer resultados menos otimizados.

2. Algoritmo guloso: Apresenta o método de geração mais direta, criando caminhos com tendência de pegar sempre a maior pontuação, trazendo resultados mais confiáveis, mas com menos adaptabilidade.

3. Algoritmo árvore: Consegue apresentar todos caminhos possíveis dentro do local, conseguindo aproveitar o máximo a pontuação dos caminhos, porém, a geração de tantos caminhos pode acabar escolhendo um desorientado, por conta dos empates de pontuação, exigindo um método de pontuação mais complexo.

4.2 Construção da solução

Os mini caminhos são os caminhos gerados dentro do alcance do sensor do robô, que simula diversos caminhos possíveis e com base de alguns fatores de pontuação, escolhe o mais atribuído. O tamanho do mini caminho foi definido

para cinco quadrados (abrangência do sensor), o ciclo de tentativas de gerar novos caminhos em 100, ou seja, pode gerar até 100 caminhos diferentes, e os 4 fatores para pontuação são:

- Caminhos retos: São os caminhos mais simples que exigem menos tempo de execução, já que as curvas normalmente exigem a parada do robô para virar.
- Duas curvas de 90° em seguida ou curva em "U": Presente bastante em percurso em zigue-zague, é um padrão utilizado em muitos trabalhos de limpeza, assim, as curvas em "U" pode servir de orientação para sempre seguir este padrão.
- Caminho perto de parede: Para criar a tendência de limpar os cantos da casa e ter um rumo para seguir, desta forma o robô evita deixar espaços não limpos entre as suas passagens.
- Caminho perto de local já limpo: Com mesmo propósito do anterior, porém agora com tendência de seguir áreas já limpas quando já não tiver paredes por perto.

A construção do mini caminho é feita quadrado por quadrado na matriz, até que atinja o limite de tamanho. Mas para diversificar os caminhos, utilizaremos o algoritmo baseados em GRASP, onde criaremos diversos mini caminhos baseado em sorteio e pontuação, e no final escolheremos o mais pontuado.

Para a criação inicial do caminho, é visualizado cada quadrado adjacente disponível do robô, onde cada um tem sua própria pontuação, medido pelos 4 fatores mencionados, e a escolha de qual direção que o robô toma é feita por um sorteio, e as chances do quadrado de ser escolhido é proporcional a sua pontuação.

Esta criação de mini caminhos é feita diversas vezes com intenção de diversificar os caminhos, e no final, aquele que tiver a maior pontuação será escolhido para fazer o caminho, no caso de empate, será escolhido aleatoriamente entre os empatados. Este processo é repetido até que faça a varredura completa ou em caso de ainda ter áreas não limpas, também está implementado um modelo de *backtracking*, para movimentar o robô a uma área não limpa mais próximo e continuar o percurso.

Também foi criado um modelo mais guloso, onde já não existe mais o sorteio, e a escolha de cada quadrado é feita pela maior pontuação. Esse algoritmo tem capacidade de trazer maior confiabilidade nos resultados, já que retira parte do fator aleatório do mini caminho, porém, ainda existe em caso de empate com 2 melhores pontuados.

Por fim, mais um modelo foi implementado, chamado de árvore, para considerar todos os caminhos possíveis dentro da área do sensor. O algoritmo usa uma

função recursiva que gera uma árvore que a partir de um quadrado inicial, ele procura sempre os quadrados adjacentes recursivamente até que não exista mais possibilidade ou atinja o tamanho máximo definido. Em seguida, para cada caminho criado, é colocado a pontuação baseada nos mesmos 4 fatores, e quando há empate entre os melhores, a escolha é aleatória entre eles.

4.3 Melhorias da solução(offline)

Com o primeiro passo do algoritmo concluído, onde teve o propósito de completar a varredura e ter o conhecimento total da área. Assim, podemos acrescentar um próximo método para aumentar a eficiência do percurso, o qual é uma adaptação da varredura completa para tentar atingir o menor tempo possível do percurso, os métodos utilizados são: *Fix and Optimize* e algoritmo genético.

Apesar de a pontuação normalmente trazer uma cobertura completa aceitável para maioria das instâncias, ainda existem diversos aspectos que podem trazer o pior do tempo de percurso, ainda mais quando tratamos um ambiente de muitas mobílias e de formato irregular.

Deste modo, a pontuação de cada um dos 4 fatores deve variar para adaptar a sala, assim, as pontuações apenas são fixas na cobertura inicial, pois em futuras coberturas, ela pode variar conforme o algoritmo simula diversas vezes a instância conhecida, com a tentativa de encontrar melhores maneiras de pontuar, pois, cada fator pode ser mais ou menos importante para conseguir melhores tempos de percurso.

A pontuação varia conforme os 4 parâmetros que são criados para cada um dos fatores de pontuações. Eles multiplicam os valores da pontuação, podendo até mesmo inutilizar um dos fatores de pontuação se não for relevante para atingir o melhor percurso. Desta forma a adaptação é feita por 2 algoritmos com intuito de aperfeiçoar os parâmetros e criar a melhor cobertura completa para determinada instância.

O primeiro é o *Fix and Optimize*, ele faz o ajuste de um parâmetro por vez, onde decrementa e incrementa e visualiza se em um dos dois teve alguma otimização, se houve, ele mantém aquele com o melhor tempo, refaz até que o parâmetro já não melhore em mais nada, e passa para o próximo. Este ciclo acontece até que nenhum deles já não aprimore o tempo de percurso, e os parâmetros finais será a base para as varreduras seguintes.

O segundo é o algoritmo genético, onde com base de uma família inicial com 4 parâmetros criados aleatoriamente, são feitos mutações, onde faz troca de um dos parâmetros por um aleatório, e *crossover*, onde troca um dos parâmetros por outro da mesma família. No fim, são geradas mais duas famílias, das quais serão

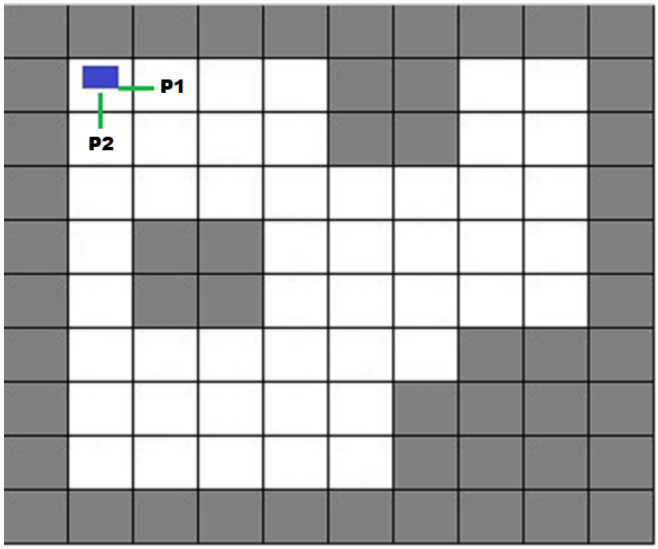
selecionadas apenas com os melhores tempos de todas as famílias, para servir como a próxima geração e continuar o ciclo. Este procedimento é feito em um determinado número de vezes dependendo do tempo de simulações, onde quanto mais tempo é fornecido, melhores os parâmetros, e ao final, é utilizado os dados do primeiro membro da família que apresenta o melhor tempo obtido.

4.4 Exemplo

Esta seção contém as etapas dos procedimentos para a varredura completa, onde o algoritmo passa pelas seguintes etapas: Criação de mini caminhos, escolha de mini caminho, movimento do percurso.

A primeira etapa, na Figura 11, o robô visualiza todos os locais livres ao seu redor e pontua cada quadrado adjacente, e escolhe um deles com uma chance proporcional a quantidade de pontos. O tamanho de caminho estabelecido é de 5 quadrados e a criação se repete até 100 vezes, podendo gerar até 100 caminhos diferentes.

Figura 11 – Escolha inicial para construção de mini caminhos.



Fonte: Próprio autor

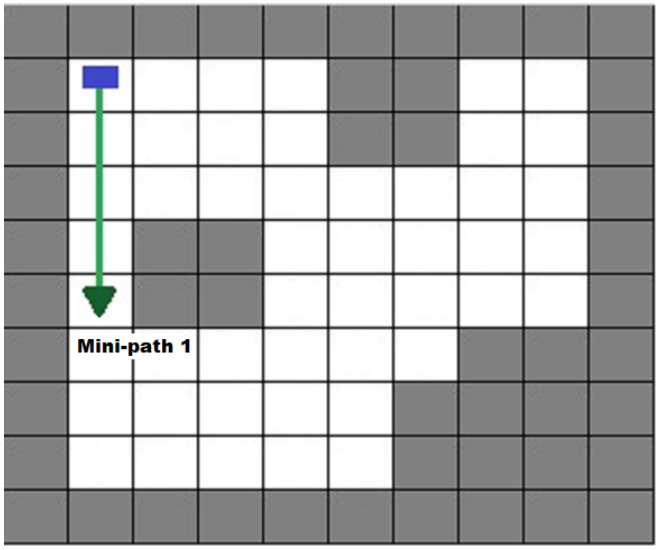
Neste exemplo temos 2 quadrados com pontuações P1 e P2, como estão sobre os mesmos aspectos, possuem as mesmas pontuações, mas para facilitar as próximas explicações, consideramos que P1 tenha 6 pontos e P2, 4 pontos.

- GRASP: Um sorteio, onde a escolha de P1 seria de 60% e P2 de 40%.
- Guloso: A escolha seria do P1, já que possui maior pontuação.

- Árvore: Diferente das anteriores, ela sempre gera todos os mini caminhos possíveis até o tamanhos máximo e armazena as pontuações de cada quadrado.

Após a formação do conjunto de caminhos possíveis, na segunda etapa, é somado o total de pontuação para cada mini caminho criado, ou seja, a soma dos 5 quadrados ou menos em caso de caminhos menores. A escolha é pelo maior pontuado do conjunto ou sorteado entre os melhores em caso de empate. Na Figura 12 se trata do caminho com maior pontuação já escolhida.

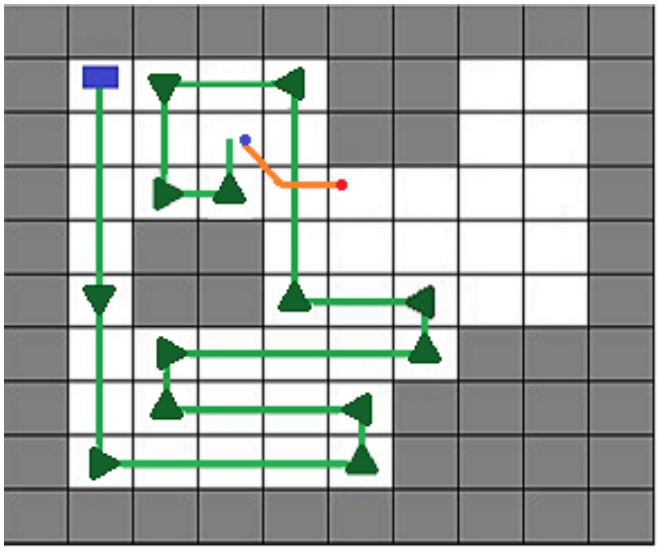
Figura 12 – Conclusão do primeiro mini caminho.



Fonte: Próprio autor

Em seguida, na Figura 13, a terceira etapa é a condição de saída ou *backtracking*. Durante a varredura, se caso não encontrar mais posições válidas para percorrer em volta, é feito uma busca na memória para encontrar um espaço mais próximo ainda não preenchido, realizar o reposicionamento com *backtracking* para este novo local e recomeçar o ciclo. Deste modo, após todos os espaços da área de conhecimento do robô forem limpos, é finalizado o algoritmo, e a cobertura é tratada como completa.

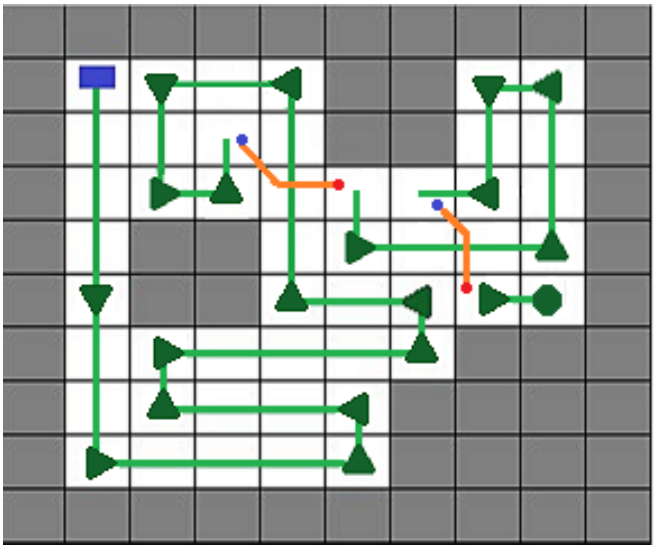
Figura 13 – Reposicionamento do robô com para um local não limpo (*Backtracking*).



Fonte: Próprio autor

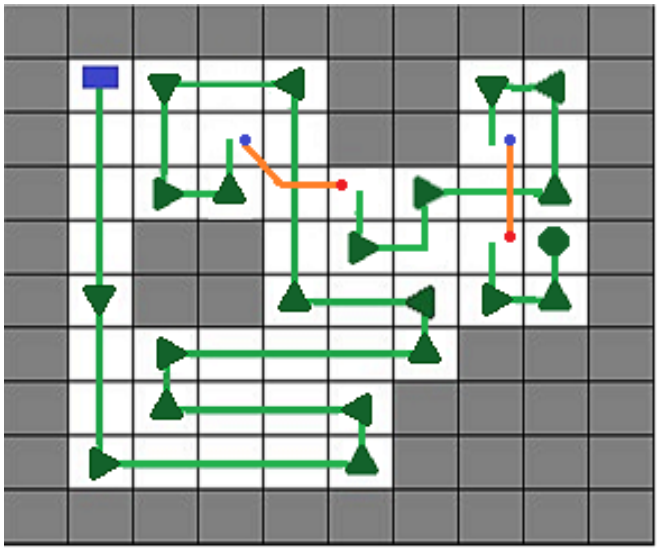
Os 3 métodos propostos inicialmente, possuem seleção de caminhos diferentes, os quais geram varreduras completas diferentes, apresentadas nas Figuras 14, 15 e 16.

Figura 14 – Conclusão do percurso completo pelo método GRASP.



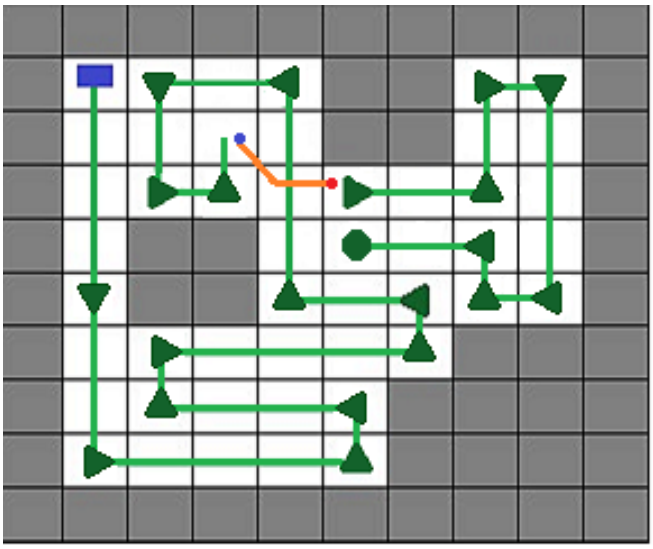
Fonte: Próprio autor

Figura 15 – Conclusão do percurso completo pelo método guloso.



Fonte: Próprio autor

Figura 16 – Conclusão do percurso completo pelo método de árvore.



Fonte: Próprio autor

Uma vez já conhecida a área, se torna possível tratá-la com simulações para aperfeiçoar as próximas limpezas para uma varredura mais rápida. Deste modo, adaptamos as pontuações para que sejam mais refinadas para esta instância.

Inicialmente os 4 fatores de pontuação apresentam um ganho de:

1. Reto: 2 pontos, apresenta a melhor pontuação por ser o movimento que exige menos tempo de percurso.

2. Curva U: 1 ponto, é importante para o percurso ter uma tendência de fazer zigue-zague.

3. Parede adjacente: 1 ponto para servir como orientação e não deixar o robô desamparado e evitar deixar locais não limpos para trás.

4. Quadrado limpo adjacente: 1 ponto mesmo propósito de não deixar o robô desamparado e evitar os locais não limpos para trás.

Para os próximos métodos, os 4 fatores são tratados como 4 parâmetros para alterar as pontuações no algoritmo. Assim, é feito a busca pelos parâmetros que sejam ótimos para a instância que é conhecida, e gerar uma ótima varredura completa. Pelo modelo de Fix-and-Optimize, os parâmetros iniciais de (2, 1, 1, 1) se tornam 2 novos parâmetros para teste, sendo eles: (1, 1, 1, 1) e (3, 1, 1, 1). No caso de algum deles renderem um tempo menor de percurso, se tornam as novas pontuações para a instância. e refaz o processo, considerando o (1, 1, 1, 1) como novo parâmetro, renderá 2 novos, sendo eles: (0.5, 1, 1, 1) e (1.5, 1, 1, 1).

No caso de nenhum trazer benefícios, é mantido o anterior, e a mudança passa a ser no próximo fator de pontuação. Deste modo, o parâmetro (1, 1, 1, 1) se torna agora, (1, 0.5, 1, 1) e (1, 1.5, 1, 1).

Esse ciclo acontece até que nenhuma melhoria seja encontrada na mudança de cada pontuação, por fim, o último parâmetro passa a ser a base para realizar as próximas coberturas desta instância.

Porém, este modelo apenas faz uma leve e rápida adaptação, já que as instâncias normalmente exigem variações mais drásticas nas pontuações para realmente terem mudanças no percurso e chegar em uma ótima solução.

Para este caso, foi implementado o modelo de genético que gera uma família de 20 parâmetros, e realizam *crossover* e mutações, gerando mais 40 membros, totalizando 60 parâmetros que serão testados e medidos seus tempos de percurso, para somente os 20 melhores passarem para o próximo ciclo e repetir os processos.

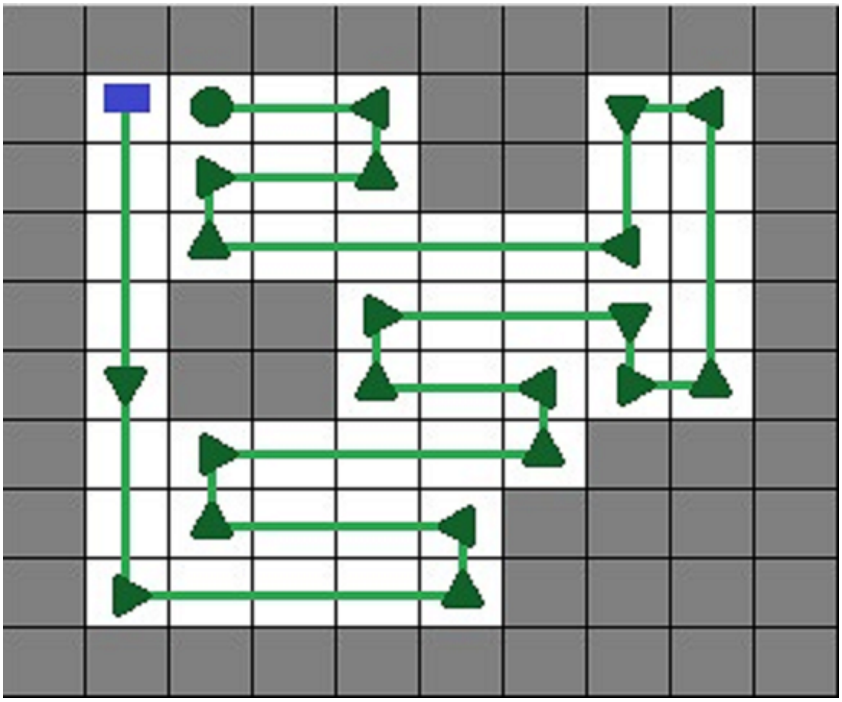
- *Crossover*
  - Probabilidade de ocorrer: 20%
  - Sorteia um dos membros da familia, para trocar um de seus parâmetros com ele.
- Mutação
  - Probabilidade de ocorrer: 80%

– Sorteia um dos parâmetros para alterar seu valor, em uma faixa de -30% até +30% do próprio valor.

Foi tomada a decisão de uma maior probabilidade de ocorrer mutação, com intenção de sempre buscar a maior variação entre as gerações. Deste modo, aumenta as chances de uma família chegar a uma coleção de parâmetros ideal para a instância tratada.

Este ciclo é feito 20 vezes, até que no final, é utilizado somente o melhor membro da família para utilizar como base para os próximos percursos, Figura 17.

Figura 17 – Conclusão do percurso completo após adaptação do genético.



Fonte: Próprio autor

Este modelo exige bastante recurso, e torna-se inviável para ser executado em um robô aspirador doméstico. Deste modo, se trata de um modelo que pode ser aplicado com ajuda de um servidor, que pode processar as simulações e enviar os parâmetros adaptados para o robô. Assim, não cria tanta exigência em hardware para o robô, e possibilita um modelo de otimização de percurso.

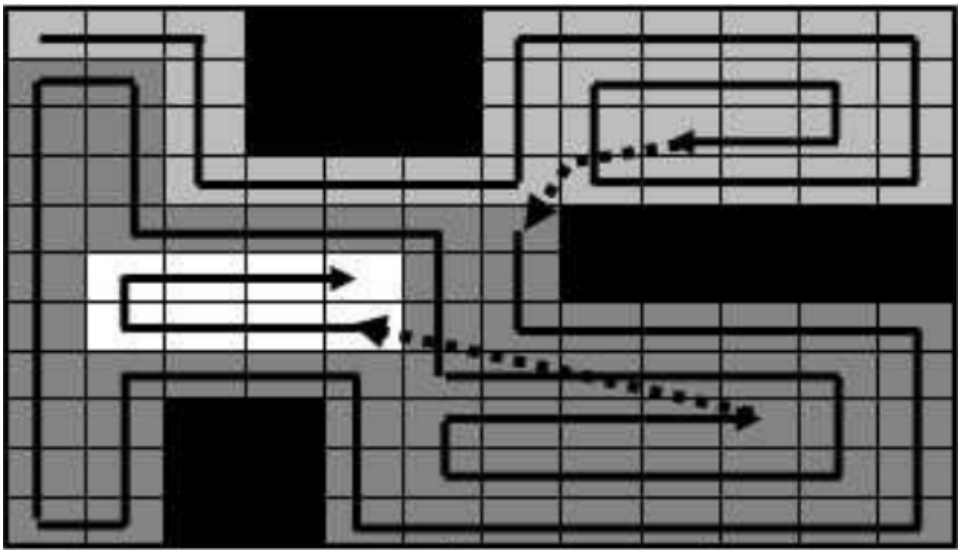
5 RESULTADOS COMPUTACIONAIS

Todos os algoritmos foram desenvolvidos em C++, compilados no Visual Studio Code e testados em uma máquina com as seguintes especificações:

1. Processador: Intel core i7-6700HQ CPU 2.60GHz
2. Memória Ram: 16 GB.

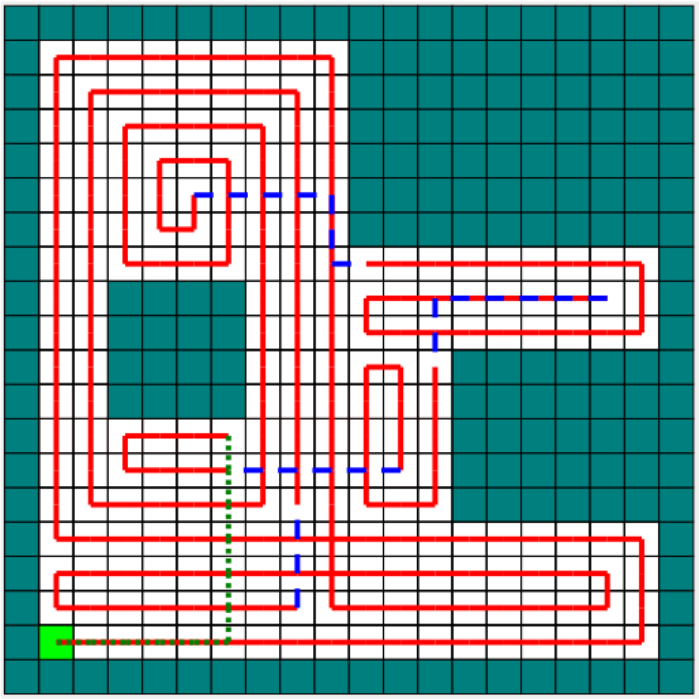
Para a verificação do desempenho do algoritmo proposto em relação aos da literatura, foram testados em diversas instâncias apresentadas na literatura, conforme as Figuras 18 a 33.

Figura 18 – Planejamento Backtracking Algorithm (BSA).



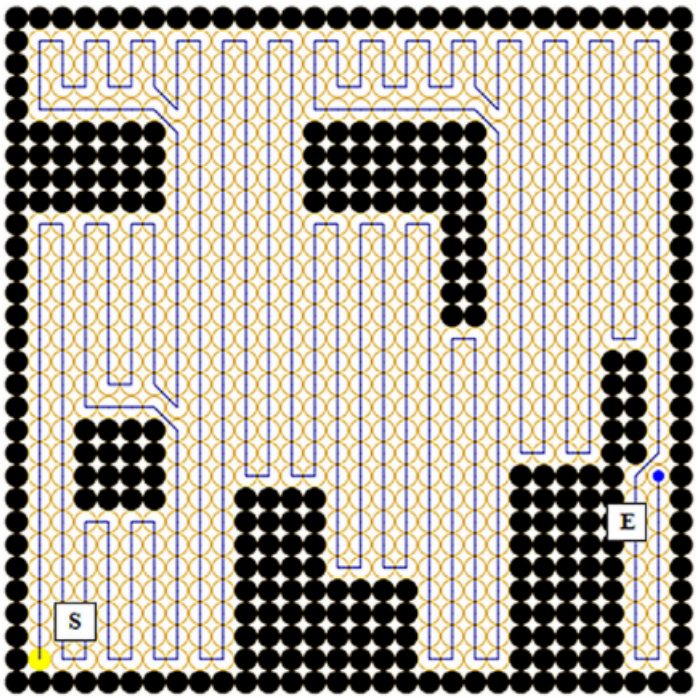
Fonte: (GONZALEZ et al., 2005)

Figura 19 – Planejamento considerando o gasto de energia com Turn-away Starting-point (TASP).



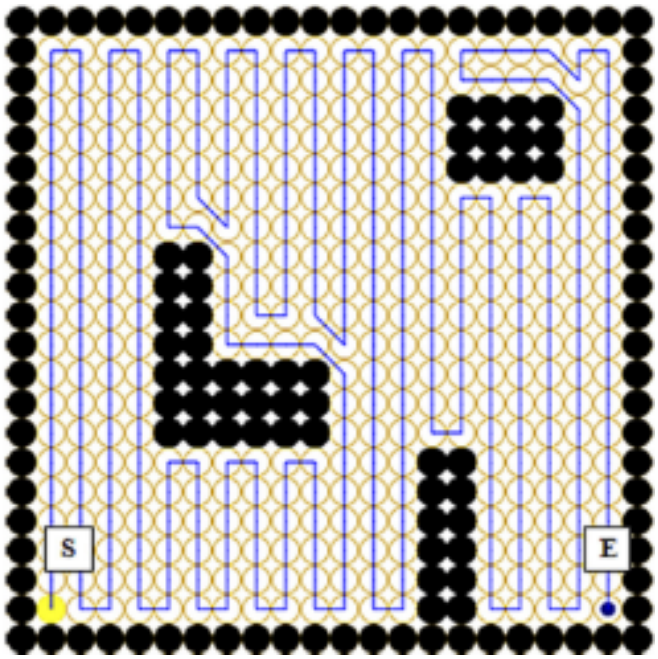
Fonte: (MITSCHKE; UCHIYAMA; SAWODNY, 2018)

Figura 20 – Planejamento com algoritmo genético (GAYA).



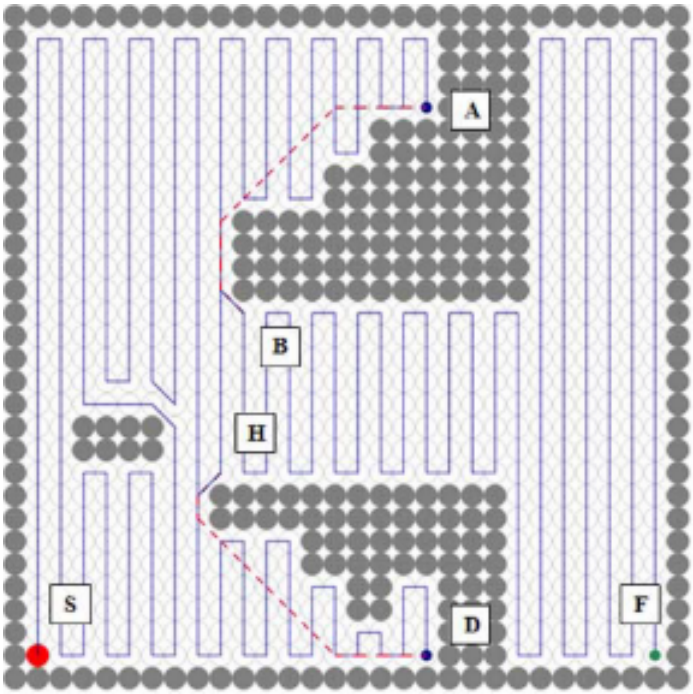
Fonte: (YAKOUBI; LASKRI, 2016)

Figura 21 – Planejamento com algoritmo genético (GAYA2).



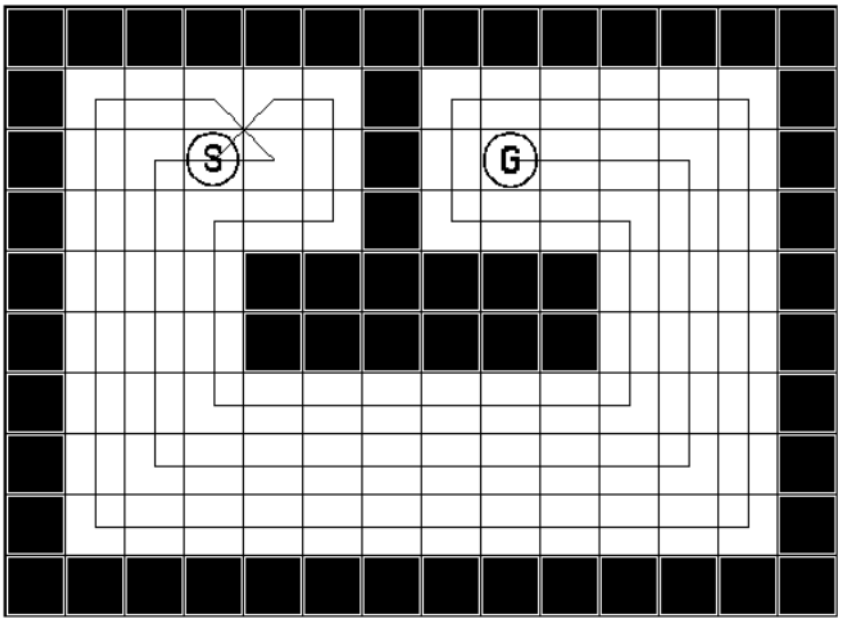
Fonte: (YAKOUBI; LASKRI, 2016)

Figura 22 – Planejamento com rede naural (NNY3).



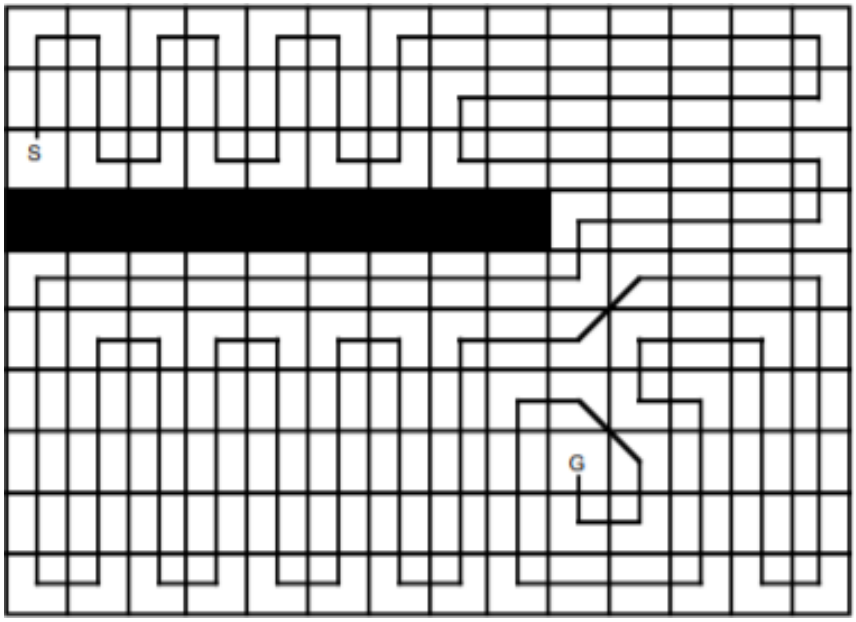
Fonte: (YAKOUBI; LASKRI, 2016)

Figura 23 – Planejamento com transformação de distância Wavefront (PTZ).



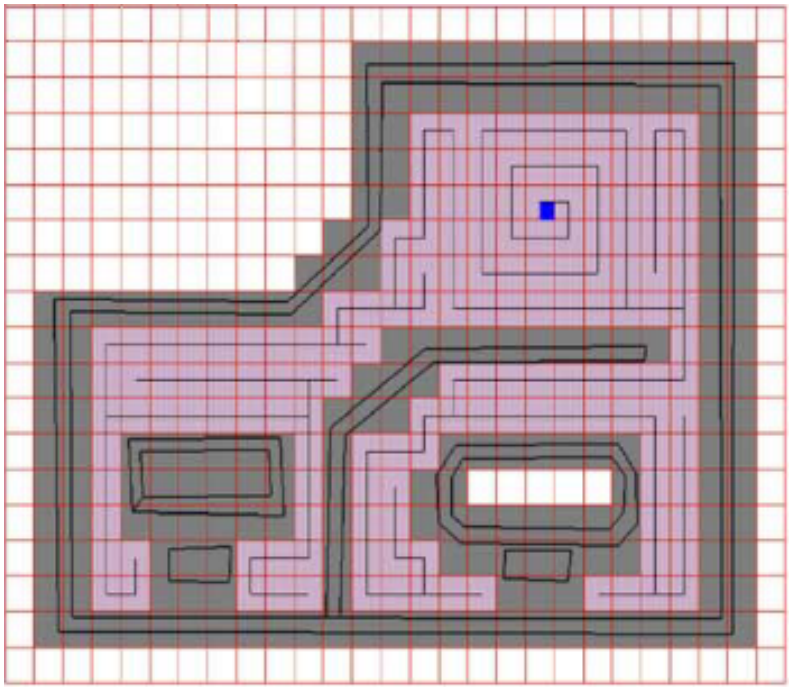
Fonte: (ZELINSKY et al., 1993)

Figura 24 – Planejamento com transformação de distância Wavefront (PTZ2).



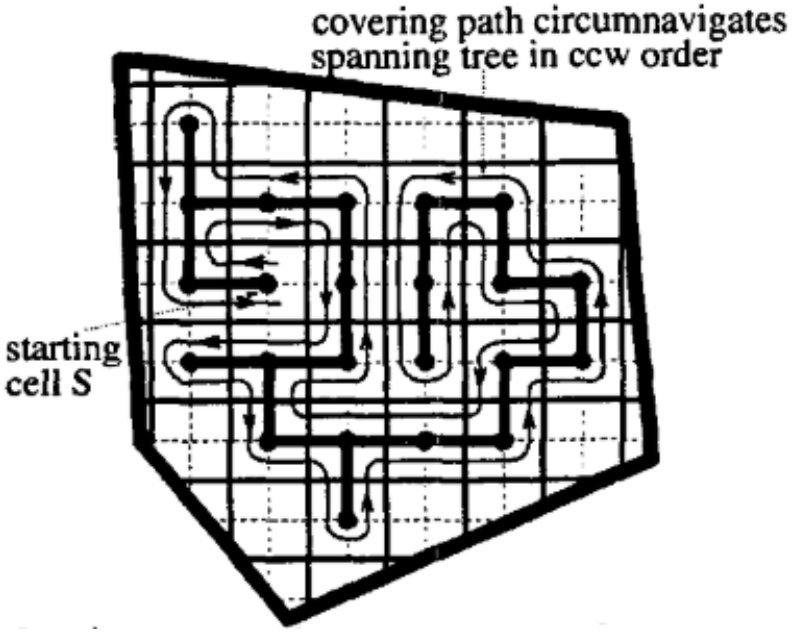
Fonte: (ZELINSKY et al., 1993)

Figura 25 – Planejamento em espiral Spanning Tree Coverage (STC).



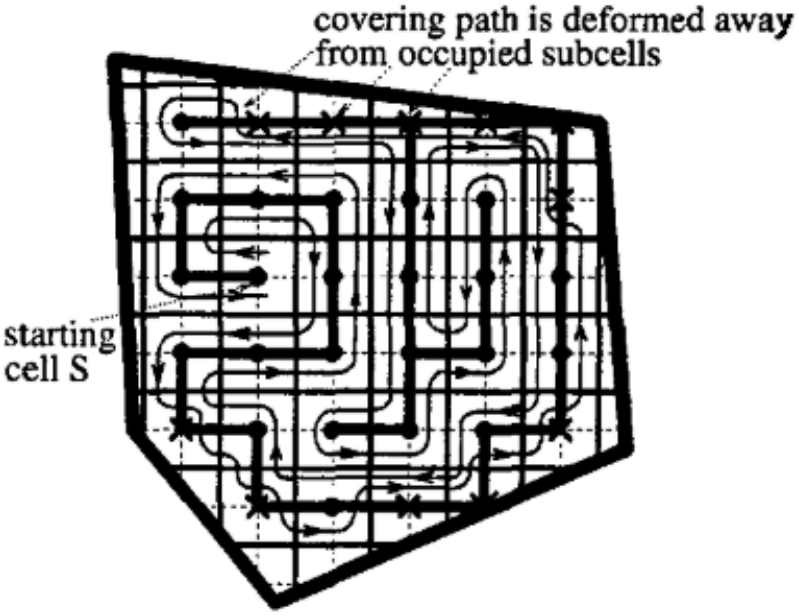
Fonte: (GABRIELY; RIMON, 2002)

Figura 26 – Planejamento em espiral Spanning Tree Coverage (STC1).



Fonte: (GABRIELY; RIMON, 2002)

Figura 27 – Planejamento em espiral Spanning Tree Coverage (STC2).



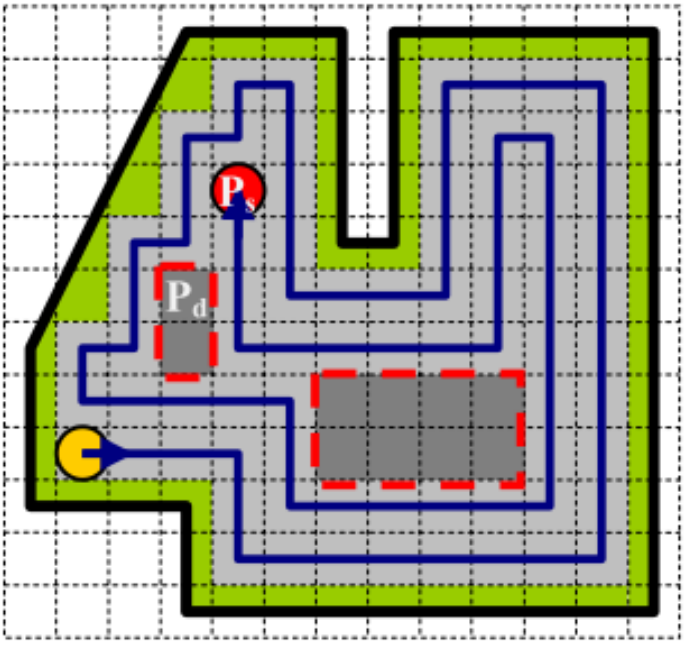
Fonte: (GABRIELY; RIMON, 2002)

Figura 28 – Planejamento com Wavefront (WES).

S	8	7	7	7	7	7	7	7	7	7	8	9	10
9	8	7	6	6	6	6	6	6	6	7	8	9	10
8				5	5	5	5	5			8	9	10
7					4	4	4				9	9	9
6						3				10	9	8	8
6	5					2							7
6	5	4			1	1	1					6	7
6	5	4	3	2	1	G	1	2	3	4	5	6	7
6	5	4	3	2	1	1	1	2	3	4	5	6	7

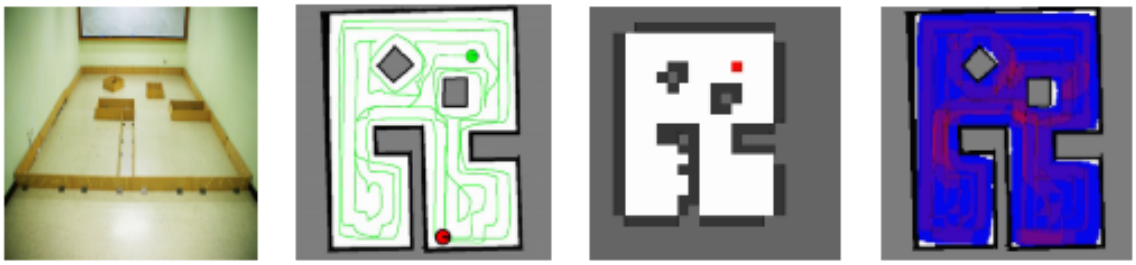
Fonte: (GALCERAN, 2013)

Figura 29 – Planejamento em espiral com transformação de distância inversa (YCM).



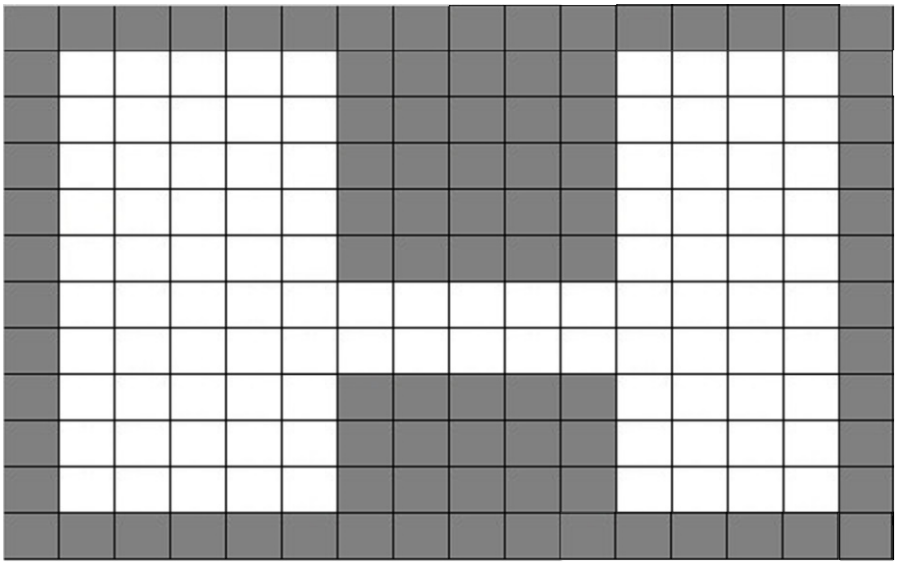
Fonte: (CHOI et al., 2009)

Figura 30 – Planejamento em espiral com transformação de distância inversa (YCM2).



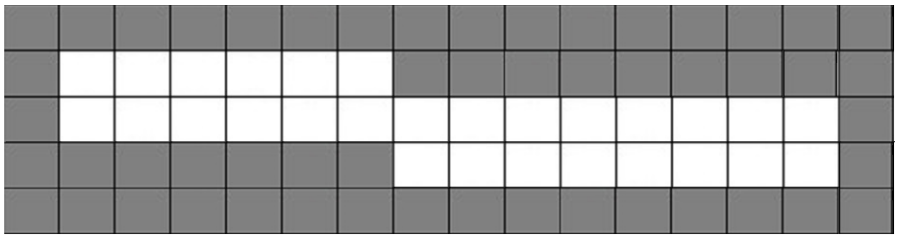
Fonte: (CHOI et al., 2009)

Figura 31 – Instância criada para teste 1.



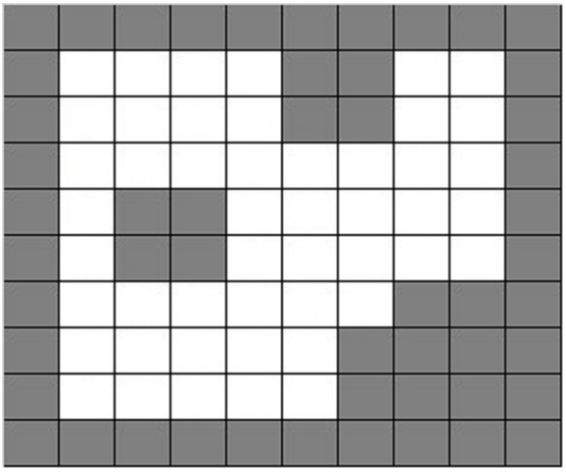
Fonte: Próprio Autor

Figura 32 – Instância criada para teste 2.



Fonte: Próprio Autor

Figura 33 – Instância criada para teste 3.



Fonte: Próprio Autor

5.1 Desempenho dos algoritmos

Mediremos o desempenho feito por cada algoritmo, da literatura e do proposto, baseado no seu tempo de percurso completo de cada instância mencionada anteriormente. Para isso, o tempo gasto do percurso é calculado com base na Tabela 1, o qual pertence a um trabalho que mediu o tempo necessário que o robô leva em média para fazer cada movimento em específico.

Tabela 1 – Tempo e energia necessária para cada movimento do robô.

Movimento	Energia (J)	Tempo (s)
Reto	2.65	1.034
Acelerar	14.29	1.38
Desacelerar	6.18	1.8
Virar esquerda	73.41	8.0
Virar direita	73.41	8.0
Meia-volta	129.57	11.82

Fonte: (MITSCHKE; UCHIYAMA; SAWODNY, 2018)

Como o *Backtracking* realiza curvas de outros ângulos, também foi estabelecido outros movimentos na Tabela para complementar o tempo de todas as curvas possíveis no trajeto.

Tabela 2 – Tempo necessário para o robô virar em determinados ângulos.

Movimento	Tempo (s)
Virar 45 graus	6.09
Virar 90 graus	8.00
Virar 135 graus	9.91
Virar 180 graus	11.82

Fonte: Próprio autor

Desta forma, com base dos números de caminhos retos ou curvas, é possível estimar o tempo total em segundos de um percurso completo feito pelo robô. Os tempos dos algoritmos propostos serão medidos em: melhor caso, média de 1000 simulações, e o pior caso obtido de cada instância, assim, apesar dos fatores aleatórios na construção da cobertura completa, podemos estimar sua performance a partir destes dados. Assim, foi estabelecido uma fórmula para calcular o tempo total de cada instância:

$$\begin{aligned}\text{TempoTotal} = (1,38 + 1,8) \cdot (2\ \text{NumBacktracking} + 1) + 1,034\ \text{PassosRetos} \\ + 8\ \text{Viradas90} + 11,82\ \text{Viradas180} + 6,09\ \text{Viradas45} + 9,91\ \text{Viradas135}\end{aligned}$$

Também está expresso, o tempo computacional em milissegundos necessários para finalizar o processo de construção da cobertura completa de cada algoritmo.

A Tabela 3 apresenta os dados da cobertura completa do algoritmo proposto em todas as instâncias mencionados. Por serem métodos que exigem alguns fatores de aleatoriedade, foram definidos seus dados em melhor caso, caso médio e pior caso, nas últimas três colunas apresentam o tempo computacional de cada algoritmo. A seguir as Tabelas 4 e 5, apresenta os dados após as adaptações pelo *Fix and optimize* e algoritmo genético respectivamente. E as últimas Tabelas 6 e 7 mostram os dados obtidos pelos algoritmos da literatura e seu tempo computacional.

Por fim, para visualizar a diferença de desempenho de cada algoritmo, teremos os gráficos nas Figuras 34 e 35. Eles demonstram a média relativa de todas instâncias com relação à melhor solução obtida em cada instância dentre todos algoritmos testados, ou seja, quanto menor a porcentagem, menor a diferença da melhor solução obtida, e assim, melhor o desempenho. Neste gráfico, os algoritmos propostos são expressos em: Guloso, GRASP e Árvore (TREE), e suas adaptações: *Fix and Optimize* (FO) e Genetic Algorithm (GA), e os algoritmos da literatura:

- LeftHand de GONZALEZ et al.
- Espiral
- Spanning Tree Coverage de GABRIELY; RIMON
- Wavefront de ZELINSKY et al..
- LongestPath de MITSCHKE; UCHIYAMA; SAWODNY
- Zigue-Zague

Tabela 3 – Tempo de percurso inicial, dividido em: melhor caso, caso médio, pior caso, em segundos; e tempo computacional em milisegundos

	Guloso			Grasp			Tree			Tempo computacional		
Instância	Melhor	Médio	Pior	Melhor	Médio	Pior	Melhor	Médio	Pior	Guloso	Grasp	Tree
BSA	388.14	388.14	388.14	342.43	359.13	398.88	342.43	358.50	398.88	5.55	6.67	1.03
TASP	582.46	628.55	730.85	611.97	711.11	958.41	621.39	714.79	990.00	11.10	15.98	3.37
GAYA	1532.07	1686.49	1834.48	1580.72	1925.85	2329.42	1593.31	1926.89	2406.46	35.88	47.87	15.0
GAYA2	843.60	880.93	920.60	843.42	1030.84	1273.48	866.77	1028.80	1355.06	16.97	33.41	5.99
NNY3	1471.83	1565.41	1773.47	1304.49	1625.40	2024.01	1328.09	1674.92	1995.07	34.47	57.86	13.7
PTZ	323.57	333.59	344.53	309.07	405.53	474.56	309.07	415.72	474.56	3.27	4.33	0.82
PTZ2	349.56	368.32	389.31	331.49	431.82	567.70	331.49	431.17	548.99	5.99	9.07	1.62
STC	683.83	683.83	683.83	722.50	752.90	799.22	722.50	748.93	796.08	9.65	10.80	3.89
STC1	363.91	384.58	389.66	321.41	383.37	474.27	338.45	392.01	474.27	3.99	6.61	1.35
STC2	407.33	488.04	626.86	415.49	529.01	661.34	415.49	541.88	661.34	5.35	7.69	2.12
WFS	342.58	349.15	354.56	410.88	417.56	435.75	410.88	417.20	435.75	3.94	5.52	0.70
YCM	346.24	346.24	346.24	328.33	361.81	443.84	328.33	365.19	438.59	4.02	5.42	1.06
YCM2	1141.91	1273.99	1408.19	**1066.18**	1273.35	1654.11	1103.46	1276.91	1554.35	21.82	30.78	7.47
TEST1	344.81	349.79	358.02	292.20	374.88	456.65	292.20	374.11	456.65	3.89	5.39	0.92
TEST2	**83.71**	**83.71**	**83.71**	87.48	125.79	166.59	87.48	123.12	166.59	1.02	1.18	0.10
TEST3	250.88	254.00	260.63	220.21	244.32	252.95	220.21	241.92	252.95	1.85	2.77	0.30

Tabela 4 – Tempo de percurso com adaptação do *Fix and optimize*, dividido em: melhor caso, caso médio, pior caso, em segundos; e tempo computacional em milisegundos

	Guloso			Grasp			Tree			Tempo computacional		
Instância	Melhor	Médio	Pior	Melhor	Médio	Pior	Melhor	Médio	Pior	Guloso	Grasp	Tree
BSA	351.89	352.11	352.32	342.43	376.69	441.33	342.43	383.90	441.33	581.78	759.61	277.23
TASP	582.46	629.71	733.52	**560.17**	629.32	802.70	**560.17**	622.64	802.70	1411.22	6226.90	1139.81
GAYA	1531.04	1691.27	1806.12	1439.59	1833.05	2279.53	1494.28	1777.64	2201.77	4598.65	21129.27	3716.96
GAYA2	**823.58**	1003.29	1172.26	848.46	1006.77	1185.93	839.25	927.54	1069.61	2970.21	2551.73	1595.02
NNY3	1441.78	1550.23	1746.40	**1270.44**	1603.57	2039.56	1311.09	1533.23	2060.50	4454.07	5781.41	2905.85
PTZ	315.57	326.56	338.17	**243.18**	348.79	500.47	295.18	334.30	389.64	430.48	829.75	165.53
PTZ2	**281.60**	325.39	380.84	289.60	380.76	508.01	289.60	390.12	521.90	694.67	1892.78	271.56
STC	683.83	683.83	683.83	664.44	664.44	664.44	664.44	664.44	664.44	843.01	1047.33	974.52
STC1	**285.23**	345.63	428.34	289.41	322.25	410.22	289.41	328.76	360.76	513.57	1838.43	187.03
STC2	413.83	544.37	620.60	423.60	459.00	574.71	408.32	452.96	581.48	1466.83	693.92	268.27
WFS	342.58	348.50	354.56	377.80	383.57	392.09	377.80	385.65	392.09	409.03	604.93	158.75
YCM	**292.10**	383.22	500.56	324.10	390.66	525.69	324.10	398.02	520.69	593.83	1220.45	197.73
YCM2	1143.53	1247.37	1397.34	1101.87	1237.49	1435.61	1114.70	1215.11	1403.79	3706.28	3408.66	775.56
TEST1	332.99	343.18	348.99	292.20	327.40	361.48	292.20	337.09	361.48	397.03	1076.06	164.33
TEST2	**83.71**	**83.71**	**83.71**	87.48	94.84	107.06	87.48	97.21	107.06	98.68	188.13	18.33
TEST3	**203.78**	238.67	266.88	**203.78**	244.36	286.70	**203.78**	227.65	286.70	354.15	387.06	48.92

Tabela 5 – Tempo de percurso com adaptação do algoritmo genético, dividido em: melhor caso, caso médio, pior caso, em segundos; e tempo computacional em milisegundos

	Guloso			Grasp			Tree			Tempo computacional		
Instância	Melhor	Médio	Pior	Melhor	Médio	Pior	Melhor	Médio	Pior	Guloso	Grasp	Tree
BSA	359.28	359.28	359.28	**326.43**	326.95	437.96	**326.43**	**326.43**	**326.43**	2044.97	2888.87	968.98
TASP	565.39	617.85	694.24	570.53	606.87	766.90	570.53	622.64	791.26	4855.01	5410.40	1815.53
GAYA	1432.33	1692.17	2076.08	1498.08	1741.59	2138.55	1602.77	1777.64	2010.86	11514.14	15981.12	5583.45
GAYA2	839.31	885.29	942.62	865.56	941.03	1076.06	835.70	927.54	1004.94	6049.24	8347.44	2531.99
NNY3	1421.70	1595.58	1923.52	1362.83	1615.01	2020.63	1363.75	1533.23	1610.53	11430.20	15834.90	4949.51
PTZ	246.33	298.55	363.28	246.33	295.08	375.53	246.33	334.30	375.82	1631.61	2131.73	861.29
PTZ2	314.03	314.03	314.03	306.03	306.18	341.56	306.03	306.03	306.03	2483.81	3250.68	1104.81
STC	674.03	674.03	674.03	664.44	664.44	664.44	664.44	664.44	664.44	3280.17	4060.19	1852.08
STC1	**285.23**	365.64	459.76	286.27	343.25	438.23	289.41	328.76	400.16	1883.53	2529.91	1028.14
STC2	389.58	409.39	430.01	415.33	467.59	583.85	423.60	452.96	505.40	2298.09	2916.18	1230.36
WFS	342.58	349.15	354.56	377.80	377.80	377.80	377.80	377.80	377.80	1702.06	2576.69	822.83
YCM	324.96	324.96	324.96	294.60	419.48	423.62	367.46	367.46	367.46	1831.65	2586.81	942.02
YCM2	1128.65	1224.22	1412.17	1083.31	1246.26	1425.41	1118.03	1215.11	1443.19	6770.78	9839.22	3125.76
TEST1	316.63	324.21	332.63	308.20	315.66	369.23	293.23	293.23	293.23	1790.90	2312.56	897.07
TEST2	**83.71**	**83.71**	**83.71**	87.48	87.48	87.48	87.48	87.48	87.48	904.99	970.67	619.44
TEST3	211.78	211.78	211.78	220.21	220.21	220.21	**203.78**	**203.78**	**203.78**	1181.68	1531.88	673.78

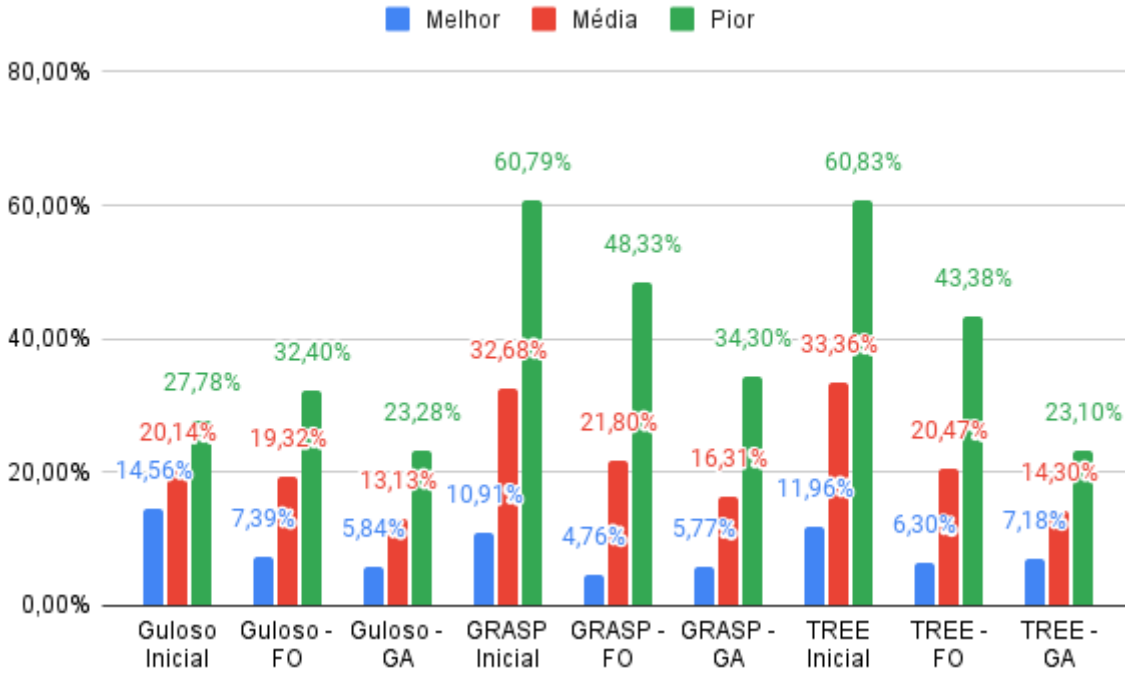
Tabela 6 – Tempo de percurso gasto em segundos pelas heurísticas da literatura

	LeftHand	Espiral	STC	Wavefront	LongestPath			Zigue-zague		
Instância	Caso único	Caso único	Caso único	Caso único	Melhor	Média	Pior	Melhor	Média	Pior
BSA	392.92	418.57	482.98	494.97	392.09	392.09	392.09	382.74	382.74	382.74
TASP	608.82	582.46	759.31	865.76	643.46	647.38	651.46	585.45	639.03	690.92
GAYA	1644.69	1727.77	1842.61	2271.84	1568.48	1594.24	1619.89	**1341.50**	1537.48	1746.75
GAYA2	867.20	879.72	1123.96	1318.78	888.98	904.89	920.66	869.16	1049.67	1229.02
NNY3	1397.05	1504.25	1670.04	1839.95	1520.39	1539.82	1558.04	1524.49	1546.70	1563.69
PTZ	338.49	364.72	256.90	389.29	367.95	367.95	367.95	354.04	404.99	460.02
PTZ2	289.60	333.83	309.40	537.60	341.83	341.83	341.83	346.77	371.01	395.06
STC	678.41	**608.61**	1059.12	810.32	781.95	781.95	781.95	764.39	777.16	789.00
STC1	390.87	**285.23**	368.24	429.34	293.23	293.23	293.23	338.87	370.50	442.54
STC2	558.46	431.15	606.82	579.22	383.33	444.58	468.39	**367.33**	422.39	501.62
WFS	387.91	**334.58**	633.21	492.20	354.56	354.56	354.56	516.00	536.77	546.12
YCM	365.63	377.23	430.55	483.99	353.48	353.48	353.48	340.56	350.15	367.66
YCM2	1186.48	1231.45	1393.83	1595.93	1145.02	1208.27	1256.64	1190.36	1273.82	1524.44
TEST	**291.05**	340.99	328.59	469.41	322.87	322.87	322.87	353.19	366.65	370.22
TEST2	**83.71**	87.48	108.21	99.10	**83.71**	**83.71**	**83.71**	87.48	87.48	87.48
TEST3	252.63	281.63	320.17	305.83	273.99	273.99	273.99	249.49	249.49	249.49

Tabela 7 – Tempo computacional gasto em milissegundos pelas heurísticas.

Instância	LeftHand	LongestPath	Zigue-zague	Espiral	STC	Wavefront
BSA	3.38	2.45	3.25	3.53	5.35	2.31
TASP	3.81	2.94	4.17	3.06	6.35	4.50
GAYA	6.46	10.40	5.96	14.32	11.58	11.31
GAYA2	5.22	3.73	3.56	4.67	4.75	4.61
NNY3	5.38	11.14	8.01	11.73	9.46	4.51
PTZ	2.76	2.44	3.30	4.24	2.37	2.64
PTZ2	3.48	2.23	3.22	2.84	3.09	3.38
STC	6.73	5.30	5.53	4.80	8.39	8.18
STC1	4.73	2.30	2.56	2.71	3.29	2.75
STC2	3.04	2.48	2.64	3.16	4.14	2.34
WFS	2.62	2.44	2.85	2.46	3.17	4.16
YCM	3.91	2.18	2.30	5.02	3.03	2.51
YCM2	6.29	5.93	7.29	6.77	9.26	6.40
TEST	2.80	2.34	3.01	3.97	3.37	2.97
TEST2	4.03	2.02	2.68	2.57	2.63	2.44
TEST3	1.90	2.11	2.42	1.94	2.69	2.72

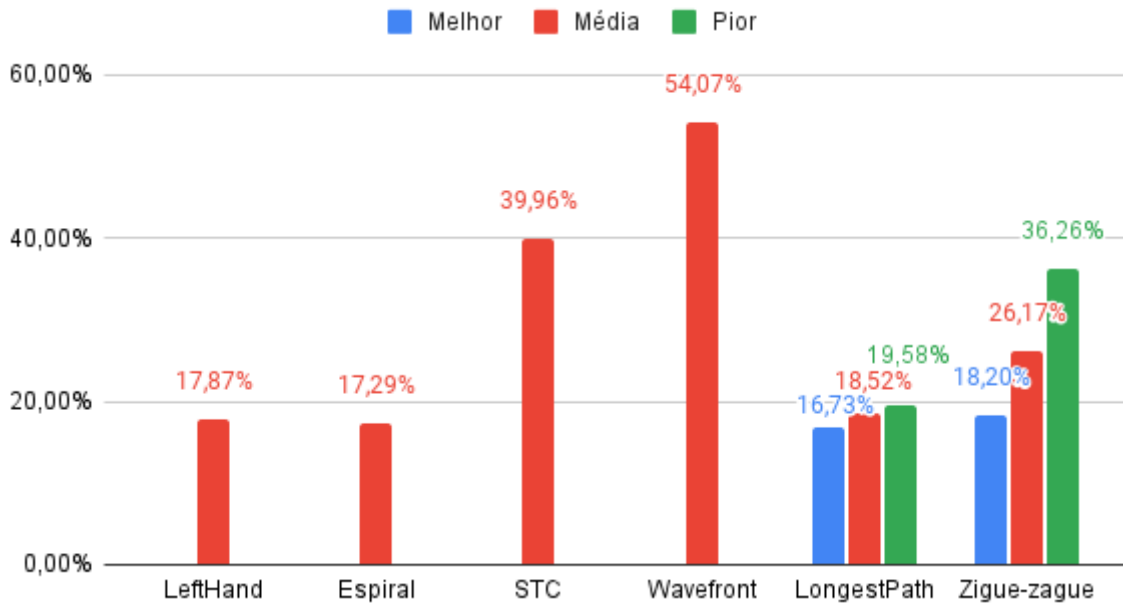
Figura 34 – Diferença relativa dos algoritmos propostos com relação à melhor solução.



Fonte: Próprio Autor

Como expresso na Figura 34, os percursos iniciais realizados pelo algoritmo proposto guloso apresenta uma média melhor entre os propostos com uma diferença de 20,14% em relação ao melhor caso. O GRASP consegue atingir melhores tempos se consideramos apenas os melhores casos, com uma perda de 10,91% em relação ao

Figura 35 – Diferença relativa das heurísticas com relação à melhor solução.



Fonte: Próprio Autor

melhor caso, porém com uma média pior de 32,68%, as mesmas características são parecidas com o algoritmo Tree.

As adaptações trazem grande melhorias para os algoritmos iniciais, onde a média do guloso inicial com 20,14% caiu para 13,13% no guloso - GA. Enquanto o GRASP inicial no melhor caso era 10,91%, e chegou ao 4,76% após *Fix and optimize*, ou seja, apresenta uma média de apenas 4,76% pior em relação aos melhores casos, o melhor dentre todos os algoritmos se considerarmos apenas o melhor caso. No entanto, ainda é preciso observar que os piores casos deles está muito elevado, por conta de exigir mais fatores aleatórios.

Na Figura 35 os melhores desempenhos estão entre o Espiral, LeftHand e o LongestPath com porcentagens próximos a 18%, ou seja, a média de suas instâncias normalmente são 18% piores que o melhor caso. Enquanto STC e Wavefront apresentaram os piores resultados nas instâncias tratadas, com porcentagens de 39,96% e 54,07%, apresentando tempo bem distante do melhor casor.

O algoritmo guloso proposto inicial apresenta um tempo perto da média entre os melhores tempo das heurísticas da literatura, deste modo, é um tempo aceitável para um algoritmo online fazer seu percurso completo. Além disso, a adaptação posterior traz um tempo melhor em quase todas as instâncias. No entanto, é um processo que exige muito mais tempo de processamento, o qual não seria normalmente feito pelo robô, e sim, com auxílio de um servidor com grande capacidade de processamento, e retornaria os parâmetros para o robô utilizar em seu percurso.

6 CONCLUSÕES

O algoritmo online proposto partiu de um método com criação de diversos mini caminhos para procurar o melhor no local, posteriormente, foram implementados modelos de adaptação para diminuir o tempo de percurso. Ambos métodos renderam uma varredura completa com sucesso, a proposta de criar mini caminhos propõe uma grande adaptabilidade para a construção da varredura completa, o que resulta em menos tempo de limpeza para uma grande variedade de residências e o algoritmo offline aplicado para adaptação, traz melhorias significativas no tempo para a maioria das instâncias testadas.

O tempo da solução inicial é razoável em comparação com demais trabalhos relacionados, mas após o conhecimento da área, os modelos de adaptação apresentam capacidade de aperfeiçoar ainda mais a cobertura, o que melhora o tempo do percurso completo, com capacidade de superar a maioria dos algoritmos mencionados, além de encontrar o melhor tempo de algumas instâncias tratadas. No entanto, é preciso pensar em uma forma de adaptação mais eficiente, sua grande exigência de recursos, não torna conveniente sua implementação direta no robô, exigindo um auxílio exterior, como um servidor. Também podemos aperfeiçoar o robô com 2 modos, um para apenas utilizar o melhor cobertura de caminho já simulado, e outro modo de descoberta, para casos de novos quartos, ou mudanças na sala.

Em um futuro, este trabalho pode ser usado como uma base para diversos trabalhos que podem minimizar o tempo de limpeza, como o uso de múltiplos robôs que trabalham em sincronia e memória compartilhada. Também podemos aperfeiçoar os movimentos do robô para minimizar o tempo perdido, como as curvas, os quais mais exigem tempo, é possível realizar uma curva, ao invés da parada para o robô girar 90°.

REFERÊNCIAS BIBLIOGRÁFICAS

CHOI, Y. H. et al. **Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform**. In: IEEE. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2009. p. 5788–5793. Citado 3 vezes nas páginas 16, 17 e 40.

CHOSET, H. **Coverage for robotics–a survey of recent results**. *Annals of mathematics and artificial intelligence*, Springer, v. 31, n. 1-4, p. 113–126, 2001. Citado na página 12.

FESTA, P.; RESENDE, M. G. Grasp: An annotated bibliography. In: _____. *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002. p. 325–367. ISBN 978-1-4615-1507-4. Disponível em: <https://doi.org/10.1007/978-1-4615-1507-4_15>. Citado 2 vezes nas páginas 11 e 25.

GABRIELY, Y.; RIMON, E. **Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot**. In: IEEE. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. [S.l.], 2002. v. 1, p. 954–960. Citado 7 vezes nas páginas 4, 15, 17, 23, 38, 39 e 43.

GALCERAN, E. **A survey on coverage path planning for robotics**. *Robotics and Autonomous systems*, Elsevier, v. 61, n. 12, p. 1258–1276, 2013. Citado 3 vezes nas páginas 12, 13 e 39.

GONZALEZ, E. et al. **BSA: a complete coverage algorithm**. In: IEEE. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. [S.l.], 2005. p. 2040–2044. Citado 6 vezes nas páginas 4, 15, 17, 20, 34 e 43.

HASAN, K. **Path planning algorithm development for autonomous vacuum cleaner robots**. In: IEEE. *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*. [S.l.], 2014. p. 1–6. Citado 2 vezes nas páginas 13 e 17.

HERT, S. **A terrain-covering algorithm for an AUV**. In: *Underwater Robots*. [S.l.]: Springer, 1996. p. 17–45. Citado na página 13.

HUANG, W. H. **Optimal line-sweep-based decompositions for coverage algorithms**. In: IEEE. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. [S.l.], 2001. v. 1, p. 27–32. Citado 2 vezes nas páginas 16 e 17.

MITSCHKE, M.; UCHIYAMA, N.; SAWODNY, O. Online coverage path planning for a mobile robot considering energy consumption. In: IEEE. *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. [S.l.], 2018. p. 1473–1478. Citado 9 vezes nas páginas 4, 16, 17, 20, 21, 35, 42, 43 e 53.

VELOSO, T. ***Vendas de aspirador de pó robô disparam*** 802% ***no Brasil***. Techtudo, 20 de maio de 2020. Acesso em: 26/08/2020. Disponível em: <https://www.techtudo.com.br/noticias/2020/05/vendas-de-aspirador-de-po-robo-disparam-802percent-no-brasil.ghtml>. Citado na página 10.

YAKOUBI, M. A.; LASKRI, M. T. **The path planning of cleaner robot for coverage region using genetic algorithms**. *Journal of innovation in digital ecosystems*, Elsevier, v. 3, n. 1, p. 37–43, 2016. Citado 5 vezes nas páginas 15, 17, 25, 35 e 36.

ZELINSKY, A. et al. **Planning paths of complete coverage of an unstructured environment by a mobile robot**. In: *Proceedings of international conference on advanced robotics*. [S.l.: s.n.], 1993. v. 13, p. 533–538. Citado 7 vezes nas páginas 4, 14, 17, 22, 23, 37 e 43.

ANEXO A – Desempenho dos Algoritmos da literatura

O anexo contém as tabelas com mais especificações sobre a cobertura completa do caminho do robô de todos algoritmos implementados, com as seguintes características:

- Passos: Quantidade de movimentos realizados para passar de um quadrado na matriz para um quadrado adjacente.
- Viradas: O robô para realizar as curvas tem a necessidade de parar e fazer o movimento de giro para completar a virada em si. As viradas também são distintos pelo ângulo de giro, estabelecidos em 45°, 90°, 135° e 180°, para os lados direito e esquerdo.
- PassosBT e ViradasBT: Designa os passos e viradas que são feito pelo *Backtracking*, saindo de um local até a área não limpa mais próximo.
- Quantidade de BT: Mostra quantos *Backtracking* foram necessários para o algoritmo finalizar a varredura completa.
- Tempo total: Tempo necessário para o robô executar a varredura completa com base nos caminhos, viradas e a tabela de tempo de cada movimento.
- Tempo Computacional: Tempo gasto para o algoritmo executar a varredura completa desconsiderando o tempo que o robô leva andando na sala.

Figura 36 – Tempo e energia gastas para realizar os movimentos.

Motion	Energy [J]	Time [s]
Straight	2.65	1.034
Acceleration	14.29	1.38
Deceleration	6.18	1.8
Turn left	73.41	8.0
Turn right	73.41	8.0
Turn 180 °	129.57	11.82

Fonte: (MITSCHKE; UCHIYAMA; SAWODNY, 2018)

Tabela 8 – Desempenho do planejamento Grasp no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	9	58	14	20	8	5	47	7	14	18	7	31	8	6	2
Viradas 90	25	41	91	51	72	17	18	31	22	22	15	22	62	16	4	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	3	8	4	4	2	0	18	2	5	12	1	6	3	0	1
Viradas 90 BT	1	0	5	1	1	3	4	6	2	1	1	1	3	1	0	0
Virada 135 BT	0	1	2	0	0	0	0	2	0	2	4	2	2	1	0	1
Virada 180 BT	0	0	1	1	2	2	0	2	0	1	1	0	1	1	1	0
Quant. de BT	1	2	8	2	2	3	2	9	2	5	7	3	8	2	1	1
Tempo Total(s)	342.43	611.97	1580.72	843.42	1304.49	309.07	331.49	722.50	321.41	415.49	410.88	328.33	1066.18	292.20	87.48	220.21
Tempo Comp.(ms)	6.67	15.98	47.87	33.41	57.86	4.33	9.07	10.80	6.61	7.69	5.52	5.42	30.78	5.39	1.18	2.77

Tabela 9 – Desempenho do planejamento Grasp no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	11	36	112	57	104	23	36	47	11	26	17	17	70	22	12	8
Viradas 90	25	59	143	71	104	24	28	34	33	36	20	26	97	21	7	17
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	0	11	27	17	18	8	12	20	4	9	10	4	22	11	3	3
Viradas 90 BT	4	7	12	4	17	6	8	8	4	6	2	4	10	3	1	2
Virada 135 BT	0	3	5	3	2	2	0	2	2	4	3	2	6	3	1	1
Virada 180 BT	2	1	2	4	5	1	0	3	0	2	1	1	3	1	1	0
Quant. de BT	3	8	18	11	19	7	5	11	4	9	7	6	16	6	3	2
Tempo Total(s)	398.88	958.41	2329.42	1273.48	2024.01	474.56	567.70	799.22	474.27	661.34	435.75	443.84	1654.11	456.65	166.59	252.95
Tempo Comp.(ms)	6.67	15.98	47.87	33.41	57.86	4.33	9.07	10.80	6.61	7.69	5.52	5.42	30.78	5.39	1.18	2.77

Tabela 10 – Desempenho do planejamento Grasp com Fix and Optimize no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	15	30	18	5	3	0	45	7	20	21	0	30	8	6	0
Viradas 90	25	31	81	47	76	15	19	24	18	20	12	29	58	16	4	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	1	10	6	0	0	0	19	2	4	9	0	8	3	0	0
Viradas 90 BT	1	2	5	1	0	1	0	5	2	4	3	0	6	1	0	0
Virada 135 BT	0	1	0	2	0	2	0	1	0	2	3	0	4	1	0	0
Virada 180 BT	0	1	0	0	1	0	0	3	0	1	1	0	1	1	1	0
Quant. de BT	1	3	6	4	1	1	0	9	2	5	7	0	10	2	1	0
Tempo Total(s)	342.43	560.17	1439.59	848.46	1270.44	243.18	289.60	664.44	289.41	423.60	377.80	324.10	1101.87	292.20	87.48	203.78
Tempo Comp.(ms)	759.61	6226.90	21129.27	2551.73	5781.41	829.75	1892.78	1047.33	1838.43	693.92	604.93	1220.45	3408.66	1076.06	188.13	387.06

Tabela 11 – Desempenho do planejamento Grasp com Fix and Optimize no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	14	47	156	49	112	31	21	45	15	32	19	23	69	14	7	6
Viradas 90	26	41	111	65	110	25	27	24	26	23	14	35	73	19	4	22
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	4	8	39	15	25	10	11	19	2	10	12	7	27	10	2	1
Viradas 90 BT	5	4	21	10	10	5	3	5	4	8	1	5	7	0	0	3
Virada 135 BT	1	2	4	1	3	2	1	1	2	4	4	1	7	2	0	1
Virada 180 BT	0	4	2	0	4	2	0	3	0	0	0	1	0	0	1	0
Quant. de BT	5	7	22	11	15	6	5	9	4	11	7	4	15	3	2	2
Tempo Total(s)	441.33	802.70	2279.53	1185.93	2039.56	500.47	508.01	664.44	410.22	574.71	392.09	525.69	1435.61	361.48	107.06	286.70
Tempo Comp.(ms)	759.61	6226.90	21129.27	2551.73	5781.41	829.75	1892.78	1047.33	1838.43	693.92	604.93	1220.45	3408.66	1076.06	188.13	387.06

Tabela 12 – Desempenho do planejamento Grasp com algoritmo genético no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	12	66	21	36	2	2	45	8	12	21	4	43	8	6	2
Viradas 90	24	35	71	45	65	17	19	24	19	22	12	22	55	18	4	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	0	11	5	8	0	0	19	0	3	9	1	8	3	0	1
Viradas 90 BT	0	1	11	2	5	2	1	5	2	4	3	0	4	1	0	0
Virada 135 BT	0	0	1	3	4	0	0	1	0	1	3	1	6	1	0	1
Virada 180 BT	0	2	1	1	0	0	0	3	0	1	1	0	0	1	1	0
Quant. de BT	1	2	10	5	6	1	1	9	2	5	7	1	10	2	1	1
Tempo Total(s)	326.43	570.53	1498.08	865.56	1362.83	246.33	306.03	664.44	286.27	415.33	377.80	294.60	1083.31	308.20	87.48	220.21
Tempo Comp.(ms)	2888.87	5410.40	15981.12	8347.44	15834.90	2131.73	3250.68	4060.19	2529.91	2916.18	2576.69	2586.81	9839.22	2312.56	970.67	1531.88

Tabela 13 – Desempenho do planejamento Grasp com algoritmo genético no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	11	37	143	31	144	19	7	45	18	25	21	24	71	12	6	2
Viradas 90	28	44	98	54	92	19	21	24	25	32	12	21	77	21	4	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	5	11	28	11	40	8	0	19	5	6	9	8	20	5	0	1
Viradas 90 BT	3	2	21	7	11	3	2	5	3	5	3	4	6	2	0	0
Virada 135 BT	1	1	6	5	4	2	0	1	3	4	3	2	7	1	0	1
Virada 180 BT	0	1	4	1	2	0	0	3	0	1	1	0	0	1	1	0
Quant. de BT	4	6	22	10	16	4	2	9	6	8	7	6	16	4	1	1
Tempo Total(s)	437.96	766.90	2138.55	1076.06	2020.63	375.53	341.56	664.44	438.23	583.85	377.80	423.62	1425.41	369.23	87.48	220.21
Tempo Comp.(ms)	2888.87	5410.40	15981.12	8347.44	15834.90	2131.73	3250.68	4060.19	2529.91	2916.18	2576.69	2586.81	9839.22	2312.56	970.67	1531.88

Tabela 14 – Desempenho do planejamento Guloso no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	8	13	50	17	44	12	7	43	11	12	12	7	26	10	2	6
Viradas 90	25	33	75	44	75	18	20	30	26	20	11	22	63	20	3	17
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	4	11	7	8	8	0	17	2	4	9	0	9	5	2	3
Viradas 90 BT	3	1	14	4	8	0	4	4	2	2	1	5	6	1	1	2
Virada 135 BT	0	2	1	1	0	2	0	1	0	4	4	0	5	1	0	1
Virada 180 BT	1	0	0	0	2	0	0	3	0	0	0	1	0	1	0	0
Quant. de BT	3	3	11	4	8	2	2	8	3	5	7	3	10	3	1	2
Tempo Total(s)	388.14	582.46	1532.07	843.60	1471.83	323.57	349.56	683.83	363.91	407.33	342.58	346.24	1141.91	344.81	83.71	250.88
Tempo Comp.(ms)	5.55	12.54	38.11	19.90	38.36	3.73	6.55	9.39	4.55	5.94	4.13	4.06	22.72	4.25	1.11	2.07

Tabela 15 – Desempenho do planejamento Guloso no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	8	22	65	25	56	11	8	43	9	17	14	7	59	11	2	4
Viradas 90	25	40	99	43	94	20	21	30	27	42	11	22	73	20	3	20
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	7	13	12	21	5	4	17	1	4	9	0	16	6	2	1
Viradas 90 BT	3	4	19	2	13	1	4	4	4	7	1	5	13	2	1	2
Virada 135 BT	0	3	1	6	1	3	0	1	1	2	5	0	6	2	0	1
Virada 180 BT	1	1	2	0	0	0	0	3	0	1	0	1	1	0	0	0
Quant. de BT	3	6	14	6	13	3	3	8	3	6	7	3	15	3	1	2
Tempo Total(s)	388.14	730.85	1834.48	920.60	1773.47	344.53	389.31	683.83	389.66	626.86	354.56	346.24	1408.19	358.02	83.71	260.63
Tempo Comp.(ms)	5.55	12.54	38.11	19.90	38.36	3.73	6.55	9.39	4.55	5.94	4.13	4.06	22.72	4.25	1.11	2.07

Tabela 16 – Desempenho do planejamento Guloso com Fix and Optimize no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	6	13	49	5	48	12	0	43	7	13	12	0	42	10	2	0
Viradas 90	23	33	76	54	74	17	18	30	18	22	11	25	51	19	3	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	4	11	0	8	8	0	17	0	3	9	0	10	6	2	0
Viradas 90 BT	3	1	13	2	7	0	0	4	3	2	1	0	7	0	1	0
Virada 135 BT	0	2	1	0	0	2	0	1	0	3	4	0	8	2	0	0
Virada 180 BT	0	0	0	0	1	0	0	3	0	1	0	0	1	0	0	0
Quant. de BT	2	3	11	1	7	2	0	8	2	4	7	0	14	3	1	0
Tempo Total(s)	351.89	582.46	1531.04	823.58	1441.78	315.57	281.60	683.83	285.23	413.83	342.58	292.10	1143.53	332.99	83.71	203.78
Tempo Comp.(ms)	581.78	1411.22	4598.65	2970.21	4454.07	430.48	694.67	843.01	513.57	1466.83	409.03	593.83	3706.28	397.03	98.68	354.15

Tabela 17 – Desempenho do planejamento Guloso com Fix and Optimize no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	8	23	68	34	75	11	10	43	13	28	14	11	74	10	2	6
Viradas 90	21	42	93	69	89	21	23	30	27	38	11	42	67	20	3	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	3	6	11	5	19	5	2	17	4	5	9	5	15	6	2	4
Viradas 90 BT	2	5	20	10	12	0	3	4	4	6	1	1	11	1	1	2
Virada 135 BT	1	2	3	3	3	3	0	1	2	3	5	1	5	2	0	0
Virada 180 BT	0	1	1	2	0	0	0	3	0	1	0	0	5	0	0	1
Quant. de BT	3	5	16	9	12	2	2	8	4	7	7	2	16	3	1	2
Tempo Total(s)	352.32	733.52	1806.12	1172.26	1746.40	338.17	380.84	683.83	428.34	620.60	354.56	500.56	1397.34	348.99	83.71	266.88
Tempo Comp.(ms)	581.78	1411.22	4598.65	2970.21	4454.07	430.48	694.67	843.01	513.57	1466.83	409.03	593.83	3706.28	397.03	98.68	354.15

Tabela 18 – Desempenho do planejamento Guloso com algoritmo genético no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	7	16	30	22	21	2	2	49	7	14	12	4	29	10	2	0
Viradas 90	23	32	86	38	87	17	20	27	17	18	11	24	58	18	3	20
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	1	8	4	4	0	0	18	1	3	9	0	9	3	2	0
Viradas 90 BT	3	3	3	6	1	2	1	4	2	6	1	3	5	1	1	0
Virada 135 BT	0	1	0	2	2	0	0	0	1	1	4	0	7	1	0	0
Virada 180 BT	0	0	0	1	0	0	0	4	0	0	0	0	1	1	0	0
Quant. de BT	3	3	3	7	2	1	1	8	2	5	7	2	10	3	1	0
Tempo Total(s)	359.28	565.39	1432.33	839.31	1421.70	246.33	314.03	674.03	285.23	389.58	342.58	324.96	1128.65	316.63	83.71	211.78
Tempo Comp.(ms)	2044.97	4855.01	11514.14	6049.24	11430.20	1631.61	2483.81	3280.17	1883.53	2298.09	1702.06	1831.65	6770.78	1790.90	904.99	1181.68

Tabela 19 – Desempenho do planejamento Guloso com algoritmo genético no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	7	20	66	34	81	17	2	49	23	16	14	4	50	10	2	0
Viradas 90	23	39	138	41	114	21	20	27	25	20	11	24	79	19	3	20
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	4	16	11	19	9	0	18	6	5	9	0	20	3	2	0
Viradas 90 BT	3	5	12	7	6	1	1	4	8	5	1	3	8	2	1	0
Virada 135 BT	0	2	2	3	5	1	0	0	0	1	5	0	6	1	0	0
Virada 180 BT	0	1	1	1	1	0	0	4	0	1	0	0	1	1	0	0
Quant. de BT	3	5	9	8	10	3	1	8	6	6	7	2	12	3	1	0
Tempo Total(s)	359.28	694.24	2076.08	942.62	1923.52	363.28	314.03	674.03	459.76	430.01	354.56	324.96	1412.17	332.63	83.71	211.78
Tempo Comp.(ms)	2044.97	4855.01	11514.14	6049.24	11430.20	1631.61	2483.81	3280.17	1883.53	2298.09	1702.06	1831.65	6770.78	1790.90	904.99	1181.68

Tabela 20 – Desempenho do planejamento com Árvore no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	16	51	19	27	8	5	47	8	14	18	7	34	8	6	2
Viradas 90	25	37	92	50	75	17	18	31	25	22	15	22	60	16	4	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	3	7	4	5	2	0	18	2	5	12	1	5	3	0	1
Viradas 90 BT	1	2	7	1	1	3	4	6	1	1	1	1	8	1	0	0
Virada 135 BT	0	1	1	2	1	0	0	2	0	2	4	2	3	1	0	1
Virada 180 BT	0	1	2	1	0	2	0	2	0	1	1	0	1	1	1	0
Quant. de BT	1	3	8	3	2	3	2	9	2	5	7	3	9	2	1	1
Tempo Total(s)	342.43	621.39	1593.31	866.77	1328.09	309.07	331.49	722.50	338.45	415.49	410.88	328.33	1103.46	292.20	87.48	220.21
Tempo Comp.(ms)	1.03	3.37	15.09	5.99	13.73	0.82	1.62	3.89	1.35	2.12	0.70	1.06	7.47	0.92	0.10	0.30

Tabela 21 – Desempenho do planejamento Árvore no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	11	40	116	74	122	23	39	48	11	26	17	23	72	22	12	8
Viradas 90	25	61	139	80	95	24	22	34	33	36	20	22	85	21	7	17
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	0	7	31	12	28	8	12	18	4	9	10	8	22	11	3	3
Viradas 90 BT	4	10	19	11	11	6	9	9	4	6	2	5	6	3	1	2
Virada 135 BT	0	3	5	2	5	2	0	2	2	4	3	2	8	3	1	1
Virada 180 BT	2	2	3	2	4	1	1	3	0	2	1	0	3	1	1	0
Quant. de BT	3	8	20	11	18	7	6	11	4	9	7	6	17	6	3	2
Tempo Total(s)	398.88	990.00	2406.46	1355.06	1995.07	474.56	548.99	796.08	474.27	661.34	435.75	438.59	1554.35	456.65	166.59	252.95
Tempo Comp.(ms)	1.03	3.37	15.09	5.99	13.73	0.82	1.62	3.89	1.35	2.12	0.70	1.06	7.47	0.92	0.10	0.30

Tabela 22 – Desempenho do planejamento Árvore com Fix and Optimize no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	15	72	36	0	6	0	45	7	20	21	0	44	8	6	0
Viradas 90	25	31	73	40	84	17	19	24	18	18	12	29	58	16	4	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	1	7	9	0	1	0	19	2	8	9	0	7	3	0	0
Viradas 90 BT	1	2	10	2	0	4	0	5	2	5	3	0	7	1	0	0
Virada 135 BT	0	1	1	1	0	1	0	1	0	0	3	0	3	1	0	0
Virada 180 BT	0	1	1	1	0	0	0	3	0	0	1	0	1	1	1	0
Quant. de BT	1	3	11	4	0	3	0	9	2	5	7	0	11	2	1	0
Tempo Total(s)	342.43	560.17	1494.28	839.25	1311.09	295.18	289.60	664.44	289.41	408.32	377.80	324.10	1114.70	292.20	87.48	203.78
Tempo Comp.(ms)	277.23	1139.81	3716.96	1595.02	2905.85	165.53	271.56	974.52	187.03	268.27	158.75	197.73	775.56	164.33	18.33	48.92

Tabela 23 – Desempenho do planejamento Árvore com Fix and Optimize no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	14	47	139	42	102	17	23	45	12	33	19	15	74	14	7	6
Viradas 90	26	41	111	51	122	21	27	24	24	25	14	44	70	19	4	22
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	4	8	35	16	17	1	9	19	1	7	12	4	15	10	2	1
Viradas 90 BT	5	4	17	6	7	8	6	5	3	10	1	3	11	0	0	3
Virada 135 BT	1	2	6	4	5	1	1	1	1	2	4	0	8	2	0	1
Virada 180 BT	0	4	0	1	3	0	0	3	0	1	0	0	1	0	1	0
Quant. de BT	5	7	22	9	15	6	5	9	3	11	7	2	16	3	2	2
Tempo Total(s)	441.33	802.70	2201.77	1069.61	2060.50	389.64	521.90	664.44	360.76	581.48	392.09	520.69	1403.79	361.48	107.06	286.70
Tempo Comp.(ms)	277.23	1139.81	3716.96	1595.02	2905.85	165.53	271.56	974.52	187.03	268.27	158.75	197.73	775.56	164.33	18.33	48.92

Tabela 24 – Desempenho do planejamento Árvore com algoritmo genético no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	12	41	28	17	2	2	45	7	20	21	8	40	9	6	0
Viradas 90	24	35	89	37	80	17	19	24	18	20	12	26	58	16	4	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	0	10	9	3	0	0	19	2	4	9	2	10	3	0	0
Viradas 90 BT	0	1	12	4	1	2	1	5	2	4	3	1	6	1	0	0
Virada 135 BT	0	0	0	1	1	0	0	1	0	2	3	2	4	1	0	0
Virada 180 BT	0	2	0	1	1	0	0	3	0	1	1	0	1	1	1	0
Quant. de BT	1	2	11	6	3	1	1	9	2	5	7	3	9	2	1	0
Tempo Total(s)	326.43	570.53	1602.77	835.70	1363.75	246.33	306.03	664.44	289.41	423.60	377.80	367.46	1118.03	293.23	87.48	203.78
Tempo Comp.(ms)	968.98	1815.53	5583.45	2531.99	4949.51	861.29	1104.81	1852.08	1028.14	1230.36	822.83	942.02	3125.76	897.07	619.44	673.78

Tabela 25 – Desempenho do planejamento Árvore com algoritmo genético no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	34	106	38	67	14	2	45	13	25	21	8	83	9	6	0
Viradas 90	24	47	105	45	97	22	19	24	25	26	12	26	74	16	4	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	2	5	22	11	5	7	0	19	2	2	9	2	18	3	0	0
Viradas 90 BT	0	7	16	8	3	1	1	5	5	4	3	1	9	1	0	0
Virada 135 BT	0	1	4	3	1	3	0	1	0	3	3	2	4	1	0	0
Virada 180 BT	0	1	2	2	2	0	0	3	1	2	1	0	4	1	1	0
Quant. de BT	1	6	18	9	6	3	1	9	4	8	7	3	16	2	1	0
Tempo Total(s)	326.43	791.26	2010.86	1004.94	1610.53	375.82	306.03	664.44	400.16	505.40	377.80	367.46	1443.19	293.23	87.48	203.78
Tempo Comp.(ms)	968.98	1815.53	5583.45	2531.99	4949.51	861.29	1104.81	1852.08	1028.14	1230.36	822.83	942.02	3125.76	897.07	619.44	673.78

Tabela 26 – Desempenho do planejamento considerando o gasto de energia no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	15	21	68	31	56	21	7	60	7	12	14	14	35	12	2	12
Viradas 90	19	32	68	40	66	16	17	34	18	21	11	24	52	18	3	15
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	5	5	17	9	8	9	0	18	1	1	9	0	15	5	2	5
Viradas 90 BT	1	5	14	5	12	3	4	5	2	4	1	3	6	0	1	3
Virada 135 BT	3	3	1	1	2	1	1	2	1	1	5	0	7	1	0	1
Virada 180 BT	1	0	1	2	3	1	0	4	0	0	0	1	0	1	0	0
Quant. de BT	5	5	15	7	15	5	3	10	2	5	7	3	14	3	1	4
Tempo Total(s)	392.09	643.46	1568.48	888.98	1520.39	367.95	341.83	781.95	293.23	383.33	354.56	353.48	1145.02	322.87	83.71	273.99
Tempo Comp.(ms)	2.45	2.94	10.40	3.73	11.14	2.44	2.23	5.30	2.30	2.48	2.44	2.18	5.93	2.34	2.02	2.11

Tabela 27 – Desempenho do planejamento considerando o gasto de energia no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	15	21	53	27	59	21	7	60	7	22	14	14	48	12	2	12
Viradas 90	19	33	81	40	66	16	17	34	18	19	11	24	62	18	3	15
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	5	5	14	10	10	9	0	18	1	4	9	0	13	5	2	5
Viradas 90 BT	1	5	11	5	14	3	4	5	2	5	1	3	9	0	1	3
Virada 135 BT	3	3	4	4	2	1	1	2	1	2	5	0	7	1	0	1
Virada 180 BT	1	0	0	2	3	1	0	4	0	3	0	1	0	1	0	0
Quant. de BT	5	5	13	7	16	5	3	10	2	8	7	3	15	3	1	4
Tempo Total(s)	392.09	651.46	1619.89	920.66	1558.04	367.95	341.83	781.95	293.23	468.39	354.56	353.48	1256.64	322.87	83.71	273.99
Tempo Comp.(ms)	2.45	2.94	10.40	3.73	11.14	2.44	2.23	5.30	2.30	2.48	2.44	2.18	5.93	2.34	2.02	2.11

Tabela 28 – Desempenho do planejamento LeftHand

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	7	10	30	21	26	15	0	31	7	13	19	5	51	9	2	4
Viradas 90	29	38	106	48	81	20	19	31	33	40	15	28	60	19	4	18
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	3	2	9	5	3	7	0	18	1	5	11	0	21	3	2	3
Viradas 90 BT	0	2	2	1	2	0	0	3	0	1	0	1	4	0	0	0
Virada 135 BT	1	2	3	1	3	1	0	1	1	2	3	0	3	1	0	1
Virada 180 BT	0	0	1	2	0	1	0	2	0	1	1	2	1	0	0	1
Quant. de BT	2	2	5	4	3	2	0	10	1	5	7	2	9	1	1	2
Tempo Total(s)	392.92	608.82	1644.69	867.20	1397.05	338.49	289.60	678.41	390.87	558.46	387.91	365.63	1186.48	291.05	83.71	252.63
Tempo Comp.(ms)	3.31	4.00	7.78	3.24	4.69	2.33	2.29	6.49	2.34	2.54	2.46	2.57	4.73	2.34	2.35	2.16

Tabela 29 – Desempenho do planejamento Espiral.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	13	13	71	18	46	14	7	26	7	15	12	15	53	10	6	12
Viradas 90	23	33	84	40	75	16	16	26	17	18	10	22	59	20	4	15
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	7	4	23	10	10	2	0	16	1	6	9	4	14	6	0	3
Viradas 90 BT	3	1	8	5	7	6	4	2	2	3	1	3	8	0	0	4
Virada 135 BT	1	2	5	2	2	2	1	0	1	2	4	2	6	2	0	1
Virada 180 BT	0	0	1	1	2	1	0	3	0	2	0	0	1	0	1	1
Quant. de BT	5	3	15	7	9	7	3	9	2	7	7	4	14	3	1	4
Tempo Total(s)	418.57	582.46	1727.77	879.72	1504.25	364.72	333.83	608.61	285.23	431.15	334.58	377.23	1231.45	340.99	87.48	281.63
Tempo Comp.(ms)	3.43	4.04	14.30	4.53	8.16	3.54	3.18	6.68	3.23	2.84	2.47	2.26	5.98	2.43	2.32	2.16

Tabela 30 – Desempenho do planejamento Zigue-Zague no melhor caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	7	18	35	25	74	18	8	49	10	12	22	11	42	16	6	5
Viradas 90	28	36	74	48	81	20	18	42	18	14	25	22	65	19	4	17
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	4	4	10	6	13	7	2	21	6	5	10	5	8	4	0	2
Viradas 90 BT	1	1	0	1	5	1	4	2	1	1	3	1	5	2	0	1
Virada 135 BT	0	0	0	2	1	2	0	1	0	5	5	1	5	0	0	0
Virada 180 BT	0	0	1	1	0	0	0	1	2	0	0	0	2	2	1	2
Quant. de BT	1	2	3	3	5	3	2	7	3	5	10	2	11	4	1	2
Tempo Total(s)	382.74	585.45	1341.50	869.16	1524.49	354.04	346.77	764.39	338.87	367.33	516.00	340.56	1190.36	353.19	87.48	249.49
Tempo Comp.(ms)	3.25	4.17	5.96	3.56	8.01	3.30	3.22	5.53	2.56	2.64	2.85	2.30	7.29	3.01	2.68	2.42

Tabela 31 – Desempenho do planejamento Zigue-Zague no pior caso.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST1	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	7	21	70	24	37	20	3	48	19	10	31	14	72	17	6	5
Viradas 90	28	42	95	88	94	28	30	45	27	35	23	24	81	19	4	17
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	4	6	21	8	11	5	0	21	6	6	15	5	14	4	0	2
Viradas 90 BT	1	3	7	4	4	4	1	3	3	0	3	2	13	4	0	1
Virada 135 BT	0	2	1	0	1	3	0	1	0	2	5	1	5	0	0	0
Virada 180 BT	0	0	3	2	0	1	0	1	2	0	0	0	4	2	1	2
Quant. de BT	1	3	10	5	4	4	1	6	4	5	11	2	19	4	1	2
Tempo Total(s)	382.74	690.92	1746.75	1229.02	1563.69	460.02	395.06	789.00	442.54	501.62	546.12	367.66	1524.44	370.22	87.48	249.49
Tempo Comp.(ms)	3.25	4.17	5.96	3.56	8.01	3.30	3.22	5.53	2.56	2.64	2.85	2.30	7.29	3.01	2.68	2.42

Tabela 32 – Desempenho do planejamento Wavefront.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	91	116	89	86	379	89	27	47
Passos BT	3	0	16	44	4	1	0	23	5	3	17	5	13	12	3	10
Viradas 90	40	79	173	88	142	35	50	46	33	49	19	41	120	36	4	19
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	3	0	10	16	4	1	0	20	1	2	10	0	9	5	2	5
Viradas 90 BT	0	0	4	0	0	0	0	4	2	0	1	2	2	1	1	2
Virada 135 BT	1	0	4	4	0	1	0	1	1	3	6	0	7	1	0	2
Virada 180 BT	1	0	4	2	2	0	0	2	1	0	3	2	2	0	0	0
Quant. de BT	3	0	11	7	2	1	0	10	3	3	10	3	10	4	2	4
Tempo Total(s)	494.97	865.76	2271.84	1318.78	1839.95	389.29	537.60	810.32	429.34	579.22	492.20	483.99	1595.93	469.41	99.10	305.83
Tempo Comp.(ms)	2.95	2.96	11.65	5.01	5.43	3.33	3.27	6.63	3.38	3.17	2.73	2.35	5.67	2.49	3.16	3.15

Tabela 33 – Desempenho do planejamento STC.

Instancias	BSA	TASP	GAYA	GAYA2	NNY3	PTZ	PTZ2	STC	STC1	STC2	WFS	YCM	YCM2	TEST	TEST2	TEST3
Passos	106	223	598	349	615	80	130	159	90	116	89	86	379	89	27	47
Passos sem STC	19	20	51	22	24	22	11	80	0	24	42	28	49	18	16	14
Passos BT	22	26	60	43	45	2	2	68	0	16	26	15	74	15	6	14
Viradas 90	26	45	111	67	95	15	17	53	34	40	35	24	73	15	4	16
Viradas 180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Viradas 45 BT	4	9	16	14	18	0	0	19	0	5	17	6	17	3	1	2
Viradas 90 BT	5	2	5	1	4	1	2	4	0	3	1	5	6	4	0	4
Virada 135 BT	0	1	0	2	2	0	0	5	0	3	2	0	3	1	1	0
Virada 180 BT	2	1	2	1	0	1	0	4	0	0	1	0	2	0	0	3
Quant. de BT	5	4	9	5	6	1	1	11	0	6	7	4	13	3	1	5
Tempo Total(s)	482.98	759.31	1842.61	1123.96	1670.04	256.90	309.40	1059.12	368.24	606.82	633.21	430.55	1393.83	328.59	108.21	320.17
Tempo Comp.(ms)	2.74	3.52	11.90	4.95	7.49	2.62	3.32	7.99	3.28	3.08	2.96	2.61	8.21	2.65	2.58	3.46