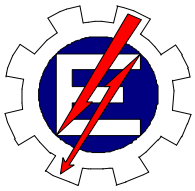


UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENERGIA ELÉTRICA

Cláudio Rodrigo Torres

SISTEMA INTELIGENTE BASEADO NA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL Eτ PARA CONTROLE E NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM UM AMBIENTE NÃO ESTRUTURADO

Agosto de 2010

Itajubá - MG

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Cláudio Rodrigo Torres

SISTEMA INTELIGENTE BASEADO NA LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL Eτ PARA CONTROLE E NAVEGAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM UM AMBIENTE NÃO ESTRUTURADO

Tese submetida ao programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Doutor em Ciências em Engenharia Elétrica.

Área de Concentração: Automação e Sistemas Elétricos.

Orientador: Prof. Dr. Germano Lambert Torres
Orientador: Prof. Dr. Jair Minoro Abe

Agosto de 2010

Itajubá - MG

Aos meus pais, que com simplicidade e sabedoria souberam construir uma família da qual me orgulho de fazer parte. E graças ao apoio dessa família, pude chegar até aqui.

AGRADECIMENTOS

A única certeza que tenho é a de que sozinho nunca conseguiria realizar este trabalho. São muitas as pessoas a quem devo meus agradecimentos. Seria impossível mencionar todos aqui.

Gostaria de expressar meus especiais agradecimentos aos amigos Prof. Dr. Germano Lambert Torres, Prof. Dr. Jair Minoro Abe e Prof. Dr. João Inácio da Silva Filho. Pessoas que me deram oportunidades e orientações fundamentais para a minha formação profissional e pessoal.

Também gostaria de agradecer à Prof.ª Dr.ª Helga Gonzaga Martins, ao Prof. Dr. Maurílio Pereira Coutinho e ao Prof. Dr. Ronaldo Rossi pelas sugestões para a escrita deste texto.

Importante, também, foi a ajuda de Sheila Souza. Sem seu conhecimento e dedicação não seria possível apresentarmos os resultados expostos neste texto.

A eles meu muito obrigado.

RESUMO

Apresenta-se nesta tese um Sistema de Navegação e Controle de Robôs Móveis Autônomos baseado na Lógica Paraconsistente Anotada Evidencial Eτ através da utilização das Redes Neurais Artificiais Paraconsistentes. Esse sistema se divide em três módulos: Subsistema de Sensoriamento, Subsistema de Planejamento e Subsistema Mecânico. O funcionamento independente, mas, interligado, desses três módulos, constituem um robô móvel autônomo capaz de encontrar um ponto destino pré-determinado num ambiente não estruturado. Neste trabalho optou-se por dar maior ênfase às implementações dos Subsistemas de Sensoriamento e de Planejamento onde são aplicadas as técnicas de construção dos algoritmos baseados na Lógica Paraconsistente Anotada Evidencial Eτ. Os resultados envolvendo os algoritmos nesses dois Subsistemas mostraram-se muito promissores, capacitando-os a serem empregados com êxito em sistemas de navegação móvel autônoma. Como trabalho futuro, deixa-se a sugestão de construção do Subsistema Mecânico sobre uma plataforma similar com a construída no Robô Emmy II que, em trabalho anterior, utilizou procedimentos de controle baseados na Lógica Paraconsistente Evidencial Eτ.

Palavras Chave

1. Lógica Paraconsistente Anotada; 2. Redes Neurais Artificiais Paraconsistentes; 3. Sistema de Controle; 4. Sistema Inteligente; 5. Robô Móvel Autônomo.

ABSTRACT

This thesis presents a Navigation and Control System of an Autonomous Mobile Robot based on the Evidential Paraconsistent Annotated Logic Eτ through the use of the Paraconsistent Artificial Neural Network. This system is divided in three modules: Sensing Subsystem, Planning Subsystem and Mechanical Subsystem. The independent but interconnected functioning of these three modules makes an autonomous mobile robot able to find a predetermined point in a non-structured environment. This work emphasizes the implementation of the Sensing Subsystem and the Planning Subsystem where is applied techniques of algorithms construction based on the Evidential Paraconsistent Annotated Logic Eτ. The results reached by these algorithms seems to be promising, making the algorithms able to be used in autonomous mobile navigation systems. As a future work, we suggest the construction of the Mechanical Subsystem in a mechanical platform similar to the one used in the Emmy II robot. The Emmy II robot, in a previous work, had its control based on the Evidential Paraconsistent Annotated Logic Eτ.

Key Words

1. Paraconsistent Annotated Logic; 2. Paraconsistent Artificial Neural Network; 3. Control System; 4. Intelligent System; 5. Autonomous Mobile Robot.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIMBOLOS

μ	Grau de Evidência Favorável
λ	Grau de Evidência Contrária
G_c	Grau de Certeza
G_{ct}	Grau de Contradição
V	Estado verdadeiro
F	Estado falso
T	Estado inconsistente
$\perp$	Estado paracompleto
$QV \rightarrow T$	Quase-verdadeiro tendendo ao inconsistente
$QV \rightarrow \perp$	Quase-verdadeiro tendendo ao paracompleto
$QF \rightarrow T$	Quase-falso tendendo ao inconsistente
$QF \rightarrow \perp$	Quase-falso tendendo ao paracompleto
$QT \rightarrow V$	Quase-inconsistente tendendo ao verdadeiro
$QT \rightarrow F$	Quase-inconsistente tendendo ao falso
$Q\perp \rightarrow V$	Quase-paracompleto tendendo ao verdadeiro
$Q\perp \rightarrow F$	Quase-paracompleto tendendo ao falso
V_{cve}	Valor de controle de veracidade
V_{cfa}	Valor de controle de falsidade
V_{cic}	Valor de controle de inconsistência
V_{cpa}	Valor de controle de paracompleteza

φ	Intervalo de Certeza
$G_{ve_{\max}}$	Grau de Veracidade Máximo
$G_{fa_{\max}}$	Grau de Falsidade Máximo
$\varphi_{(\pm)}$	Intervalo de Certeza Sinalizado
G_{C_r}	Grau de Certeza Resultante
G_{Cest}	Grau de Certeza Estimado
μ_i	Grau de Evidência Favorável Inicial
μ_f	Grau de Evidência Favorável Final
λ_i	Evidência Contrária Inicial
λ_f	Grau de Evidência Contrária Final
G_{CR}	Grau de Certeza Rea
G_{Crr}	Grau de Certeza Resultante Real
NAP	Sistema ou Nó de Análise Paraconsistente
RNA	Rede Neural Artificial
μ_E	Grau de Evidência Resultante
μ_{Er}	Grau de Evidência Resultante Real
φ_E	Intervalo de Evidência Resultante
μ_{ctr}	Grau de Contradição Normalizado
μ_{Es}	Grau de Evidência de Saída
e	Grau de Especialidade
CNAPa	Célula Neural Artificial Paraconsistente Analítica
CNAPpa	Célula Neural Artificial Paraconsistente de Passagem
CNAPmax	Célula Neural Artificial Paraconsistente de Conexão Lógica Simples no Processo de Maximização (OU)

Ft_{ct}	Fator de Tolerância à Contradição
Ft_c	Fator de Tolerância à Certeza
a	Tamanho do lado do quadrado que compõe a célula
β	Ângulo de abertura do sensor
n	Número de vezes que o Grau de Evidência é calculado
Dmax	Distância máxima entre o obstáculo e o sensor que o Subsistema de Sensoriamento considera
Dmin	Distância mínima entre o obstáculo e o sensor que o Subsistema de Sensoriamento considera
D	Distância do obstáculo detectada pelo sensor
α	Ângulo entre o sensor e o eixo horizontal do ambiente
Xa	Coordenada X onde o robô se encontra
Ya	Coordenada Y onde o robô se encontra
μ1	Grau de Evidência Favorável proporcional a distância entre o sensor e o obstáculo
μ2	Grau de Evidência Favorável proporcional a posição da célula no arco formado pelo “cone” gerado pelo sensor de ultrassom
μ3	Grau de Evidência Favorável atual armazenado no banco de dados do sistema
Xo	Coordenada X de origem
Yo	Coordenada Y de origem
Xd	Coordenada X de destino
Yd	Coordenada Y de destino
NC	Número de células que o Subsistema de Planejamento deve analisar antes de definir quais ações o robô deve tomar
LE	Valor limite do Grau de Evidência Favorável para que o Subsistema de Planejamento considere a célula ocupada

SM	Sequência de movimentos que a plataforma mecânica deve executar

CAPÍTULO 1 - INTRODUÇÃO

1.1 Considerações iniciais

Propõe-se neste trabalho o sistema de navegação e controle de um robô móvel autônomo. Esse sistema divide-se em três partes: Subsistema de Sensoriamento, Subsistema de Planejamento e Subsistema Mecânico. O sistema de navegação e controle composto por estes três subsistemas permite a um robô móvel autônomo encontrar uma posição destino em um ambiente não estruturado.

As ideias aqui apresentadas estendem estudos do Robô Móvel Autônomo Emmy I (DA SILVA FILHO, 1999), (DA SILVA FILHO & ABE, 1999), (DA SILVA FILHO & ABE, 1999b), (DA SILVA FILHO & ABE, 1999c), (DA SILVA FILHO & ABE, 2001), (DA SILVA FILHO & ABE, 2001b), (DA SILVA FILHO & ABE, 2001d), (DA SILVA FILHO, TORRES & ABE, 2006) e do Robô móvel Autônomo Emmy II (TORRES, 2004), (TORRES, ABE, & LAMBERT-TORRES, 2004), (TORRES, ABE & LAMBERT-TORRES, 2005), (TORRES, ABE & LAMBERT-TORRES, 2005b), (TORRES, ABE & LAMBERT-TORRES, 2006), (TORRES, LAMBERT-TORRES, SILVA & ABE, 2006), (ABE, TORRES, LAMBERT-TORRES, NAKAMATSU & KONDO, 2006), (ABE, TORRES, LAMBERT-TORRES, NAKAMATSU & KONDO, 2006b), (TORRES, ABE, LAMBERT-TORRES, 2007), (TORRES & BOMBACINI, 2007). O robô Emmy I é capaz de desviar de obstáculos num ambiente não estruturado e seu controle se baseia na Lógica Paraconsistente Anotada Evidencial Eτ. O robô Emmy II possui, basicamente, as mesmas características que o robô Emmy I, mas sua estrutura mecânica é mais compacta e seu sistema de controle possui algumas diferenças em relação ao robô Emmy I. O sistema de navegação e controle proposto nesta tese deve ser utilizado na construção do Robô Móvel Autônomo Emmy III (ABE,

LAMBERT-TORRES, DA SILVA FILHO, TORRES, MARTINS, 2007), (TORRES, ABE, LAMBERT-TORRES, DA SILVA FILHO & MARTINS, 2009), (ABE, TORRES, LAMBERT-TORRES & MARTINS, 2009).

Neste capítulo apresentam-se, sucintamente, os elementos que compõem o sistema de navegação e controle. A parte nuclear deste trabalho é composta pelos Subsistemas de Sensoriamento e de Planejamento do robô, descritos nos capítulos 5 e 6, respectivamente.

Um robô móvel autônomo deve ser capaz de se locomover num ambiente não estruturado. O projeto de um robô com estas características é uma típica situação em que se deve lidar com contradições (inconsistências), incertezas ou paracompletezas. Os Sistemas Inteligentes tradicionais são usualmente baseados na lógica clássica ou em algumas de suas extensões. E, em certas situações, quando se defronta com incertezas, contradições (inconsistência) ou paracompletezas, muitas vezes algumas simplificações ou adaptações são necessárias para que o sistema continue funcionando, podendo torná-lo complexo ou, paradoxalmente, ineficiente.

O projeto proposto tem como base a Lógica Paraconsistente Anotada Evidencial Eτ. A utilização deste tipo de lógica permite se lidar com sinais elétricos que expressam situações incertas, contraditórias ou paracompletas de forma não trivial.

1.2 Robôs móveis

Os robôs móveis podem ser divididos em três categorias: teleoperados, semiautônomos e autônomos. Nos robôs móveis teleoperados um operador define todos os movimentos que o robô deve executar. Já nos robôs móveis semiautônomos, um operador indica o macrocomando a ser executado e o robô o executa sozinho. E os robôs móveis autônomos realizam suas tarefas sozinhos, tomando suas próprias decisões.

Existem também os robôs móveis terrestres, aquáticos e aéreos.

Esta tese trata de um sistema de controle e navegação de um robô móvel autônomo terrestre.

Os robôs móveis são principalmente utilizados em ambientes nocivos e/ou perigosos ao ser humano. Como exemplo prático desse tipo de situação, pode-se citar os robôs utilizados para exploração submarina (Autonomous Underwater Vehicle - AUV), os robôs enviados para a exploração de Marte e o Robô Pioneer (MAIMONE et. al., 1998), utilizado para explorar as instalações de Chernobyl. Desde a Segunda Guerra Mundial existe a utilização de robôs móveis por militares. As bombas voadoras utilizadas durante a Segunda Guerra são um exemplo. Pode-se também citar o Robart I, II e III (SPAWAR…), utilizados pela marinha norte-americana para patrulhamento. O avião espião Predator (AIRFORCE...), utilizado na guerra do Golfo, é um exemplo de veículo aéreo não tripulado (Unmanned Aerial Vehicle – UAV).

Os robôs móveis também são encontrados na indústria. Eles podem ser utilizados para transporte de carga (Automated Guided Vehicle - AGV) (NEHMZOW, 2000), limpeza (PRASSLER, RITTER, SCHAEFFER & FIORINI, 2000) e outras tarefas.

Outra aplicação interessante de um robô móvel é o Helpmate (HELPMATE ROBOTICS...), utilizado em hospitais para transporte de refeições, medicamento e roupas sujas.

Encontram-se robôs móveis como produtos de consumo, seja para o entretenimento ou para realização de alguns trabalhos como limpeza (ELECTROLUX...) (SONY...).

Existem diversas técnicas de controle e navegação de robôs móveis autônomos. E muitas outras estão sendo estudadas. Nesta tese apresenta-se um sistema de controle de navegação e controle que possui como principal diferencial o uso da Lógica Paraconsistente Anotada Evidencial Eτ.

1.3 Descrição geral do sistema de navegação e controle proposto

O Sistema de Navegação e Controle divide-se em três partes: Subsistema Mecânico, Subsistema de Planejamento e Subsistema de Sensoriamento.

Chama-se de Subsistema Mecânico a estrutura física do robô. Essa estrutura deve ser capaz de carregar todos os sensores e dispositivos de controle que compõe o Sistema de navegação e Controle. O Subsistema Mecânico deve ser capaz de fornecer as informações necessárias para que o Sistema de Navegação e Controle funcione adequadamente, além de executar os movimentos determinados pelo Subsistema de Planejamento.

O primeiro protótipo do robô proposto, ainda em construção, compõe-se, basicamente, de uma placa metálica com dimensão de 400 mm de largura e 400 mm de comprimento. Três rodas suportam essa estrutura, sendo uma livre e as outras duas conectadas a servomotores. Os servomotores instalados na base são responsáveis pela movimentação do robô.

O Subsistema de Planejamento é responsável por gerar a sequência de ações que a estrutura mecânica deve realizar para sair de uma posição origem e alcançar uma posição destino. Neste trabalho considera-se que o Subsistema de Planejamento mapeia o ambiente em torno do robô, dividindo-o em coordenadas. Assim, no Sistema de Navegação e Controle o Subsistema de Sensoriamento é responsável por informar o Sistema de Planejamento quais coordenadas estão obstruídas.

O Subsistema de Planejamento solicita uma série de ações ao Subsistema Mecânico de tal forma que a plataforma mecânica seja capaz de deixar a coordenada onde se encontra e passando, por diversas outras coordenadas, alcance a coordenada destino.

Nessas ações o Subsistema de Sensoriamento deve manipular dados oriundos dos mais diversos tipos de sensores que monitoram o ambiente em torno do robô. Esses dados, muitas

vezes, são imprecisos e/ou contraditórios, devendo receber um tratamento adequado em tempo real. Em diversos trabalhos como em (ALMEIDA PRADO, ABE & SCALZITTI, 2005), (ÁVILA, 1996), (ÁVILA, ABE & PRADO, 1997), (FERRARA, 2003), (MARIO, 2003) (MARIO, 2006), (MARIO, 2007) (MARTINS, 2003), (PRADO, 1996), (ROSA E SILVA, 2005) a Lógica Paraconsistente Evidencial Eτ e as Redes Neurais Artificiais Paraconsistentes se mostram uma boa ferramenta para lidar com incertezas, contradições (inconsistência) ou paracompletezas. Por isso, neste trabalho elas são utilizadas para compor o Subsistema de Sensoriamento e o Subsistema de Planejamento.

A figura 1.1 mostra a estrutura do sistema de controle e navegação proposto.

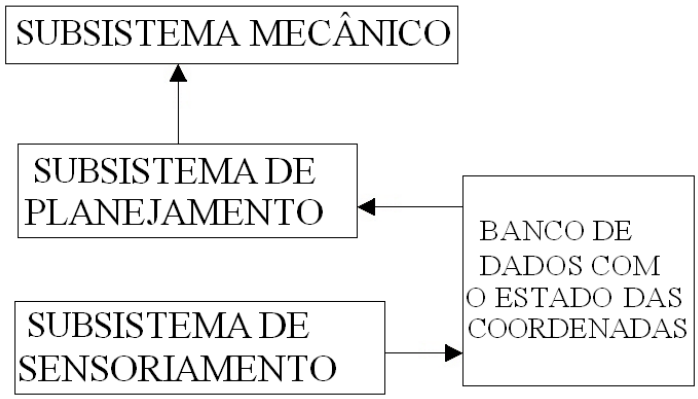


Figura 1.1 Estrutura do sistema de controle e navegação proposto

Observa-se que o esquema geral do Sistema de Navegação e Controle utiliza o Subsistema de Sensoriamento para alimentar um banco de dados com as informações sobre as condições do ambiente em torno do robô. Para cada coordenada, o banco de dados armazena informações sobre o seu estado, informando se a mesma está ocupada ou não.

Por sua vez, o Subsistema de Planejamento, antes de gerar a sequência de ações que o Subsistema Mecânico deve executar para alcançar a posição destino, consulta o banco de dados. Assim, ele é capaz de gerar e informar ao Sistema de Navegação e Controle uma trajetória que desvia dos obstáculos existentes no ambiente em torno do robô.

CAPÍTULO 2 - LÓGICA PARACONSISTENTE ANOTADA EVIDENCIAL Eτ

Apresentam-se neste capítulo os conceitos básicos da Lógica Paraconsistente Anotada Evidencial Eτ. O conhecimento desses conceitos é fundamental para o entendimento sistema de navegação e controle proposto neste trabalho. Pois, o funcionamento do mesmo baseia-se nas Redes Neurais Artificiais Paraconsistentes.

2.1 Lógica paraconsistente anotada evidencial Eτ

Antes de apresentar a estrutura da Rede Neural Artificial Paraconsistente utilizada no Subsistema de Sensoriamento e Planejamento, necessita-se apresentar os conceitos básicos da Lógica Paraconsistente Anotada Evidencial Eτ e das Células Neurais Artificiais Paraconsistentes.

A Lógica Paraconsistente, nos padrões de rigor atuais, surgiu em 1948, com o lógico polonês Stanislaw Jaśkowski (1906 – 1965) e com o lógico brasileiro Newton Carneiro Affonso da Costa em 1954 (1929 -), que de forma independente apresentaram as primeiras ideias da Lógica Paraconsistente. A Lógica Paraconsistente permite trabalhar com contradições de forma não trivial.

Em (SUBRAHMANIAN, 1987) apresentou-se pela primeira vez uma linguagem de programação baseada na Lógica Paraconsistente.

Em 1992 foi estudada a Lógica Paraconsistente Anotada Evidencial Eτ (ABE, 1992), (ABE, 1997), (ABE, 2001), (ABE & AKAMA, 1999), (ABE, ÁVILA, & NAKAMATSU, 1999), (ABE & DA SILVA FILHO, 1998), (AKAMA & ABE, 2000), (DA COSTA, ABE, & SUBRAHMANIAN, 1991), (DA COSTA, ABE, DA SILVA FILHO, MUROLO & LEITE,

1999), (DA COSTA, PRADO, ABE, ÁVIVA & RILLO, 1995), (DA SILVA FILHO, 1997), (NAKAMATSU, ABE & SUZUKI, 1999), (NAKAMATSU, ABE & SUZUKI, 2000), (SCALZITTI, DA SILVA FILHO & ABE, 2001), (SYLVAN & ABE, 1996). Os conceitos das Redes Neurais Artificiais Paraconsistentes (DA SILVA FILHO & ABE, 2001c), (DA SILVA FILHO; ABE & LAMBERT-TORRES, 2008) se baseiam nessa lógica.

Na Lógica Paraconsistente Anotada Evidencial Eτ, para cada proposição P associa-se um Grau de Evidência Favorável (μ) (ou crença favorável ou ainda outras leituras dependendo da aplicação) e um Grau de Evidência Contrária (λ). Os Graus de Evidência são valores reais entre 0 e 1 que denotam, no caso do Grau de Evidência Favorável, a crença ou a evidência de que a proposição P seja verdadeira. E no caso do Grau de Evidência Contrária o intervalo real fechado entre 0 e 1 denota a descrença ou a evidência de que a proposição P não seja verdadeira.

Define-se o Grau de Certeza como:

$$G_c = \mu - \lambda \qquad (2.1)$$

Também, define-se o Grau de Contradição da seguinte maneira:

$$G_{ct} = (\mu + \lambda) - 1 \qquad (2.2)$$

Dependendo dos valores dos Graus de Evidência podem-se ter quatro estados extremos: verdadeiro, falso, inconsistente e paracompleto.

O estado verdadeiro (V) acontece quando $\mu = 1{,}0$ e $\lambda = 0{,}0$.

O estado falso (F) acontece quando $\mu = 0{,}0$ e $\lambda = 1{,}0$.

O estado inconsistente (T) acontece quando $\mu = 1{,}0$ e $\lambda = 1{,}0$.

O estado paracompleto ($\perp$) acontece quando $\mu = 0{,}0$ e $\lambda = 0{,}0$.

A figura 2.1 mostra graficamente os estados acima descritos.

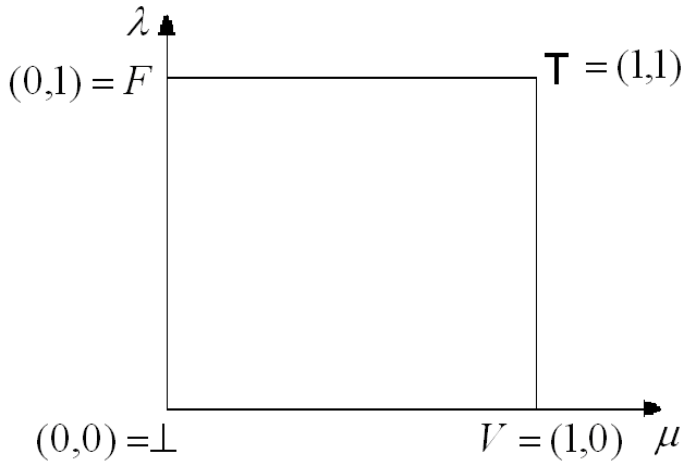


Figura 2.1 Representação gráfica dos estados extremos

Os conceitos das Redes Neurais Artificiais Paraconsistentes (DA SILVA FILHO; ABE & LAMBERT-TORRES, 2008) têm origem no Algoritmo Paranalisador proposto em (DA SILVA FILHO, 1999), (ABE & DA SILVA FILHO, 2001), (ABE & DA SILVA FILHO, 2001b), (ABE & DA SILVA FILHO, 2003). A figura 2.2 apresenta graficamente o Algoritmo Paranalisador.

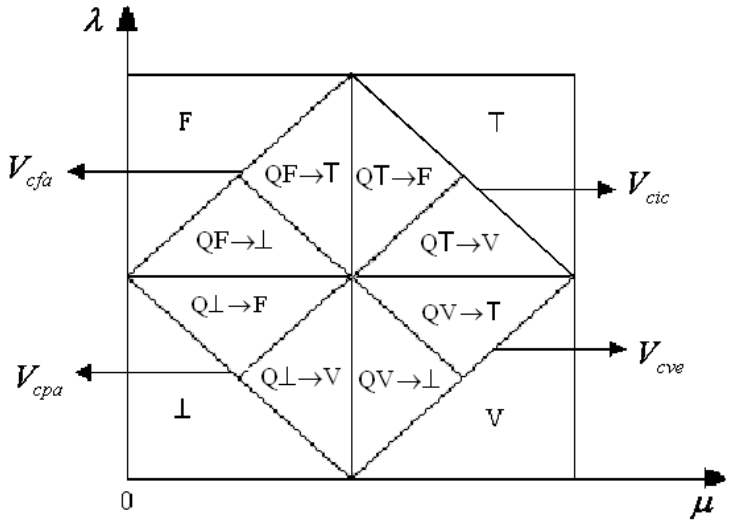


Figura 2.2 Algoritmo paranalisador

Observa-se, então, que com o Algoritmo Paranalisador, além dos quatro estados extremos, é possível se determinar mais oito estados não extremos. São eles:

- QV→T – Quase-verdadeiro tendendo ao inconsistente.

- QV→⊥ - Quase-verdadeiro tendendo ao paracompleto.
- QF→T - Quase-falso tendendo ao inconsistente.
- QF→⊥ - Quase-falso tendendo ao paracompleto.
- QT→V - Quase-inconsistente tendendo ao verdadeiro.
- QT→F - Quase-inconsistente tendendo ao falso.
- Q⊥→V - Quase-paracompleto tendendo ao verdadeiro.
- Q⊥→F - Quase-paracompleto tendendo ao falso.

No Algoritmo Paranalisador também existem quatro valores de controle externos:

- V_{cve} – Valor de controle de veracidade, $0 \leq V_{cve} \leq 1$
- V_{cfa} – Valor de controle de falsidade, $-1 \leq V_{cfa} \leq 0$
- V_{cic} – Valor de controle de inconsistência, $0 \leq V_{cic} \leq 1$
- V_{cpa} – Valor de controle de paracompleteza, $-1 \leq V_{cpa} \leq 0$

O tamanho das regiões que representam os estados extremos e não extremos no Algoritmo Paranalisador dependem desses valores de controle externos. A figura 2.3 mostra o Algoritmo Paranalisador quando, por exemplo, se varia V_{cic}.

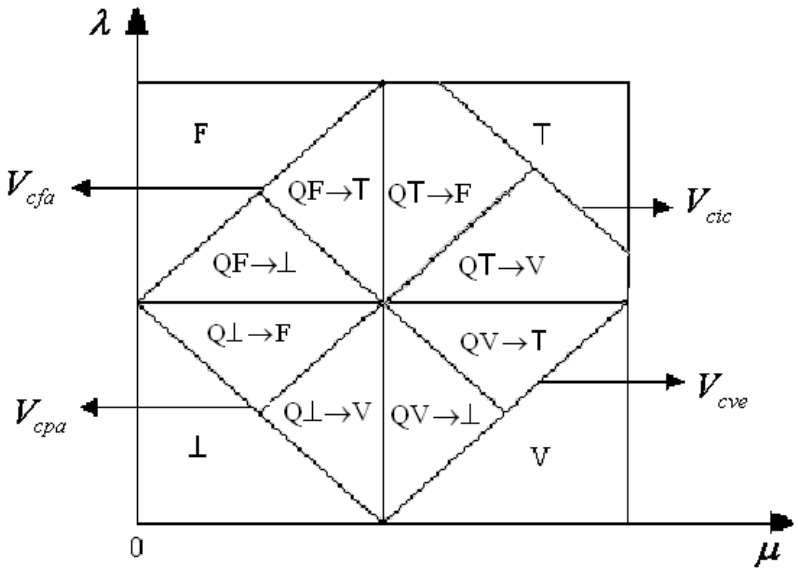


Figura 2.3 Algoritmo paranalisador com uma mudança em V_{cic}

2.2 Transformações lineares no Quadrado Unitário e Reticulado τ

Em (SCALZITTI, DA SILVA & ABE 2001) apresenta-se funções para se transformar valores representados pelo Quadrado Unitário no Plano Cartesiano para valores representados pelo Reticulado com Grau de Certeza e Grau de Contradição.

A partir do Quadrado Unitário pode-se obter o Reticulado através de uma mudança de escala, seguida de uma rotação e de uma translação.

Consegue-se a mudança de escala através da seguinte transformação linear:

$$T_1(\mu,\lambda) = \left(\sqrt{2}x, \sqrt{2}y\right)$$

Mostra-se abaixo sua correspondente matriz.

$$\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

A figura 2.4 representa essa mudança de escala.

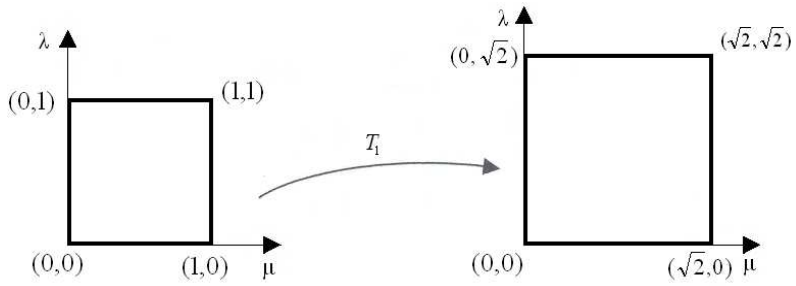


Figura 2.4 - Mudança de Escala

A figura 2.5 mostra a rotação de 45^o mantendo a coordenada de origem.

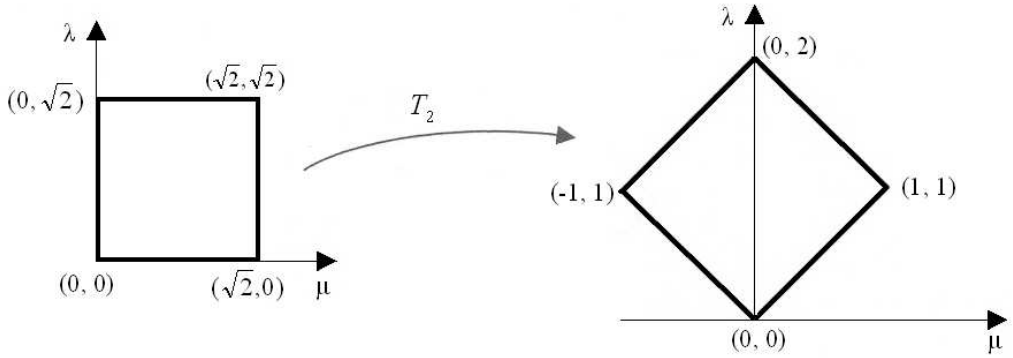


Figura 2.5 - Rotação

Consegue-se esta rotação através da seguinte transformação linear.

$$T_2(\mu,\lambda)=\left(\frac{\sqrt{2}}{2}\mu-\frac{\sqrt{2}}{2}\lambda,\frac{\sqrt{2}}{2}\mu+\frac{\sqrt{2}}{2}\lambda\right)$$

Cuja correspondente matriz é:

$$\begin{bmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}$$

A figura 2.6 mostra a translação dada pela seguinte transformação linear:

$$T_3(x,y)=(x,y-1)$$

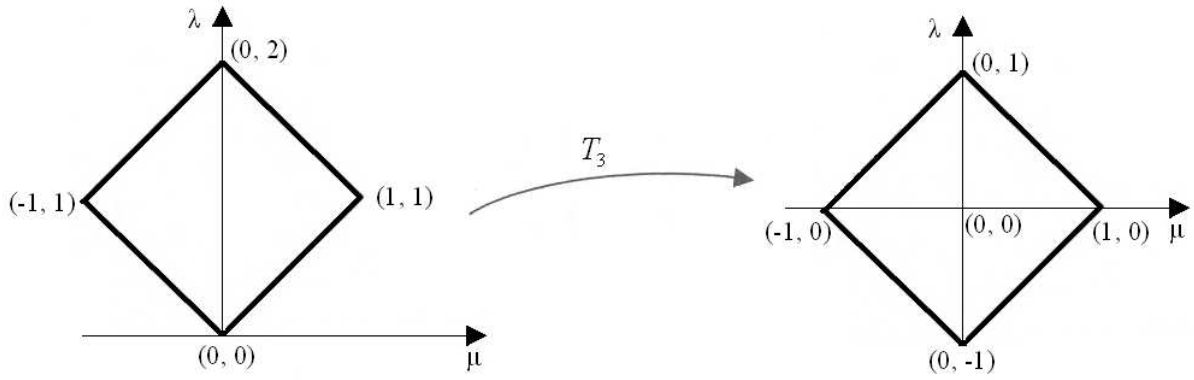


Figura 2.6 - Translação

Através da composição $T_3 \circ T_2 \circ T_1$ obtem-se a seguinte transformação linear:

$$T(\mu,\lambda)=(\mu-\lambda,\mu+\lambda-1)$$

Como $G_C=\mu-\lambda$ e $G_{ct}=\mu+\lambda-1$.

Pode-se dizer que $T(\mu,\lambda)=(G_C,G_{ct})$.

As transformações F_1, F_2 e F_3 são as transformadas inversas de T_1, T_2 e T_3 respectivamente.

A transformação linear F_3 proporciona uma translação.

$$F_3(G_C, G_{ct}) = (G_C, G_{ct} + 1) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{G_{ct}+1}{G_{ct}} \end{bmatrix} (G_C, G_{ct})$$

A figura 2.7 mostra graficamente a translação proporcionada por F_3.

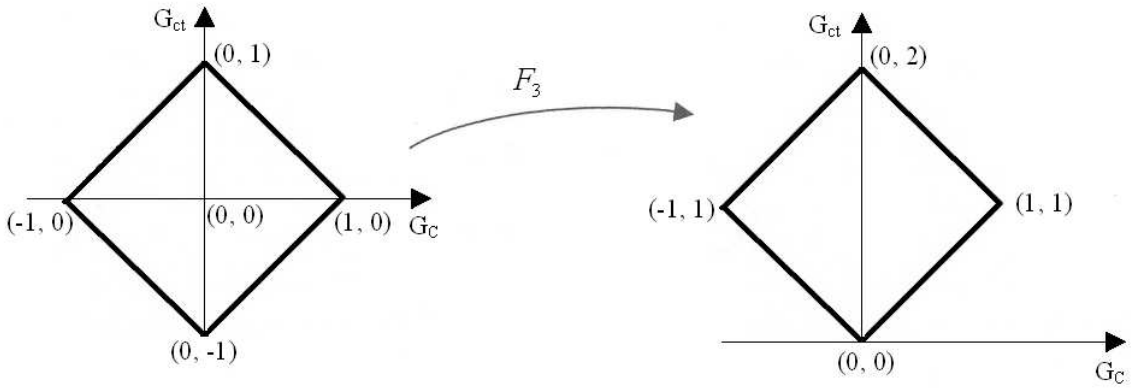


Figura 2.7 – Translação proporcionada por F_3

A transformação linear F_2 proporciona uma rotação de 45°.

$$F_2(G_C, G_{ct}) = \left(\frac{\sqrt{2}}{2} G_C + \frac{\sqrt{2}}{2} G_{ct}, -\frac{\sqrt{2}}{2} G_C + \frac{\sqrt{2}}{2} G_{ct} \right) = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} (G_C, G_{ct})$$

A figura 2.8 mostra graficamente a rotação proporcionada por F_2.

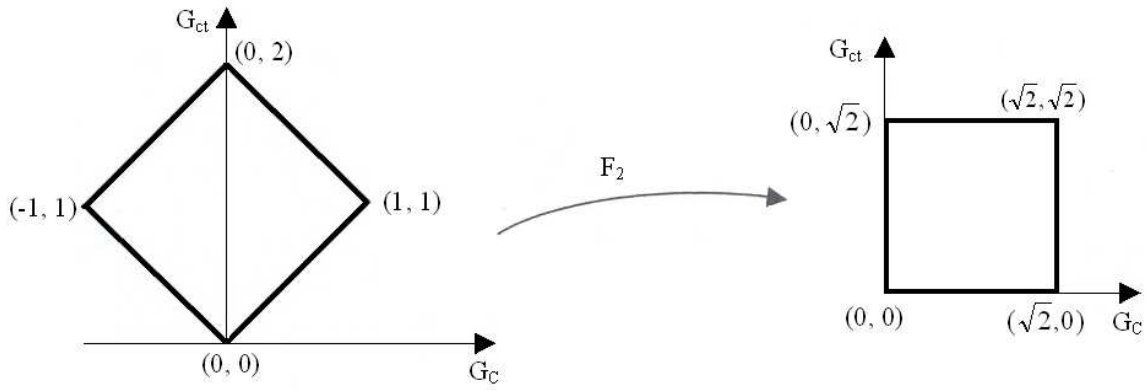


Figura 2.8 - Rotação proporcionada por F_2

A transformação linear F_1 proporciona uma mudança de escala.

$$F_1(G_C, G_{ct}) = \left(\frac{\sqrt{2}}{2} G_C, \frac{\sqrt{2}}{2} G_{ct} \right) = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} (G_C, G_{ct})$$

A figura 2.9 mostra graficamente a mudança de escala proporcionada por F_1.

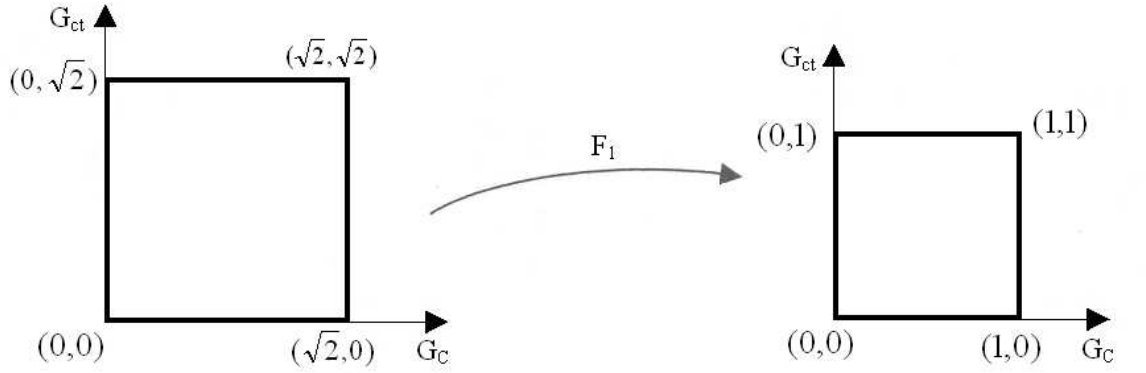


Figura 2.9 - Mudança de escala proporcionada por F_1

Se for calculado a composição $F_1 \circ F_2 \circ F_3$ obtém-se a transformação linear:

$$F(\mu, \lambda) = \frac{G_C}{2} + \frac{G_{ct}}{2} + \frac{1}{2}, -\frac{G_C}{2} + \frac{G_{ct}}{2} + \frac{1}{2} \tag{2.3}$$

A figura 2.10 mostra graficamente as transformações lineares T e F.

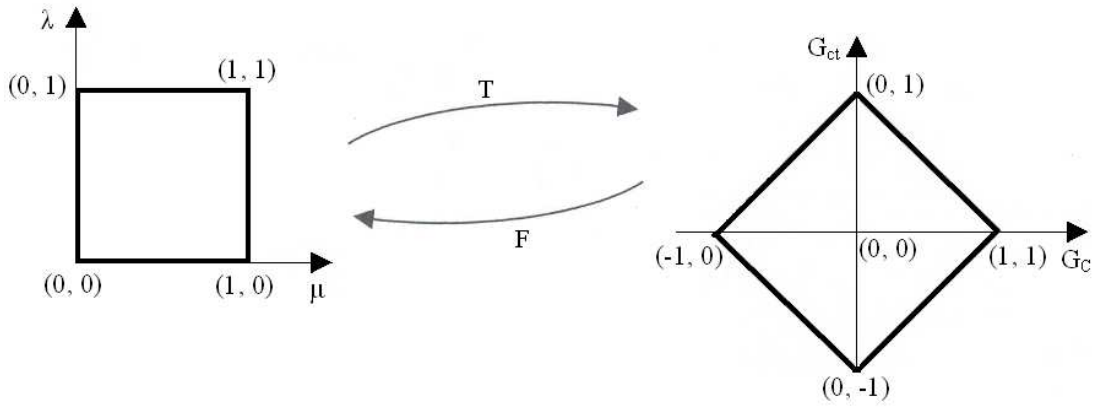


Figura 2.10 – Representação gráfica das transformações T e F

2.3 Grau de Certeza Resultante

Grau de Certeza baixo significa que as fontes de informação não têm evidências suficientes para sustentar uma afirmação ou uma refutação a respeito de uma proposição. Essa situação pode acontecer por dois motivos:

1 – As fontes estão trazendo para a análise uma baixa intensidade de evidências para afirmar ou negar uma proposição. Ou seja, μ e λ possuem valores próximos de 0,5.

2 – As fontes estão trazendo para a análise evidências com alta intensidade de inconsistência. Nessa situação o Grau de Certeza só cresce quando o Grau de Contradição cai.

Se o baixo valor do Grau de Certeza tem como origem a insuficiência de informação, podem-se reforçar os graus de evidência sem variar o Grau de Contradição com o objetivo de se aumentar o valor do Grau de Certeza para se ter uma melhor análise. Consegue-se isso através da metodologia apresentada em (DA SILVA FILHO, ABE & LAMBERT-TORRES, 2008) descrita a seguir.

Se for conseguido detectar se o baixo Grau de Certeza tem como origem a insuficiência de informação, e se sabe que se forem reforçados os graus de evidências, pode-se ter uma melhor análise.

Chama-se de Intervalo de Certeza (φ) o intervalo de valores do Grau de Certeza (G_C) que podem variar sem alterar o valor do Grau de Contradição (G_{ct}). Determina-se esse valor da seguinte forma:

$$\varphi = 1 - |G_{ct}| \tag{2.4}$$

O Grau de Veracidade Máximo ($G_{ve_{\max}}$) é o próprio valor máximo positivo do Intervalo de Certeza, ou seja, $G_{ve_{\max}} = +\varphi$.

O Grau de Falsidade Máximo ($G_{fa_{\max}}$) é o valor máximo negativo do Intervalo de Certeza, portanto, $G_{fa_{\max}} = -\varphi$.

A figura 2.11 mostra a idéia acima exposta.

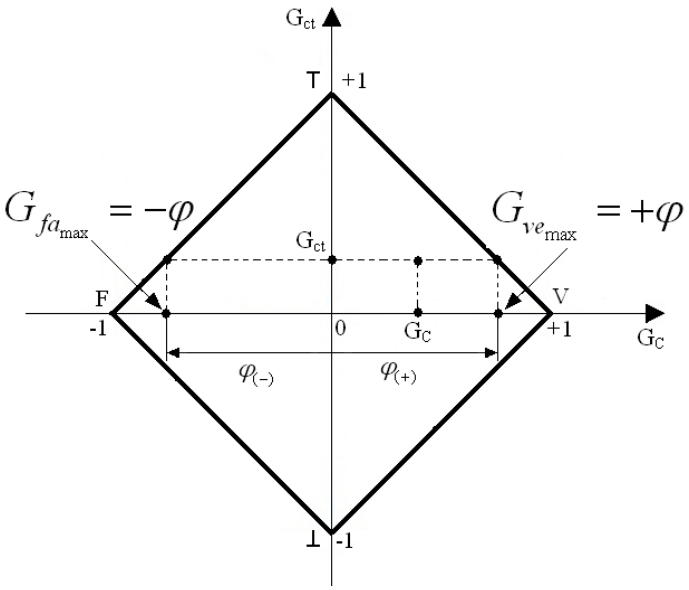


Figura 2.11 – Representação no Reticulado dos máximos valores de Graus de Certeza com Grau de Contradição constante

Chama-se de Intervalo de Certeza Sinalizado $(\varphi_{(\pm)})$ o valor do Intervalo de Certeza considerando a sinalização do Grau de Contradição (G_{ct}). Caso o Grau de Contradição (G_{ct}). seja positivo temos $\varphi_{(+)}$. Se o Grau de Contradição for negativo tem-se $\varphi_{(-)}$.

Assim, pode-se definir o Grau de Certeza Resultante (G_{C_r}) como o Grau de Certeza (G_C) acompanhado pelo valor do Intervalo de Certeza Sinalizado $(\varphi_{(\pm)})$. Representado da seguinte forma:

$$G_{C_r} = \begin{bmatrix} G_C \\ \varphi_{(\pm)} \end{bmatrix}$$

Com o valor do Grau de Certeza Resultante (G_{C_r}) se pode determinar quais são as evidências, se favoráveis ou contrária, devem ser modificadas para se alcançar o valor do Grau de Certeza desejado. Também, a partir do valor de G_{C_r}, pode-se estimar quanto se deve

aumentar ou diminuir as evidências para se obter o máximo valor possível do Grau de Certeza.

2.4 Grau de Certeza Estimado

Chama-se de Grau de Certeza Estimado (G_{Cest}) o valor do Grau de Certeza alcançado quando as evidências de entrada são alteradas de tal forma a diminuir o Grau de Contradição a zero. Descreve-se em seguida como se determina o valor do Grau de Evidência Contrária (λ) para se conseguir o Grau de Certeza Estimado (G_{Cest}).

A figura 2.12 mostra um reticulado com os valores do Grau de Certeza e Grau de Contradição.

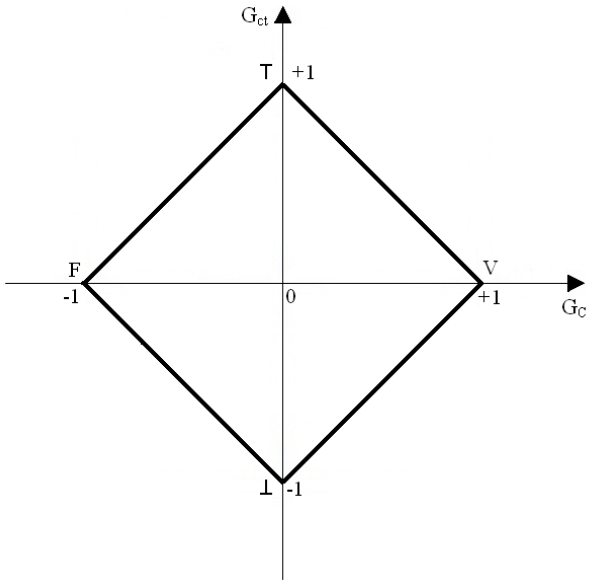


Figura 2.12 – Reticulado representando os Graus de Certeza e Contradição

Traça-se um segmento de reta paralelo ao eixo do Grau de Certeza e que passa pelo Grau de Contradição como mostrado na figura 2.13.

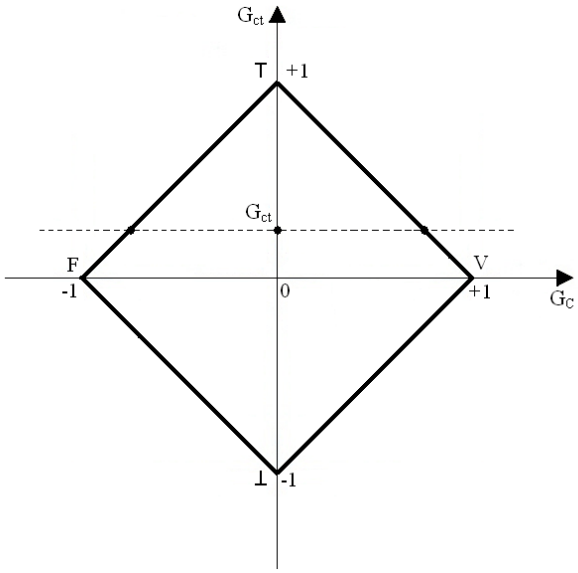


Figura 2.13 – Reticulado destacando o Grau de Contradição

Depois, traça-se um segmento de reta paralelo ao eixo do Grau de Contradição passando pelo Grau de Certeza. Vê-se isso na figura 2.14.

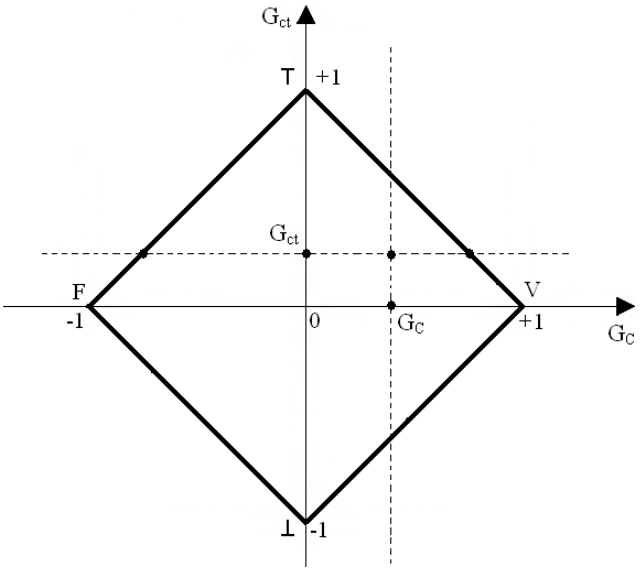


Figura 2.14 – Reticulado destacando os Graus de Certeza e Contradição

A figura 2.15 mostra o segmento de reta r que liga o ponto A ao ponto V.

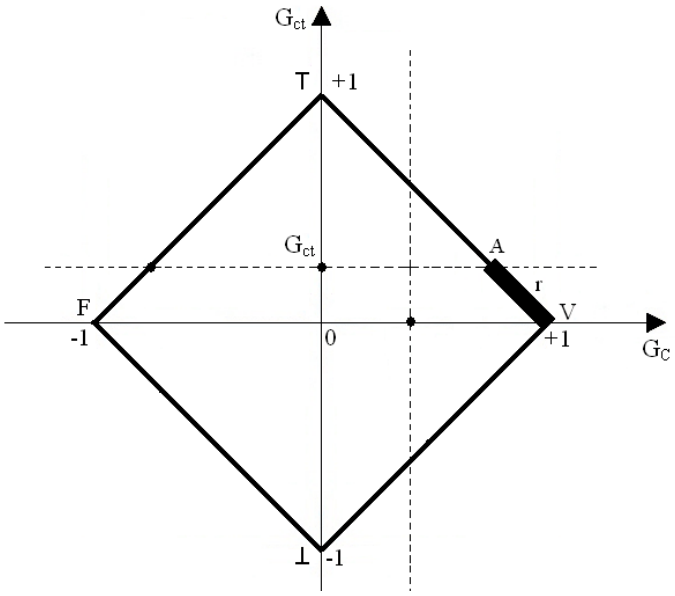


Figura 2.15 – Reticulado destacando o segmento de reta r

O comprimento do segmento de reta r pode ser calculado da seguinte forma:

$$r^2 = (1-\varphi)^2 + |G_{ct}|^2 \tag{2.5}$$

$$r = \sqrt{(1-\varphi)^2 + |G_{ct}|^2} \tag{2.6}$$

Se for transladado o segmento de reta r para a esquerda até se encontrar o ponto B, como se mostra na figura 2.16, encontraremos o Grau de Certeza Estimado (G_{Cest}), como se pode ver na figura 2.17.

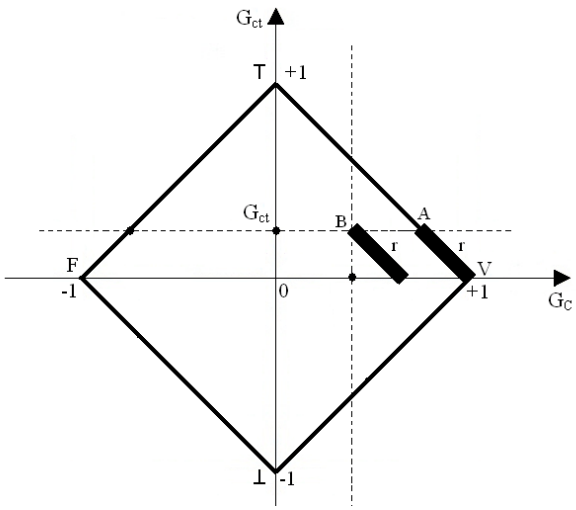


Figura 2.16 – Reticulado destacando a translação do segmento de reta r

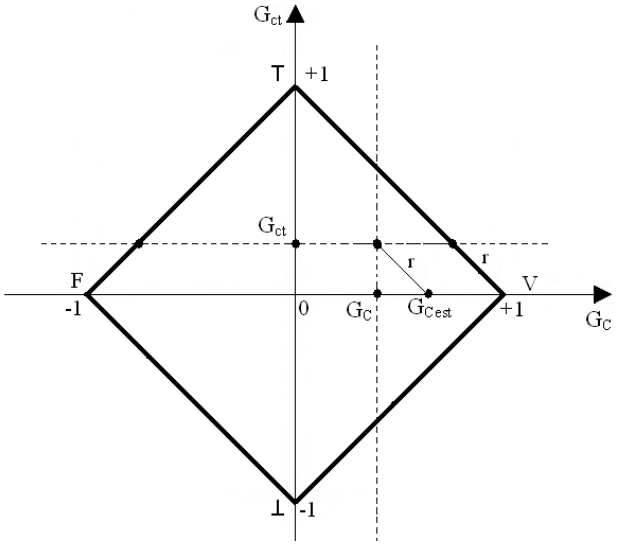


Figura 2.17 – Reticulado destacando o Grau de Certeza Estimado

Descreve-se a seguir como determinar o acréscimo no valor do Grau de Certeza (G_C) para se obter o Grau de Certeza Estimado (G_{Cest}).

2.5 Mudanças no Grau de Certeza para se obter o Grau de Certeza Estimado

Chama-se, inicialmente, de x o segmento de reta que conecta o Grau de Certeza (G_C) ao Grau de Certeza Estimado, (G_{Cest}) como pode ser visto na figura 2.18.

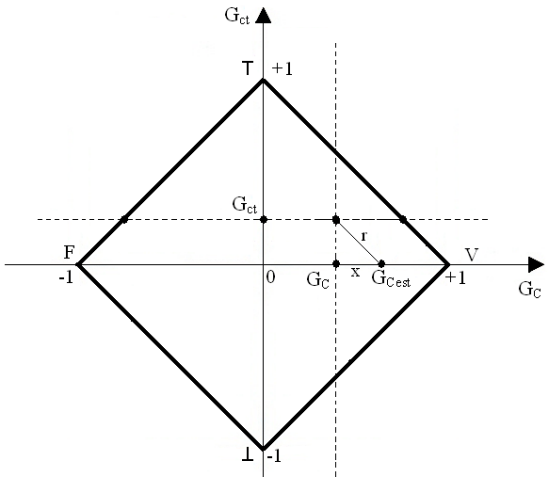


Figura 2.18 – Reticulado destacando o segmento de reta x

Determina-se o comprimento do segmento de reta x da seguinte forma:

$$x^2 = r^2 - |G_{ct}|^2$$

Assim, substituindo r^2 pela equação (2.5), temos:

$$x^2 = (1-\varphi)^2 + |G_{ct}|^2 - |G_{ct}|^2$$

$$x^2 = (1-\varphi)^2$$

$$x = 1-\varphi$$

Portanto, o valor do Grau de Certeza Estimado (G_{Cest}) é:

$$G_{Cest} = G_C + (1-\varphi) \qquad (2.7)$$

Como descrito anteriormente o reticulado é encontrado a partir do Quadrado Unitário. Assim, pode-se, no reticulado representar o eixo do Grau de Evidência Favorável e Grau de Evidência Desfavorável. Mostra-se na figura 2.19 o reticulado com os eixos dos Graus de Evidência.

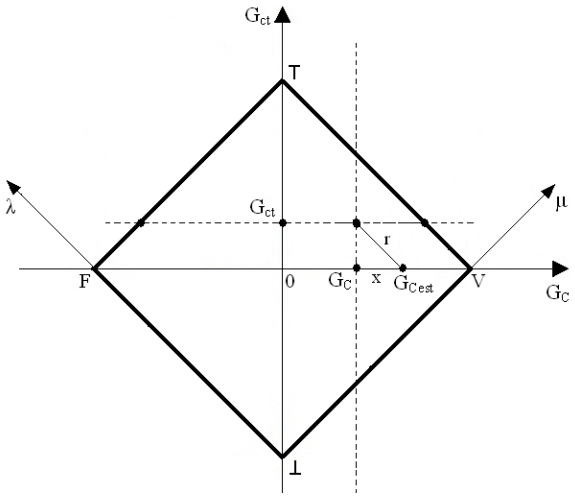


Figura 2.19 – Reticulado com o eixo dos Graus de Evidência

A partir do prolongamento do segmento de reta r em direção ao eixo do Grau de Evidência Favorável (μ) até encontrá-lo, pode-se determinar o Grau de Evidência Favorável Inicial (μ_i). Como pode ser visto na figura 2.20.

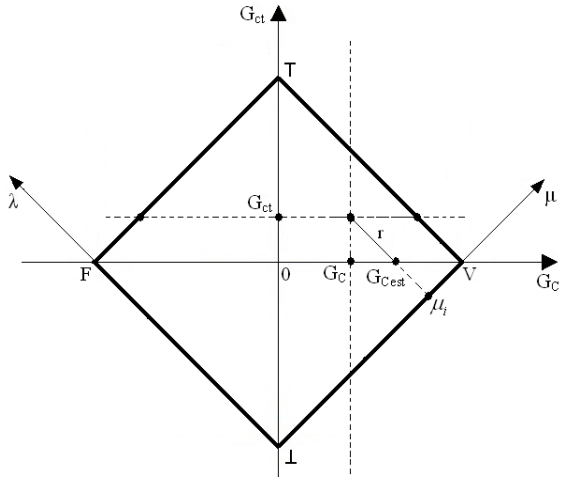


Figura 2.20 - Reticulado destacando o Grau de Evidência Favorável Inicial (μ_i)

Ao se traçar um segmento de reta paralelo ao Eixo do Grau de Evidência Favorável (μ) a partir do ponto B em direção ao Eixo do Grau de Evidência Contrária (λ), quando esse segmento de reta encontrar o Eixo do Grau de Evidência Contrária (λ), se encontrará o Grau de Evidência Contrária Inicial (λ_i). A figura 2.21 mostra como obter o Grau de Evidência Desfavorável Inicial.

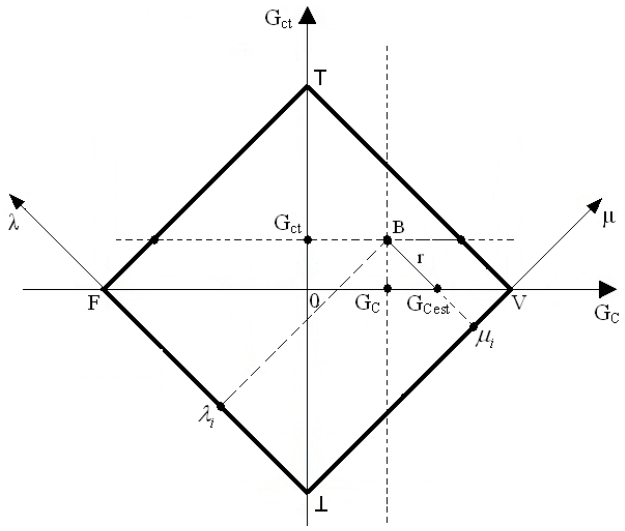


Figura 2.21 - Reticulado destacando o Grau de Evidência Contrária Inicial (λ_i)

Quando se traça um segmento de reta paralelo ao Eixo do Grau de Evidência Favorável (μ) a partir do Grau de Certeza Estimado (G_{Cest}) em direção ao Eixo do Grau de Evidência Contrária (λ), quando esse segmento de reta encontrar o Eixo do Grau de Evidência Contrária (λ), se encontrará o Grau de Evidência Contrária Final (λ_f). A figura 2.22 mostra como obter o Grau de Evidência Contrária Final.

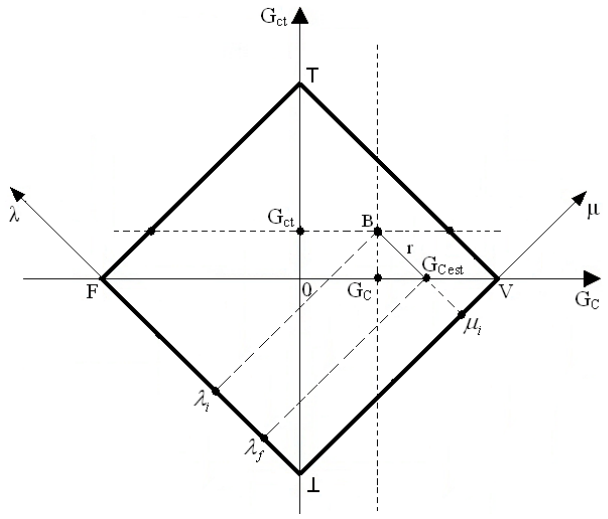


Figura 2.22 - Reticulado destacando o Grau de Evidência Contrária Final (λ_f)

Para se conseguir o Grau de Certeza Estimado, o Grau de Evidência Contrária deve variar de λ_i para λ_f. Assim, tem-se:

$$\Delta\lambda = \lambda_i - \lambda_f$$

Observa-se que $\Delta\lambda$ = r. Portanto:

$$r = \lambda_i - \lambda_f$$

$$\lambda_f = \lambda_i - r \qquad (2.8)$$

A equação (2.4) mostra que o segmento de reta r pode ser determinado da seguinte forma: $r = \sqrt{(1-\varphi)^2 + G_{ct}{}^2}$.

Os pontos B, G_{Cest} e G_C formam um triângulo retângulo com os catetos iguais como se pode ver na figura 2.23.

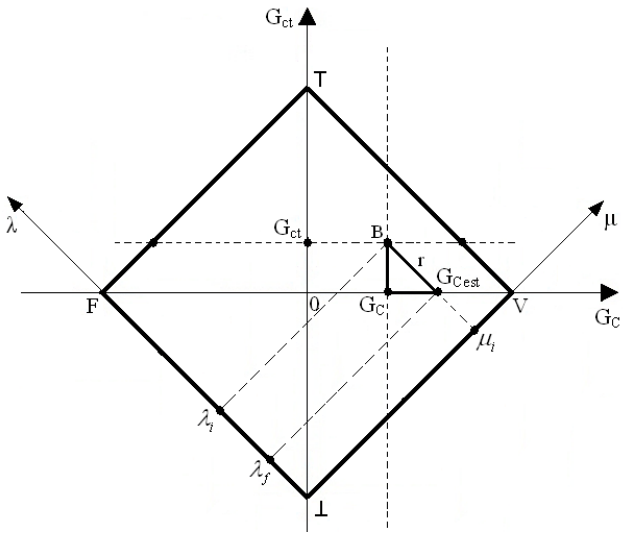


Figura 2.23 - Reticulado destacando o triângulo formado pelos pontos B, G_{Cest} e G_C

Assim, pode-se reescrever a equação (2.6) da seguinte forma:

$$r = \sqrt{G_{ct}{}^2 + G_{ct}{}^2}$$

Consequentemente:

$$r = G_{ct}\sqrt{2} \qquad (2.9)$$

Quando da passagem do Quadrado Unitário do Plano Cartesiano – QUPC para o reticulado houve um aumento de escala de $\sqrt{2}$ nos valores dos Graus de Evidência. Portanto, deve se dividir a equação (2.9) por $\sqrt{2}$ antes de substituí-la na equação (2.8) para se encontrar o valor correto do Grau de Evidência Contrária Final λ_f. Assim, temos:

$$\lambda_f = \lambda_i - G_{ct} \qquad (2.10)$$

Determina-se o Grau o de Certeza G_C subtraindo-se o Grau de Evidência Favorável (μ) pelo Grau de Evidência Contrária (λ):

$$G_C = \mu_i - \lambda_i$$

Como o Grau de Certeza Estimado (G_{Cest}) nada mais é do que um novo valor do Grau de Certeza determinado após a variação do Grau de Evidência Desfavorável de λ_i para λ_f. Pode-se, também, encontrar o Grau de Evidência Contrária Final através das equações abaixo:

$$G_{Cest} = \mu_i - \lambda_f$$

$$\lambda_f = G_{Cest} - \mu_i$$

A figura 2.24 mostra a situação quando o Grau de Certeza (G_C) é positivo e o Grau de Contradição (G_{ct}) é negativo.

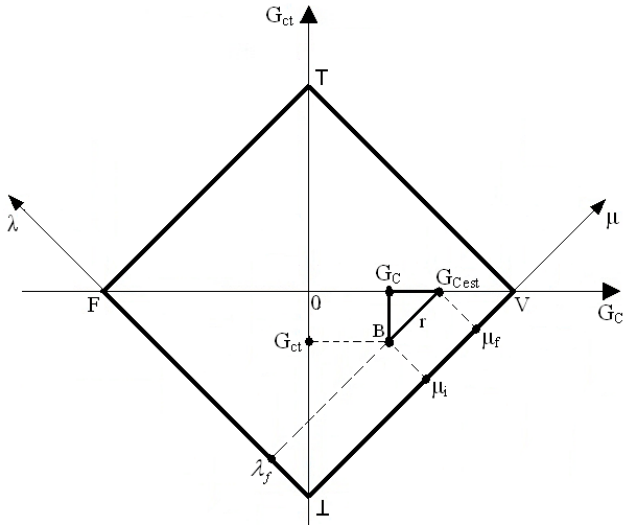


Figura 2.24 – Reticulado com Grau de Certeza positivo e Grau de Contradição negativo

Nessa situação determina-se o Grau de Certeza Estimado G_{Cest} também através da equação (2.5): $G_{Cest} = G_C + (1 - \varphi)$.

Para o Grau de Incerteza ser nulo o Grau de Evidência Favorável μ deve ser variado, portanto:

$$\Delta\mu = \mu_i - \mu_f$$

Sendo μ_f é o Grau de Evidência Favorável Final.

Verifica-se que $\mu_f = \mu_i + r$.

Assim, da mesma forma como se encontrou a equação (2.10), pode-se afirmar que: $\mu_f = \mu_i + |G_{ct}|$. Que também pode ser determinado da seguinte forma:

$$G_{Cest} = \mu_f - \lambda_i$$

$$\mu_f = \lambda_i + G_{Cest}$$

A figura 2.25 mostra a situação quando o Grau de Certeza (G_C) é negativo e o Grau de Contradição (G_{ct}) é positivo.

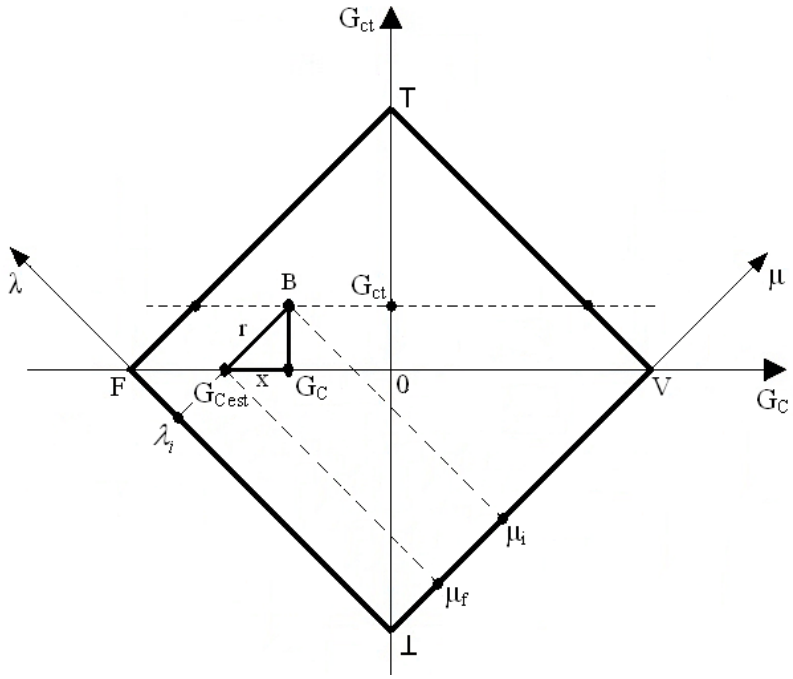


Figura 2.25 – Reticulado com Grau de Certeza negativo e Grau de Contradição positivo

Para se determinar o valor do Grau de Certeza Estimado ($G_{C\,est}$) deve somar o valor do segmento de reta x ao valor do Grau de Certeza (G_C).

Determina-se o valor do segmento de reta x da seguinte forma:

$$x = -(\sqrt{r^2 - G_{ct}{}^2})$$

Ou:

$$x = \varphi - 1$$

Assim, encontra-se o Grau de Certeza Estimado (G_{Cest}) da seguinte forma:

$$G_{Cest} = G_C + x$$

$$G_{Cest} = G_C + (\varphi - 1) \qquad (2.11)$$

Para o Grau de Contradição (G_{ct}) ser nulo o Grau de Evidência Favorável (μ) deve ser variado, portanto:

$$\Delta\mu = \mu_i - \mu_f$$

Verifica-se que $\mu_f = \mu_i - r$.

Assim, da mesma forma como se encontrou a equação (2.10), pode-se afirmar que: $\mu_f = \mu_i - |G_{Cest}|$. Que também pode ser determinado da seguinte forma:

$$G_{Cest} = \mu_f - \lambda_i$$

$$\mu_f = \lambda_i + G_{Cest}$$

A figura 2.26 mostra a situação quando o Grau de Certeza (G_C) é negativo e o Grau de Contradição (G_{ct}) também é negativo.

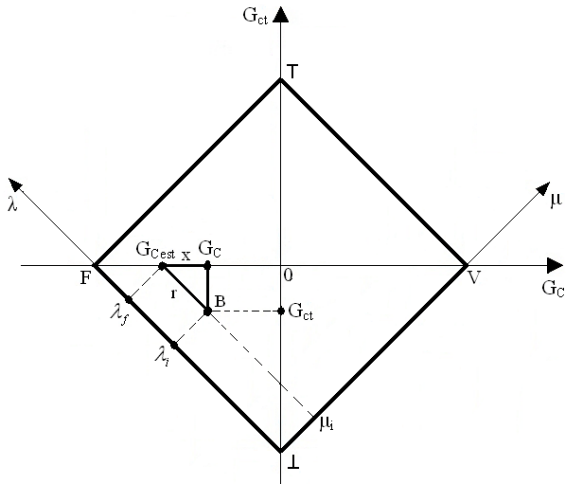


Figura 2.26 – Reticulado com Grau de Certeza negativo e Grau de Contradição negativo

Para se determinar o valor do Grau de Certeza Estimado (G_{Cest}) deve-se somar o valor do segmento de reta x ao valor do Grau de Certeza (G_C).

Determina-se o valor do segmento de reta x da seguinte forma:

$$x = -(\sqrt{r^2 - G_{ct}{}^2})$$

Ou:

$$x = \varphi - 1$$

Assim, encontra-se o Grau de Certeza Estimado (G_{Cest}) da seguinte forma:

$$G_{Cest} = G_C + x$$

$$G_{Cest} = G_C + (\varphi - 1) \qquad (2.12)$$

Para o Grau de Contradição (G_{ct}) ser nulo o Grau de Evidência Contrária (λ) deve ser variado, portanto:

$$\Delta\lambda = \lambda_i - \lambda_f$$

Verifica-se que $\lambda_f = \lambda_i + r$.

Assim, da mesma forma como se encontrou a equação (2.8), pode-se afirmar que: $\lambda_f = \lambda_i + G_{ct}$. Que também pode ser determinado da seguinte forma:

$$G_{Cest} = \mu_i - \lambda_f$$

$$\lambda_f = \mu_i - G_{Cest}$$

Resumidamente, para se saber qual valor de evidencia, favorável ou contrária, se deve variar para alcançar o Grau de Certeza Estimado (G_{Cest}), procede-se da seguinte forma:

Para $\varphi = +\varphi$

Se $G_{ct} > 0$ Diminui-se λ e mantém-se μ.

Senão Aumenta-se μ e mantém-se λ.

Para $\varphi = -\varphi$

Se $G_{ct} > 0$ Diminui-se μ e mantém-se λ.

Senão Aumenta-se λ e mantém-se μ.

2.6 Grau de Certeza de Valor Real

Descreve-se em seguida uma maneira de se determinar o valor do Grau de Certeza desconsiderando-se os efeitos das contradições. Esse valor será denominado como Grau de Certeza Real (G_{CR}).

Observa-se na figura 2.27 que o segmento de reta D pode ser determinado da seguinte forma:

$$D = \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2} \tag{2.13}$$

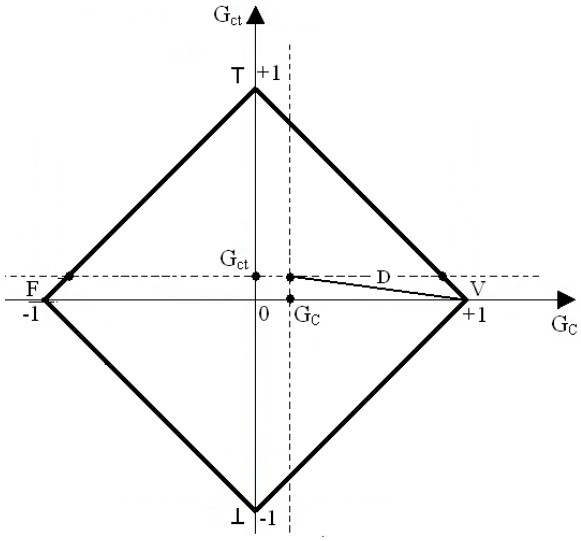


Figura 2.27 – Primeiro exemplo do segmento de reta D

As figuras 2.28 e 2.29, respectivamente, mostram o segmento de reta D gerado a partir de diferentes valores de Grau de Contradição.

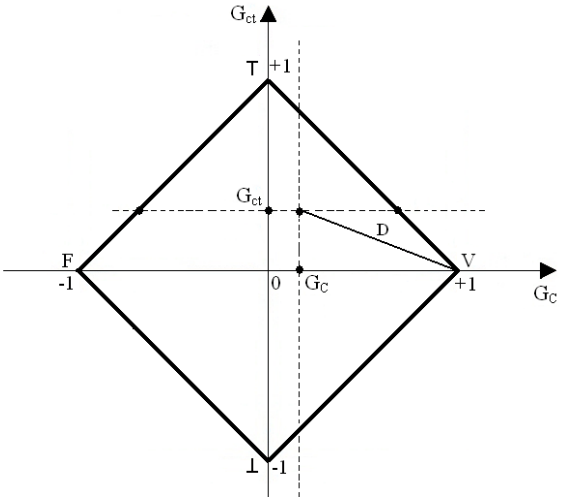


Figura 2.28 – Segundo exemplo do segmento de reta D

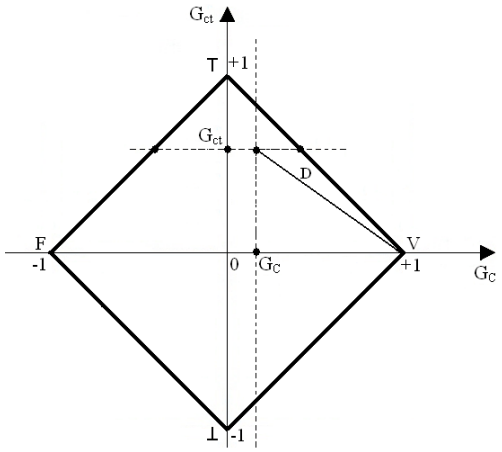


Figura 2.29 – Terceiro exemplo do segmento de reta D

A projeção do segmento de reta D no eixo do Grau de Certeza (G_C) determina o Grau de Certeza Real (G_{CR}). Assim, quanto maior o valor do Grau de Contradição (G_{ct}) menor o valor do Grau de Certeza Real (G_{CR}). As figuras 2.30, 2.31 e 2.32 demonstram isso.

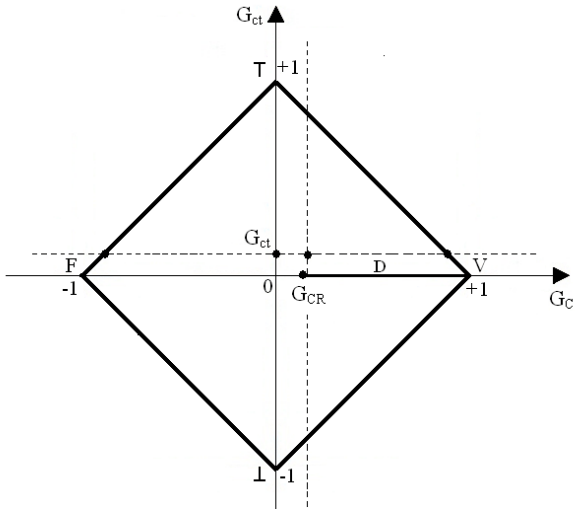


Figura 2.30 – Grau de Certeza Real determinado a partir do primeiro exemplo de segmento de reta D

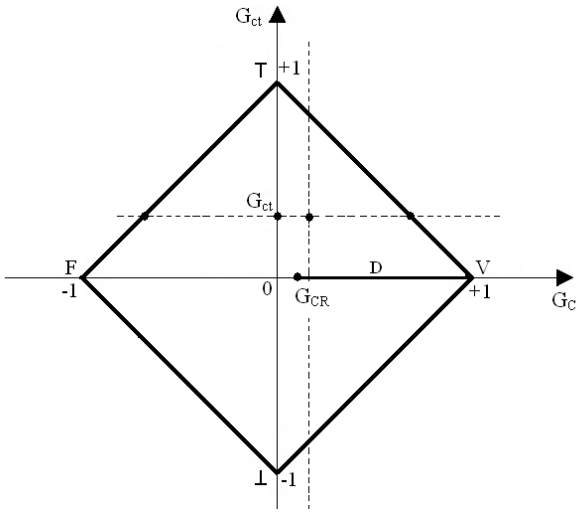


Figura 2.31 – Grau de Certeza Real determinado a partir do segundo exemplo de segmento de reta D

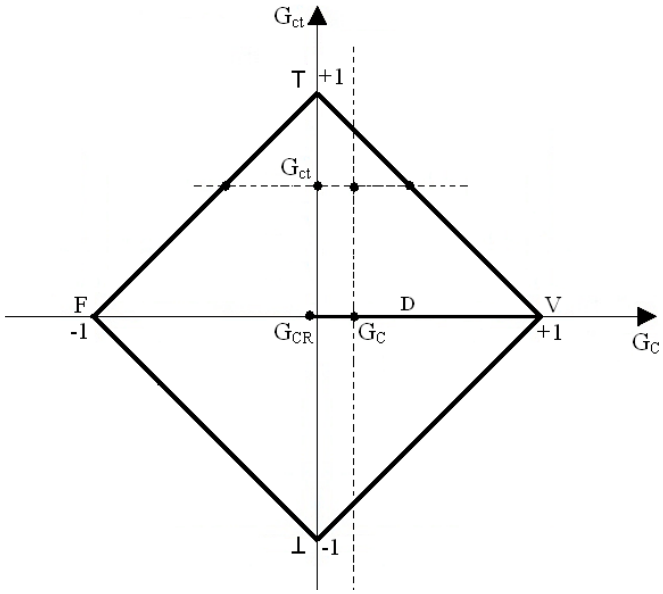


Figura 2.32 – Grau de Certeza Real determinado a partir do terceiro exemplo de segmento de reta D

Portanto, quando G_C é positivo, determina-se G_{CR} da seguinte forma:

$$G_{CR} = 1 - D \tag{2.14}$$

$$G_{CR} = 1 - \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2} \tag{2.15}$$

Por sua vez, quando G_C é negativo, determina-se G_{CR} da seguinte forma:

$$G_{CR} = D - 1 \tag{2.16}$$

$$G_{CR} = \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2} - 1 \tag{2.17}$$

Assim, pode-se definir o Grau de Certeza Resultante Real (G_{Crr}) como o Grau de Certeza Real (G_{CR}) acompanhado pelo valor do Intervalo de Certeza Sinalizado $(\varphi_{(\pm)})$. Representado da seguinte forma:

$$G_{Crr} = \begin{bmatrix} G_{CR} \\ \varphi_{(\pm)} \end{bmatrix} \tag{2.18}$$

A partir do valor do Grau de Certeza Resultante Real (G_{Crr}) pode-se recuperar o Grau de Contradição (G_{ct}), o Grau de Certeza (G_C) e qual Grau de Evidencia deve ser modificado para se diminuir a contradição.

Pode-se encontrar o Grau de Contradição (G_{ct}) a partir da equação (2.3).

$$\varphi = 1 - |G_{ct}|$$

Considerando o Intervalo de Certeza Sinalizado, temos:

$$|G_{ct}| = 1 - \varphi_{(\pm)}$$

Se o sinal de $\varphi_{(\pm)}$ for positivo, o Grau de Certeza é positivo. Caso o sinal de $\varphi_{(\pm)}$ seja negativo, o Grau de Contradição é negativo.

Considerando o Grau de Certeza Real (G_{CR}) como sendo positivo, a partir da equação (2.14) pode-se determinar a distância D.

$$G_{CR} = 1 - D$$

$$D = 1 - G_{CR} \qquad (2.19)$$

Igualando a equação (2.19) com a equação (2.13), pode-se determinar o valor do Grau de Certeza.

$$1 - G_{CR} = \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2}$$

$$G_C = 1 - \sqrt{(1 - G_{CR})^2 - G_{ct}{}^2} \qquad \text{Se } G_{CR} > 0 \qquad (2.20)$$

Caso o Grau de Certeza Real (G_{CR}) seja negativo, a partir da equação (2.16) pode-se determinar a distância D.

$$G_{CR} = D - 1$$

$$D = G_{CR} + 1 \qquad (2.21)$$

Igualando a equação (2.21) com a equação (2.13), pode-se determinar o valor do Grau de Certeza.

$$G_{CR}+1=\sqrt{(1-|G_C|)^2+{G_{ct}}^2}$$

$$G_C=\sqrt{(G_{CR}+1)^2-{G_{ct}}^2}-1 \qquad \text{Se } G_{CR}<0 \tag{2.22}$$

Assim, podem-se encontrar os valores das evidências que geraram o G_{CR} através da equação (2.3) mostrada abaixo.

$$F(\mu,\lambda)=\frac{G_C}{2}+\frac{G_{ct}}{2}+\frac{1}{2},-\frac{G_C}{2}+\frac{G_{ct}}{2}+\frac{1}{2}$$

Portanto, o Grau de Evidência Favorável (μ) e o Grau de Evidência Contrária (λ) podem ser recuperados como mostrado a seguir.

$$\mu=\frac{G_C+G_{ct}+1}{2} \tag{2.23}$$

$$\lambda=\frac{-G_C+G_{ct}+1}{2} \tag{2.24}$$

As ideias apresentadas neste capítulo são básicas para o entendimento das redes neurais paraconsistentes apresentadas no próximo capítulo.

CAPÍTULO 3 - REDES NEURAIS ARTIFICIAIS PARACONSISTENTES

O sistema de navegação e controle proposto neste trabalho utiliza as Redes Neurais Artificiais Paraconsistentes como base. Assim, neste capítulo, descreve-se o funcionamento dessas redes. O entendimento dos conceitos apresentados no capítulo anterior, onde se apresentou a Lógica Paraconsistente Anotada Evidencial Eτ, é fundamental para o entendimento do funcionamento das Redes Neurais Artificiais Paraconsistentes.

3.1 Sistema ou Nó de Análise Paraconsistente (NAP)

As equações apresentadas no capítulo anterior podem ser denominadas de Sistema ou Nó de Análise Paraconsistente (NAP). Esses nós podem ser interligados formando uma Rede Neural Artificial – RNA capaz de tratar informações incertas ou contraditórias.

Os Graus de Evidências retirados da base de dados de conhecimento incerto são as fontes de informação da rede. Assim, cada NAP gera como saída um valor de Grau de Certeza Resultante Real (G_{Crr}).

Pode-se ver na figura 3.1 a representação em blocos de um NAP.

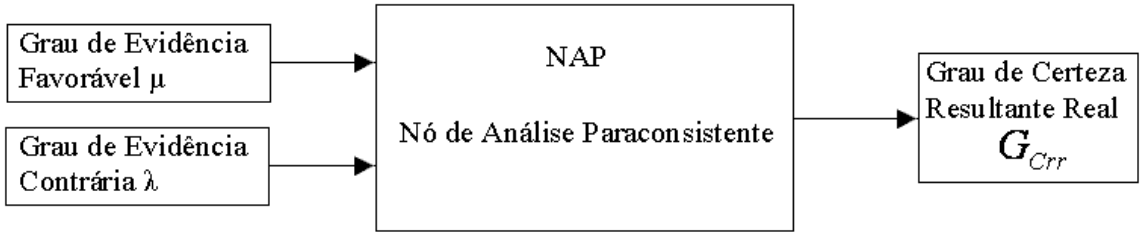


Figura 3.1 - Representação em blocos de um NAP

A figura 3.2 mostra a representação simbólica de um NAP.

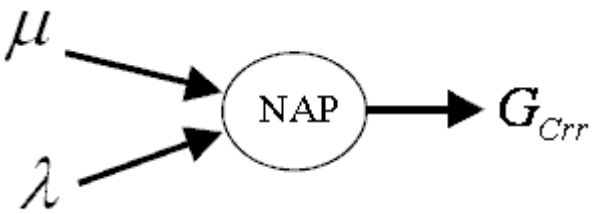


Figura 3.2 - Representação simbólica de um NAP

Em (DA SILVA, ABE & LAMBERT-TORRES) apresenta-se as seguintes normas a respeito dos NAPs:

1 – O Grau de Evidência Favorável (μ), o Grau de Evidência Contrária (λ), o Grau de Certeza (G_C), o Grau de Certeza Real (G_{CR}) ou o Grau de Contradição (G_{ct}) não devem, indiscriminadamentes, ser somados, subtraídos ou considerado as médias.

2 – Não se deve somar, subtrair ou considerar as médias dos Intervalos de Certeza.

3 – O valor do Grau de Certeza Resultante somente poderá ser reforçado ou enfraquecido através da injeção de novas evidências na entrada do sistema.

4 – O reforço ou enfraquecimento do Grau de Certeza resultante, por meio de Evidências complementares somente deve ser efetuado até o limite estabelecido pelo Intervalo de Certeza. Após este valor as Evidências devem ser ajustadas com novos valores para que se diminuam as contradições.

5 – Os valores dos Graus de Evidências poderão ser ajustados simultaneamente para se modificar o valor do Grau de Certeza Resultante.

Cada NAP deve ter como saída um valor de evidência, ou seja, um valor entre 0 e 1, já que este sinal pode ser a entrada de outro NAP. Assim, como as equações apresentadas no capítulo anterior têm como saída o Grau de Certeza Real (G_{Crr}), um valor entre –1 e 1, esse valor deve ser normalizado formando o Grau de Evidência Resultante (μ_E). Faz-se a normalização da seguinte forma.

$$\mu_E = \frac{G_C + 1}{2} \qquad (3.1)$$

$$\mu_E = \frac{(\mu - \lambda) + 1}{2} \qquad (3.2)$$

Chama-se de Grau de Evidência Resultante Real (μ_{Er}) o Grau de Evidência Resultante livre dos efeitos da contradição. Determina-se esse valor considerando-se, ao invés do Grau de Certeza (G_C), o Grau de Certeza Real (G_{CR}). As equações (2.13) e (2.15), repetidas abaixo, mostram como se determina o Grau de Certeza Real quando G_C for positivo e negativo, respectivamente.

$$G_{CR} = 1 - \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2}$$

$$G_{CR} = \sqrt{(1 - |G_C|)^2 + G_{ct}{}^2} - 1$$

Portanto, determina-se o μ_{Er} da seguinte forma:

$$\mu_{Er} = \frac{G_{CR} + 1}{2} \qquad (3.3)$$

O NAP também gera como saída um valor correspondente Intervalo de Evidência Resultante (φ_E). Esse valor corresponde ao Intervalo de Certeza Sinalizado ($\varphi_{(\pm)}$) normalizado.

Encontra-se o Intervalo de Certeza através da equação (2.2) repetida abaixo.

$\varphi = 1 - |G_{ct}|$

Assim, para se encontrar o valor do Intervalo de Evidência Resultante (φ_E), deve-se, inicialmente, encontrar o valor do Grau de Contradição Normalizado (μ_{ctr}) da seguinte maneira.

$$\mu_{ctr} = \frac{G_{ct} + 1}{2} \qquad (3.4)$$

Como $G_{ct} = (\mu + \lambda) - 1$, pode-se afirmar que:

$$\mu_{ctr} = \frac{\mu + \lambda}{2} \qquad (3.5)$$

Também se pode dizer que:

$$G_{ct} = 2.\mu_{ctr} - 1 \qquad (3.6)$$

Substituindo a equação (3.6) em (2.2), determina-se o Intervalo de Evidência Resultante (φ_E).

$$\varphi_E = 1 - |2.\mu_{ctr} - 1| \qquad (3.7)$$

Define-se o Intervalo de Evidência Resultante Sinalizado $\varphi_{E(\pm)}$ como sendo o valor de φ_E considerando a sinalização do Grau de Contradição (G_{ct}).

Assim, quando $\mu_{ctr} > 0{,}5$, situação ocorrida quando $G_{ct} > 0$, têm-se $\varphi_{E(\pm)} = \varphi_{E(+)}$.

Da mesma forma, quando $\mu_{ctr} < 0{,}5$, situação ocorrida quando $G_{ct} < 0$, têm-se $\varphi_{E(\pm)} = \varphi_{E(-)}$.

Portanto, o valor do Grau de Contradição Normalizado pode ser encontrado a partir do Intervalo de Evidência Resultante Sinalizado através das seguintes equações:

$$\text{Se } \varphi_{E(\pm)} = \varphi_{E(+)},\ \mu_{ctr} = \frac{1 + (1 - \varphi_E)}{2} \qquad (3.8)$$

$$\text{Se } \varphi_{E(\pm)} = \varphi_{E(-)},\ \mu_{ctr} = \frac{1 - (1 - \varphi_E)}{2} \qquad (3.9)$$

Define-se o Grau de Evidência de Saída (μ_{Es}) como sendo o Grau de Evidência Resultante (μ_E) acompanhado do valor do Intervalo de Evidência Resultante (φ_E).

$$\mu_{Es} = \begin{bmatrix} \mu_E \\ \varphi_E \end{bmatrix}$$

3.2 Rede de Análise Paraconsistente

Nós de Análise Paraconsistente – NAP interligados entre si compõem uma Rede de Análise Paraconsistente. Conforme a necessidade de maior ou menor precisão nas respostas, e

de tipos de fontes de informações disponíveis consideradas relevantes para a tomada de decisão, uma Rede de Análise Paraconsistente pode ser configurada interligando seus NAPs de várias maneiras.

Cada NAP pode fazer a análise de apenas uma única proposição.

Numa situação onde exista uma Proposição Objeto e que se deva analisar várias outras proposições parciais para que se possa determinar alguma coisa em relação à Proposição Objeto, cada NAP faria a análise de apenas uma proposição e os diversos NAPs devem ser conectados de tal forma que o resultado obtido seja do Grau de Certeza à Proposição Objeto.

A figura 3.3 mostra dois NAPs fazendo análises de duas proposições parciais P_1 e P_2 e interligados na Rede de Análise Paraconsistente para a análise de uma Proposição Objeto P_o.

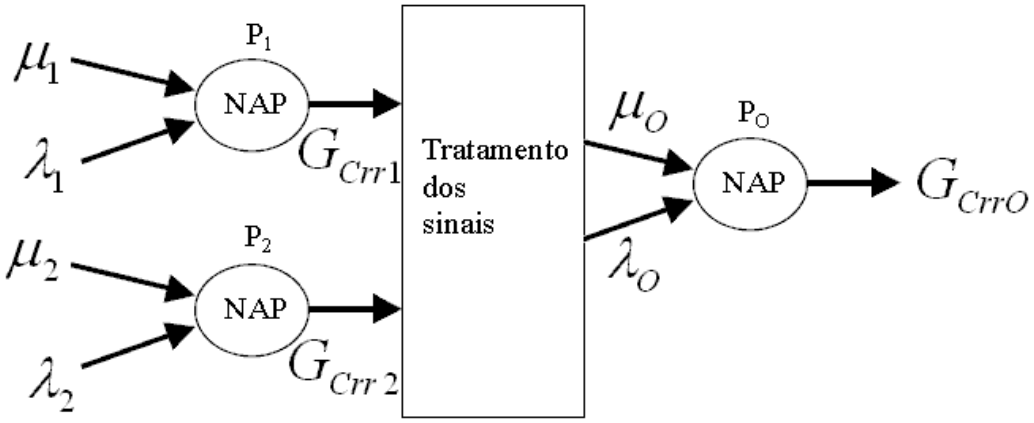


Figura 3.3 – Representação de uma Rede de Análise Paraconsistente composta por dois NAPs

A Proposição P_1, analisada no NAP1 produz um Grau de Evidência favorável μ_{Er1}, e a Proposição P_2, analisada no NAP2, produz outro Grau de Evidência μ_{Er2} que, após uma complementação transforma-se em Grau de Evidência desfavorável λ para a Proposição Objeto. Nestas interligações tem-se na saída de cada NAP o valor do Grau de Certeza Real

G_{cr} que deverá ser transformado pela normalização em Grau de Evidência μ. Dessa forma, o resultado pode ser utilizado como sinal de entrada em outros NAPs.

Segundo (DA SILVA, ABE & LAMBERT-TORRES) existem três regras básicas para se fazer as combinações de resultados dos NAPs:

1 – Proposições analisadas nos NAPs podem ser logicamente combinadas através dos Graus de Certeza reais resultantes originados das análises, e fazendo-se assim, as diferentes interligações na Rede de Análise Paraconsistente.

2 – Os valores dos Graus de Certeza reais resultantes, assim como os Intervalos de Certeza Reais originados dos NAPs referentes às diferentes proposições poderão ser tratados logicamente pelos conectivos de conjunção (AND), disjunção (OR), ou algebricamente por soma e subtração de seus valores, conforme características e topologia do projeto da Rede de Análise Paraconsistente.

3 – Os valores dos Graus de Certeza reais resultantes, poderão ser transformados por meio de normalização em valores entre 0 e 1 no intervalo real e assim considerados como Graus de Evidência de outras proposições que estão sendo analisados por outros diferentes NAPs. Desse modo as interligações entre os NAPs serão feitos através de análises de evidências.

É indesejável que o Grau de Evidência gerado pelo NAP assuma o valor de 0,5, pois, isso caracteriza uma indefinição. Essa situação pode acontecer quando houver uma alta contradição ou falta de informação na entrada do NAP.

Se a causa da indefinição for alta contradição na entrada do NAP, o Intervalo de Evidência Resultante (φ_E) pode ser utilizado para se determinar a origem da contradição. Sabe-se que quando φ_E assume um valor baixo, o Grau de Contradição é alto. Assim, quando o Intervalo de Evidência Resultante gerado pela NAP assumir um valor entre 0,25 e 0, por exemplo, a causa da indefinição é alta contradição na entrada.

A falta de informação na entrada do NAP pode ser ocasionada pelo fato da fonte da informação estar desligada. Numa situação como essa o resultado gerado pelo NAP não deve ser considerado, pois, contaminam o resultado final. Assim, utiliza-se a analise do φ_E e do Grau de Evidência gerado pelo NAP para desativar o NAP.

A desativação do NAP acontece quando φ_E for menor que 0,25, caracterizando alta contradição na entrada, e o Grau de Evidência for 0,5. Grau de Evidência gerado pelo NAP igual a 0,5 indica que o Grau de Certeza gerado pelo NAP é igual a zero. Esses dois fatores podem evidenciar que uma das entradas do NAP está desligada.

3.3 Uma Extensão da Lógica Paraconsistente Anotada Evidencial Eτ de 2 para 3 valores

As Células Neurais Artificiais Paraconsistentes trabalham com três valores de entrada, φ_E, μ e λ. Descreve-se em seguida uma abordagem apresentada em (MARTINS 2003) onde se apresenta uma forma de se manipular três variáveis de entradas com a Lógica Paraconsistente Anotada. Em seguida relacionam-se as ideias apresentadas (MARTINS 2003) as idéias propostas em (DA SILVA, ABE & LAMBERT-TORRES 2008).

A extensão para a Lógica Paraconsistente Anotada de Três Valores advém do adicionamento de um eixo perpendicular ao Quadrado Unitário do Plano Cartesiano. Chama-se este terceiro eixo de Grau de Especialidade (e).

Os valores de *e* variam no intervalo real fechado [0, 1]. Assim, um ponto obtido passa a ser interpretado como uma tripla (μ, λ, e). Representa-se no Cubo Unitário os pontos originados da tripla (μ, λ, e). A figura 3.4 mostra este cubo.

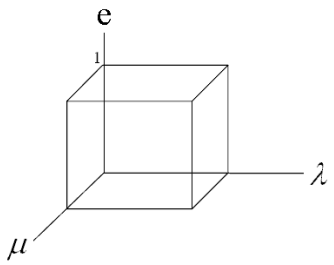


Figura 3.4 – Cubo Analisador Unitário

Os valores de *e* podem variar de 0 a 1. Quando *e* = 0, tem-se o Grau de Especialidade Mínimo, ou Neófito. Quando *e* = 1, tem-se o Grau de Especialidade Máximo, ou Especialista. Para valores intermediários têm-se Grau de Especialidades.

No Grau de Especialidade Máximo pressupõe-se que o especialista tenha condições de decidir de forma determinada sem inconsistências, indecisões ou desconhecimento. De tal forma que apenas dois estados são necessários, o Verdade e Falso relativo ao Especialista *e*, V_e e F_e respectivamente.

Já no Grau de Especialista Mínimo, o neófito, por não ter experiência alguma pode tomar qualquer decisão, correta, incorreta, inconsistente, indeterminada, ou seja, todos os estados são possíveis.

Em (DA SILVA, ABE & LAMBERT-TORRES) trata-se cada NAP como sendo um Cubo Analisador Paraconsistente. Mas, neste caso, o Grau de Especialidade é substituído pelo Intervalo de Evidência Resultante (φ_E) vindo de outro NAP e, também, a forma de se determinar os estados de saída é diferente.

A figura 3.5 mostra o símbolo representativo de um Cubo Analisador Paraconsistente.

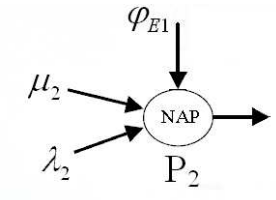


Figura 3.5 – Símbolo representativo do Cubo Analisador Paraconsistente

Representa-se nos eixos x e y o Grau de Evidência Favorável (μ_n) e o Grau de Evidência Contrária (λ_n) relativos a uma proposição qualquer n. No eixo z representa-se o Grau de Contradição Normalizado relativo a uma proposição qualquer m (μ_{ctrm}) obtido através da equação (3.7). A figura 3.6 mostra a idéia acima exposta.

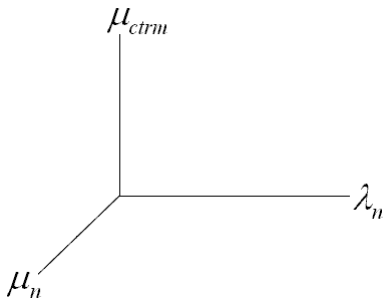


Figura 3.6 – Representação gráfica do Cubo Paraconsistente

Para cada valor de μ_{ctrm} têm-se um plano xy com diferentes estados de saída. Descreve-se a seguir a maneira de se determinar o plano xy para μ_{ctrm} igual a 0,5; 0,25; 0,0; 0,75 e 1,0 respectivamente.

A figura 3.7 mostra o plano xy formado pelos Graus de Evidência da proposição n quando μ_{ctrm} vale 0,5. Também se representa neste plano o Intervalo de Evidência Resultante da proposição m (φ_{Em}) e o Grau de Certeza da proposição n (G_{Cn}).

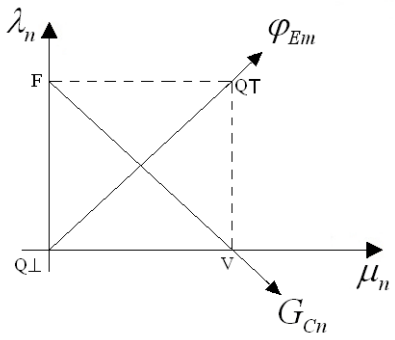


Figura 3.7 – Plano xy quando μ_{ctrm} vale 0,5

Para se determinar os estados de saída faz-se com que G_{Cn} tenha uma variação igual a φ_{Em}. Assim, quando μ_{ctrm} vale 0,5, ou seja, o Intervalo de Evidência Resultante da proposição m (φ_{Em}) vale 1, G_{Cn} também deve valer 1. Portanto, na situação apresentada pela figura 3.7, todos os pontos são passíveis de serem escolhidos.

Quando, por exemplo, μ_{ctrm} vale 0,25, ou seja, o Intervalo de Evidência Resultante da proposição m (φ_{Em}) vale 0,5. O Grau de Evidência Resultante obtido na proposição n poderá ter uma variação que totaliza 0,5, portanto, entre um mínimo de 0,25 e um máximo de 0,75. Da mesma forma, o Intervalo de Evidência Resultante da proposição m (φ_{Em}) poderá variar de 0,25 a 0,75. A figura 3.8 mostra esta situação.

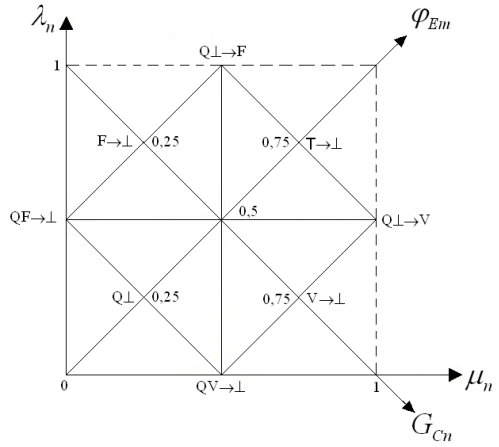


Figura 3.8 – Plano xy quando μ_{ctrm} vale 0,25

Assim, apenas os pontos contidos no quadrilátero formado pelos pontos QF→⊥, Q⊥→F, Q⊥→V e QV→⊥ são passíveis de serem considerados estados de saída. Caso o ponto determinado pelos graus de evidência da proposição n esteja fora do quadrilátero, considera-se o estado extremo mais próximo, ou seja, Q⊥, F→⊥, T→⊥ ou V→⊥.

Se μ_{ctrm} valer 0 ou 1, φ_{Em} vale também 0. Nesta situação a variação G_{Cn} deve ser também 0, já que a variação de G_{Cn} é igual a φ_{Em}. Assim, o plano xy resultante desta situação

contém apenas o estado paracompleto (⊥) localizado em seu centro como mostrado na figura 3.9.

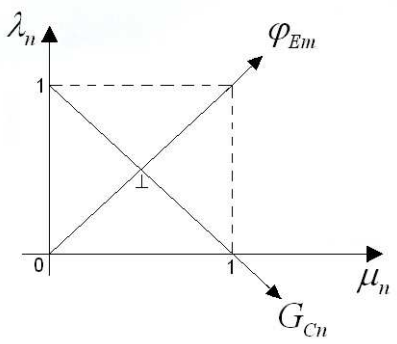


Figura 3.9 – Plano xy quando μ_{ctrm} vale 0,0

Quando μ_{ctrm} vale 0,75, ou seja, o Intervalo de Evidência Resultante da proposição m (φ_{Em}) vale 0,5. O Grau de Evidência Resultante obtido na proposição n poderá ter uma variação que totaliza 0,5, portanto, entre um mínimo de 0,25 e um máximo de 0,75. Da mesma forma, o Intervalo de Evidência Resultante da proposição m (φ_{Em}) poderá variar de 0,25 a 0,75. A figura 3.10 mostra esta situação.

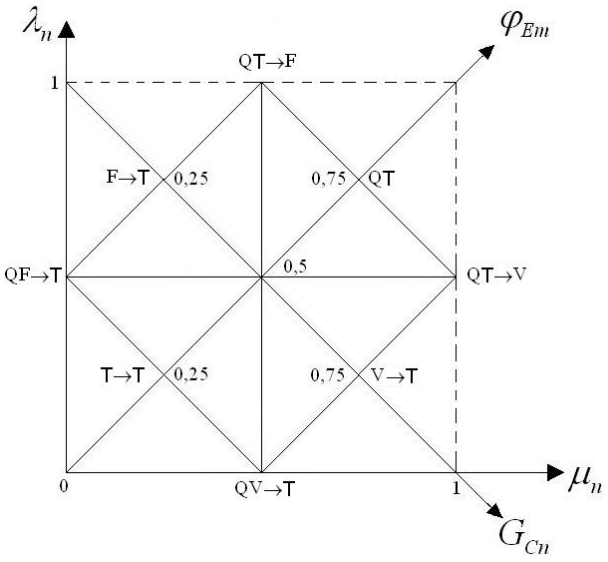


Figura 3.10 – Plano xy quando μ_{ctrm} vale 0,75

Então, apenas os pontos contidos no quadrilátero formado pelos pontos QF→T, QT→F, QT→V e QV→T são passíveis de serem considerados estados de saída. Caso o ponto determinado pelos graus de evidência da proposição n esteja fora do quadrilátero, considera-se o estado extremo mais próximo, ou seja, QT, V→F, T→ T ou F→T.

Obviamente que para valores de μ_{ctrm} diferentes de 0,5, 0,25, 0,0, 0,75 e 1,0, a forma de se determinar o plano xy é a mesma que a já descrita.

3.4 Células neurais artificiais paraconsistentes

Existem diversos tipos de Células Neurais Artificiais Paraconsistentes. As Redes Neurais Artificiais Paraconsistentes existentes nos Subsistemas de Sensoriamento e Planejamento utilizam a Célula Neural Artificial Paraconsistente Analítica – CNAPa, a Célula Neural Artificial Paraconsistente de Passagem – CNAPpa e a Célula Neural Artificial Paraconsistente de Conexão Lógica Simples no Processo de Maximização (OU) - CNAPmax. Assim, apenas essas três células são descritas nos próximos itens.

3.4.1 Célula neural artificial paraconsistente analítica – CNAPa

Basicamente a Célula Neural Artificial Paraconsistente Analítica – CNAPa recebe como entrada dois valores de graus de evidência e gera como saída um grau de evidência resultante – μ_E. A figura 3.11 mostra a representação gráfica dessa célula.

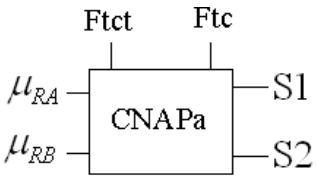


Figura 3.11 Célula neural artificial paraconsistente analítica – CNAPa

Os graus de evidência de entrada são:

μ_{RA}, tal que: $0 \leq \mu_{RA} \leq 1$.

μ_{RB}, tal que: $0 \leq \mu_{RB} \leq 1$.

Existem, também, dois valores de controle de entrada. São eles:

Fator de Tolerância à Contradição – Ft_{ct}, tal que: $0 \leq Ft_{ct} \leq 1$.

Fator de Tolerância à Certeza – Ft_c, tal que: $0 \leq Ft_c \leq 1$.

A Célula Neural Artificial Paraconsistente Analítica – CNAPa possui duas saídas.

Na saída 1 (S1) têm-se o Grau de Evidência Resultante - μ_E.

μ_E, tal que: $0 \leq \mu_E \leq 1$.

Na saída 2 (S2) têm-se o Intervalo de Evidência Resultante - φ_E.

φ_E, tal que: $0 \leq \varphi_E \leq 1$.

A Célula Neural Artificial Paraconsistente Analítica calcula o Valor de Controle de Veracidade - V_{cve}, o Valor de Controle de Falsidade - V_{cfa}, o Valor de Controle de Inconsistência - V_{cic} e o Valor de Controle de Paracompleteza - V_{cpa} da seguinte forma:

$$V_{cve} = \frac{1 + Ft_c}{2} \tag{3.10}$$

$$V_{cfa} = \frac{1 - Ft_c}{2} \tag{3.11}$$

$$V_{cic} = \frac{1 + Ft_{ct}}{2} \tag{3.12}$$

$$V_{cpa} = \frac{1 - Ft_{ct}}{2} \tag{3.13}$$

O Grau de Evidência Resultante - μ_E, se determina da seguinte forma:

$$\mu_E = \frac{G_c + 1}{2}$$

Como $G_c = \mu - \lambda$, pode-se afirmar que:

$$\mu_E = \frac{\mu - \lambda + 1}{2} \qquad (3.14)$$

Chama-se de Intervalo de Certeza (φ) o intervalo de valores do Grau de Certeza (G_C) que podem variar sem alterar o valor do Grau de Contradição (G_{ct}). Determina-se esse valor da seguinte forma:

$$\varphi = 1 - |G_{ct}| \qquad (3.15)$$

O Grau de Veracidade Máximo ($G_{ve_{\max}}$) é o próprio valor máximo positivo do Intervalo de Certeza, ou seja, $G_{ve_{\max}} = +\varphi$.

O Grau de Falsidade Máximo ($G_{fa_{\max}}$) é o valor máximo negativo do Intervalo de Certeza, portanto, $G_{fa_{\max}} = -\varphi$.

A figura 3.12 mostra a ideia acima exposta.

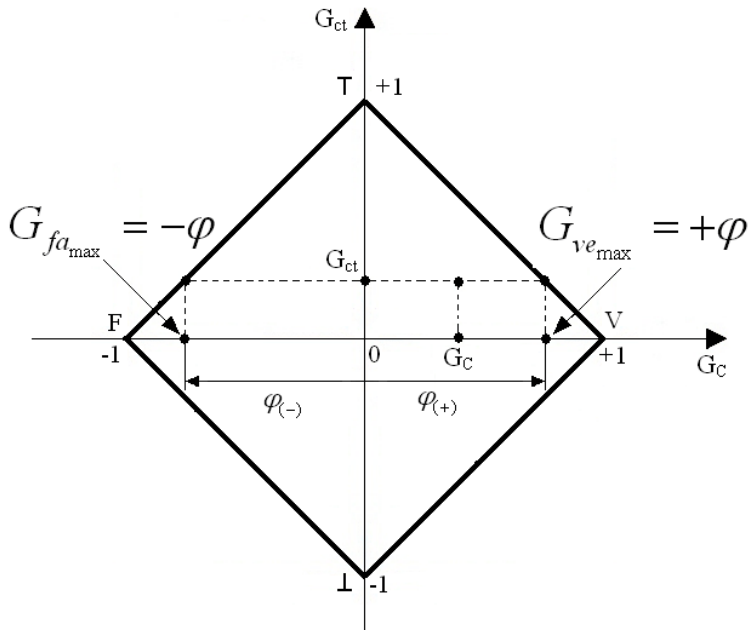


Figura 3.12 Representação no reticulado dos máximos valores de graus de certeza com grau de contradição constante

A CNAPa calcula o valor do Grau de Contradição Normalizado – $\mu_{ctr,}$, que é determinado da seguinte forma:

$$\mu_{ctr} = \frac{G_{ct} + 1}{2} \qquad (3.16)$$

Como $G_{ct} = (\mu + \lambda) - 1$, podemos afirmar que:

$$\mu_{ctr} = \frac{(\mu + \lambda - 1) + 1}{2}$$

$$\mu_{ctr} = \frac{\mu + \lambda}{2} \qquad (3.17)$$

Encontra-se o Intervalo de Certeza através da equação (3.15) repetida abaixo.

$\varphi = 1 - |G_{ct}|$

A partir da equação (3.16) pode-se afirmar que:

$$G_{ct} = 2\mu_{ctr} - 1 \qquad (3.18)$$

Assim, para se encontrar o valor do Intervalo de Evidência Resultante (φ_E), substitui-se a equação (3.18) em (3.15), portanto:

$$\varphi_E = 1 - |2.\mu_{ctr} - 1| \qquad (3.19)$$

Define-se o Intervalo de Evidência Resultante Sinalizado $\varphi_{E(\pm)}$ como sendo o valor de φ_E considerando a sinalização do Grau de Contradição (G_{ct}). Assim, quando $\mu_{ctr} > 0{,}5$, situação ocorrida quando $G_{ct} > 0$, têm-se $\varphi_{E(\pm)} = \varphi_{E(+)}$.

Da mesma forma, quando $\mu_{ctr} < 0{,}5$, situação ocorrida quando $G_{ct} < 0$, têm-se $\varphi_{E(\pm)} = \varphi_{E(-)}$.

Portanto, o valor do Grau de Contradição Normalizado pode ser encontrado a partir do Intervalo de Evidência Resultante Sinalizado através das seguintes equações:

$$\text{Se } \varphi_{E(\pm)} = \varphi_{E(+)},\ \mu_{ctr} = \frac{1 + (1 - \varphi_E)}{2} \qquad (3.20)$$

$$\text{Se } \varphi_{E(\pm)} = \varphi_{E(-)},\ \mu_{ctr} = \frac{1 - (1 - \varphi_E)}{2} \qquad (3.21)$$

A saída 2 sempre assume o valor de φ_E .

Já a saída 1 assume o valor de μ_E quando a seguinte condição é verdadeira:

$V_{cic} > \mu_{ctr} > V_{cpa}$ e $\left[(V_{cve} \leq \mu_E)\right.$ ou $\left.(\mu_E \leq V_{cfa})\right]$. Caso contrário, a saída 1 assume 0,5.

3.4.2 Célula neural artificial paraconsistente de passagem – CNAPpa

A Célula Neural Artificial Paraconsistente de Passagem – CNAPpa recebe como entrada um grau de evidência e possui como saída o valor desse mesmo grau de evidência. Mas, através do ajuste do Fator de Tolerância à Certeza, que é uma entrada de controle da célula, é possível limitar o valor da saída – S1. A figura 3.13 mostra a representação gráfica da Célula Neural Artificial Paraconsistente de Passagem.

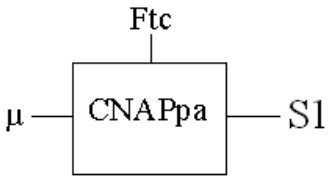


Figura 3.13 Representação gráfica da célula neural artificial paraconsistente de passagem – CNAPpa

Representa-se o grau de evidência por μ, tal que: $0 \leq \mu \leq 1$.

O Fator de Tolerância à Certeza – Ft_c é uma entrada de controle, tal que: $0 \leq Ft_c \leq 1$.

A CNAPpa calcula o Valor de Controle de Veracidade e o Valor de Controle de Falsidade conforme as equações (3.10) e (3.11) repetidas abaixo:

$$V_{cve} = \frac{1 + Ft_c}{2}$$

$$V_{cfa} = \frac{1 - Ft_c}{2}$$

A saída da CNAPpa é o Grau de Evidência Resultante, determinado conforme a equação (3.14), repetida a seguir:

$$\mu_E = \frac{\mu - \lambda + 1}{2}$$

A saída da CNAPpa assume o valor do Grau de Evidência Resultante - μ_E quando se satisfaz a condição: $\left[(V_{cve} \leq \mu_E) \text{ ou } (\mu_E \leq V_{cfa})\right]$. Caso contrário, μ_E vale 0,5.

3.4.3 Célula Neural Artificial Paraconsistente de Conexão Lógica Simples no Processo de Maximização (OU)

A figura 3.14 mostra a representação gráfica de uma Célula Neural Artificial Paraconsistente de Conexão Lógica Simples no Processo de Maximização (OU). Essa célula coloca na saída - $\mu_{\max}$ o maior valor entre os dois valores de entrada, μ_{RA} e μ_{RB}.

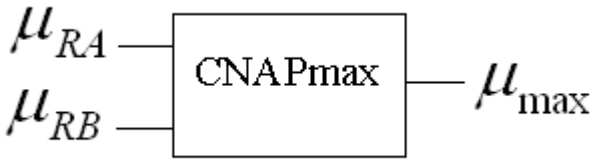


Figura 3.14 Representação gráfica da célula neural artificial paraconsistente de conexão lógica simples no processo de maximização (OU)

A Célula Neural Artificial Paraconsistente de Conexão Lógica Simples no Processo de Maximização (OU) determina o valor do Grau de Evidência Favorável pela equação (3.14) repetida abaixo.

Sendo que $\mu = \mu_{RA}$ e $\lambda = \mu_{RB}$.

Caso $\mu_E \geq 0{,}5$, conclui-se que a entrada μ_{RA} é a maior, portanto, a saída assume o valor de μ_{RA}. Caso contrário, a saída assume o valor μ_{RB}.

CAPÍTULO 4 - SUBSISTEMA MECÂNICO

4.1 Robôs Móveis Autônomos Baseados na Lógica Paraconsistente Anotada Evidencial Eτ

Descrevem-se mais detalhadamente a seguir os robôs móveis autônomos Emmy I e Emmy II. Eles serviram de inspiração para o sistema de navegação e controle proposto nesta tese.

4.1.1 Robô Móvel Autônomo Emmy I

O robô móvel autônomo Emmy I consiste de uma plataforma móvel de alumínio de formato circular de 30 cm de diâmetro e 60 cm de altura. O robô foi projetado em módulos sobrepostos separados por função no sistema de controle, facilitando a visualização da ação de cada módulo no controle de movimentação do robô. (DA SILVA FILHO, 1999), (DA SILVA FILHO & ABE, 1999), (DA SILVA FILHO & ABE, 1999b), (DA SILVA FILHO & ABE, 1999c), (DA SILVA FILHO & ABE, 2001), (DA SILVA FILHO & ABE, 2001b), (DA SILVA FILHO & ABE, 2001d), (DA SILVA FILHO, TORRES & ABE, 2006).

Na movimentação do robô Emmy I num ambiente não estruturado as informações sobre a existência ou não de obstáculo na sua trajetória são obtidas por intermédio do dispositivo denominado Parasônico. O Parasônico é capaz de captar obstáculos na trajetória transformando-os proporcionalmente as medidas de distância entre o robô e o obstáculo em sinais elétricos, na forma de uma tensão elétrica contínua que pode variar de 0 a 5 volts.

O Parasônico é basicamente composto por dois sensores de ultrassom tipo POLAROID 500 (POLAROID, 1996) e um Microcontrolador 8051. A função do

Microcontrolador 8051 é fazer a sincronização entre as medições dos dois sensores de ultrassom e a transformação da grandeza distância em tensão elétrica.

O sistema de controle de navegação do robô é feito por um Controlador Lógico Paraconsistente – Paracontrol e que recebe e faz o tratamento dos sinais elétricos.

O Paracontrol recebe informações na forma de graus de evidência favorável e contrária, faz uma análise paraconsistente que resulta em sinais representativos de estados lógicos e graus de certeza e de incerteza. As duas formas de saída podem ser utilizadas no controle, dependendo do projeto. O Paracontrol ainda oferece alternativa de ajustes de controle externamente através de potenciômetros.

Com a informação do estado resultante, o microcontrolador decide a ação a ser tomada pelo robô.

A figura 4.1 mostra as partes principais do robô Emmy I.

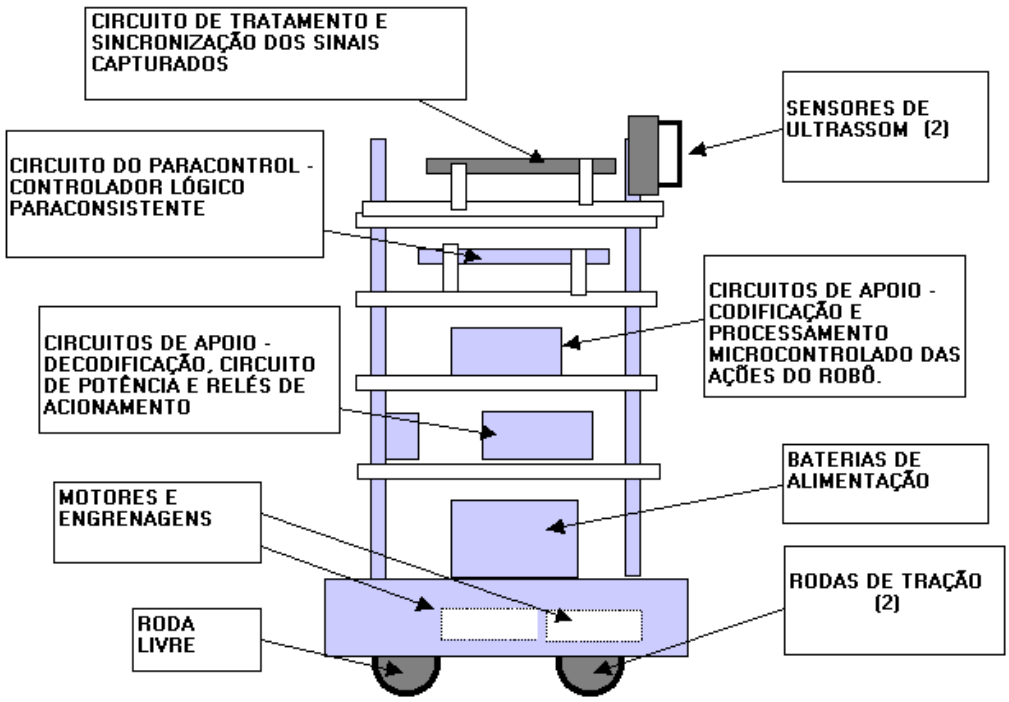


Figura 4.1 – Destaque das partes principais do robô Emmy

Nos trabalhos relacionados com controle clássico de robôs móveis autônomos descrevem-se técnicas tradicionais de planejamento e de navegação em roteiros conhecidos e desconhecidos. Muito dos resultados apresentam tempo computacional muito alto ou um sistema de controle muito complexo o que impossibilita a efetiva aplicação do controle. Porém, a maior dificuldade é o reconhecimento em situações reais de mudanças de ambiente, pois muitas indefinições e inconsistências aparecem nas análises dos sinais captados.

A Lógica Paraconsistente Anotada Evidencial Eτ possibilita considerar incertezas, inconsistências e paracompletezas de um modo não-trivial e por isso, se mostra mais propícia no enquadramento de problemas ocasionados por situações como essas.

O robô Emmy I obteve bons resultados apesar das limitações técnicas como ausência de: múltiplas velocidades, diferentes tipos de sensores, acionamentos de braços mecânicos, sincronização de velocidade entre os motores, amortecedores, freios, etc. Os testes efetuados demonstram que o Paracontrol pode ser aplicado para ajudar a solucionar problemas ligados a navegação de robôs e tratamento de sinais representativos de informações sobre o ambiente.

O sistema de controle utilizando o Paracontrol apresenta boa capacidade de modificar o comportamento do robô quando há modificações inesperadas das condições ambientais.

Pode-se ver na figura 4.2 o aspecto físico do robô Emmy I.

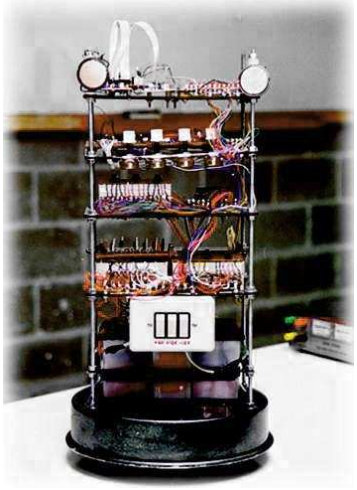


Figura 4.2 – Robô Emmy I

4.1.2 Robô Móvel Autônomo Emmy II

Investigando-se o controlador do robô Emmy I, chamado de Paracontrol, percebeu-se que o seu funcionamento poderia ser melhorado. Assim, surgiu o robô móvel autônomo Emmy II com objetivo de executar as mesmas funções que o robô Emmy I, mas, com um desempenho mais eficiente.

O robô móvel autônomo Emmy II possui as características mostradas na figura 4.3. (TORRES, 2004), (TORRES, ABE, & LAMBERT-TORRES, 2004), (TORRES, ABE & LAMBERT-TORRES, 2005), (TORRES, ABE & LAMBERT-TORRES, 2005b), (TORRES, ABE & LAMBERT-TORRES, 2006), (TORRES, LAMBERT-TORRES, SILVA & ABE, 2006), (ABE, TORRES, LAMBERT-TORRES, NAKAMATSU & KONDO, 2006), (ABE, TORRES, LAMBERT-TORRES, NAKAMATSU & KONDO, 2006b), (TORRES, ABE, LAMBERT-TORRES, 2007), (TORRES & BOMBACINI, 2007).

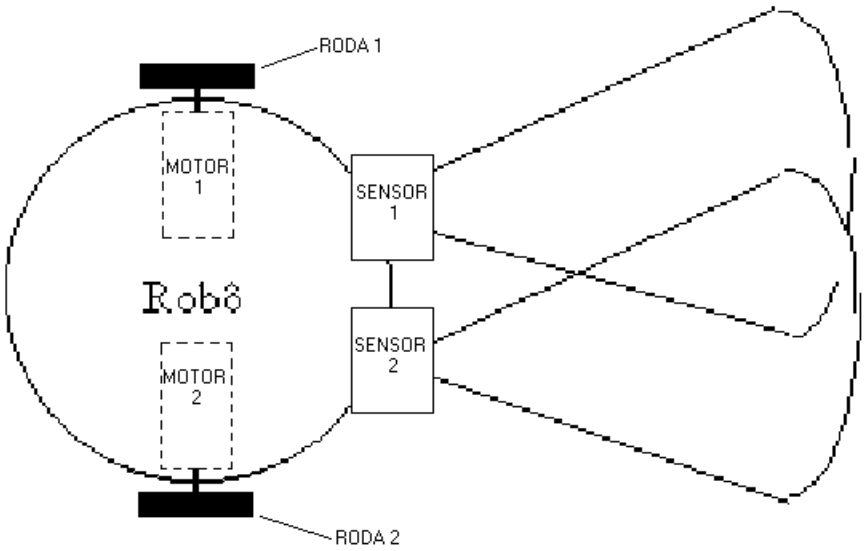


Figura 4.3 – Representação do robô Emmy II

O robô Emmy II constitui-se, basicamente, de dois sensores (S_1 e S_2) e dois motores (motor 1 e motor 2) com rodas presas a seus eixos. Esses elementos estão acoplados a uma plataforma circular de 25cm de diâmetro. Um microcontrolador é responsável por receber os sinais gerados pelos sensores de ultrassom, manipulando-os segundo o algoritmo Para-analisador, e acionar os motores.

Os movimentos possíveis neste robô são os seguintes:

1. Seguir em frente em linha reta. Os motores 1 e 2 são acionados para frente ao mesmo tempo com a mesma velocidade.

2. Ir para trás em linha reta. Os motores 1 e 2 são acionados para trás ao mesmo tempo com a mesma velocidade.

3. Girar para a direita. Apenas o motor 1 é acionado para frente com o motor 2 parado.

4. Girar para a esquerda. Apenas o motor 2 é acionado para frente com o motor 1 parado.

5. Girar para a direita. Apenas o motor 2 é acionado para trás com o motor 1 parado.

6. Girar para a esquerda. Apenas o motor 1 é acionado para trás com o motor 2 parado.

O sinal gerado pelo sensor S_1 considera-se como grau de evidência favorável e o sinal originado pelo sensor S_2 considera-se como grau de evidência contrária da proposição “A frente do robô está livre”. Quando existe um obstáculo próximo ao sensor S_1 o grau de evidência favorável é baixo e quando o obstáculo está distante do sensor S_1 o grau de evidência favorável é alto. Por outro lado, quando existe um obstáculo próximo ao sensor S_2 o grau de evidência contrária é alto e quando o obstáculo está distante do sensor S_2 o grau de evidência contrária é baixo. Nos capítulos posteriores descreve-se com detalhes como isso é feito.

O robô decide qual movimento escolher baseado nos valores do grau de evidência favorável, no grau de evidência contrária e no sistema de controle proposto de acordo com o reticulado com os respectivos estados lógicos extremos e não-extremos da figura 4.4.

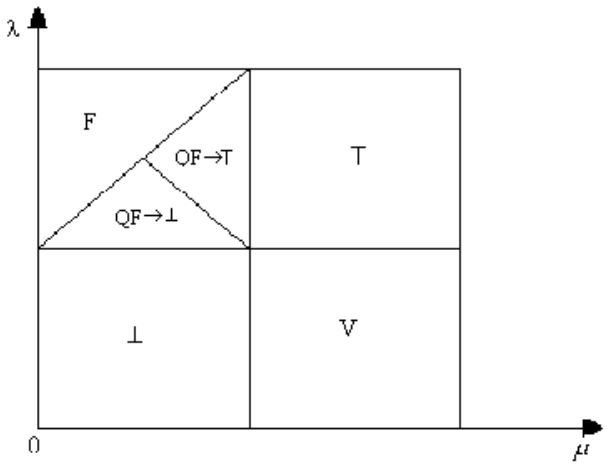


Figura 4.4 – Reticulado com os estados lógicos utilizado pelo robô Emmy II

A verificação dos valores do grau de evidência favorável e do grau de evidência contrária, tomada de decisão e movimentação dos motores é feita de forma sequencial. Tal sequência de ações é quase imperceptível ao se observar o robô movimentando-se.

Para cada estado, a respectiva decisão é a seguinte:

- Estado V: Ir para frente. Os motores 1 e 2 são acionados para frente[1] ao mesmo tempo.
- Estado F: Ir para trás. Os motores 1 e 2 são acionados para trás ao mesmo tempo.
- Estado ⊥: Girar para a direita. Apenas o motor 1 é acionado para frente. Permanecendo o motor 2 parado.
- Estado T: Girar para a esquerda. Apenas o motor 2 é acionado para frente. Permanecendo o motor 1 parado.

[1] Significa obviamente que as respectivas rodas giram fazendo com que o robô se movimente para frente. Este e outros abusos de linguagem são cometidos ao longo do trabalho.

- Estado QF→⊥: Girar para a direita. Apenas o motor 2 é acionado para trás. Permanecendo o motor 1 parado.

- Estado QF→⊤: Girar para a esquerda. Apenas o motor 1 é acionado para trás. Permanecendo o motor 2 parado.

As justificativas para as escolhas são as seguintes:

- Quando o estado for Verdadeiro (V), significa que a frente do robô está livre.

Assim, o robô pode seguir em frente.

- Na Inconsistência (⊤), μ e λ assumem valores altos (i.e., pertencentes à região ⊤). Isto significa que S_1 está distante de um obstáculo e S_2 está próximo de um obstáculo, sendo que o lado esquerdo está mais livre que o direito. Assim, a ação recomendada é girar para a esquerda. Aciona-se apenas o motor 2 para frente e mantendo o motor 1 parado.

- Quando for detectado o estado de Paracompleteza (⊥), μ e λ assumem valores baixos. Isto significa que S_1 está próximo de um obstáculo e S_2 está distante de um obstáculo, sendo que o lado direito está mais livre que o esquerdo. Portanto, a decisão deve ser girar para a direita. Aciona-se apenas o motor 1 para frente e mantendo o motor 2 parado.

- No estado de Falsidade (F), a frente do robô está obstruída por um o obstáculo muito próximo do robô. Portanto, a decisão é recuar.

- No estado Quase-falso tendendo ao Inconsistente (QF→⊤), a frente do robô continua obstruída, com as seguintes características: o obstáculo não está tão próximo como na Falsidade e o lado esquerdo está um pouco mais livre do que o direito. A decisão é girar para a esquerda acionando-se apenas o motor 1 para trás e mantendo-se o motor 2 parado.

- No estado Quase-falso tendendo ao Paracompleto (QF→⊥), a frente do robô continua obstruída, do seguinte modo: o obstáculo não está tão próximo como na falsidade e o lado direito está um pouco mais livre do que o esquerdo. A decisão é girar para a direita, acionando-se apenas o motor 2 para trás e mantendo-se o motor 1 parado.

A plataforma utilizada para a montagem do robô Emmy II possui aproximadamente 25cm de diâmetro e 23cm de altura. Os componentes principais do robô Emmy II são um microcontrolador da família 8051, dois sensores de ultrassom e dois motores de corrente contínua.

Os sensores de ultrassom são responsáveis pela verificação de obstáculos à frente do robô. Os sinais gerados pelos sensores são enviados para o microcontrolador. Os valores do grau de evidência favorável (μ) e do grau de evidência contrária (λ) na proposição "A frente do robô está livre" são determinados pelo microcontrolador com base nos sinais recebidos dos sensores de ultrassom. O microcontrolador também determina o movimento a ser executado pelo robô, ou seja, qual motor deve ser acionado, com base na decisão proveniente do Paracontrol.

A figura 4.5 mostra o diagrama em blocos simplificado do robô móvel autônomo Emmy II.

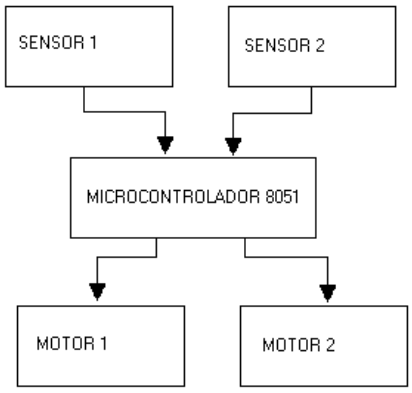


Figura 4.5 – Diagrama em blocos simplificado do robô Emmy II

O circuito eletrônico do robô Emmy II se divide em quatro partes:

- Circuito de alimentação.
- Circuito dos sensores.
- Circuito de controle.

- Circuito de atuação.

A figura 4.6 mostra a vista frontal do robô Emmy II, a figura 4.7 mostra os dois sensores ultrassônicos S_1 e S_2 e a figura 4.8 mostra os motores do robô.

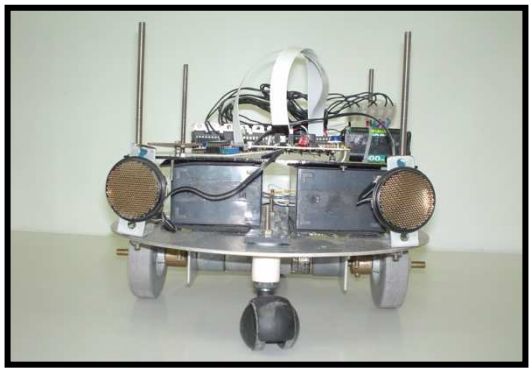


Figura 4.6 – Vista frontal do robô Emmy II

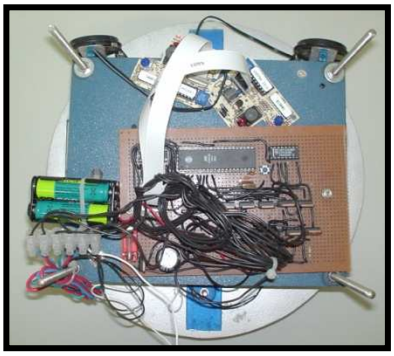


Figura 4.7 – Vista superior do robô Emmy II

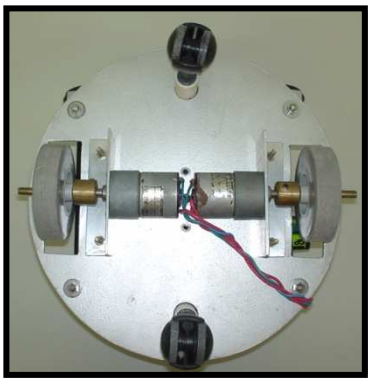


Figura 4.8 – Vista inferior do robô Emmy II

O robô Emmy II teve um desempenho satisfatório nos testes realizados. Ele conseguiu desviar da maioria dos obstáculos existentes no ambiente de teste.

4.2 Robô Móvel Autônomo Emmy III

Tanto o robô Emmy I quanto o robô Emmy II são capazes de desviar de obstáculos num ambiente não estruturado. Mas, são incapazes de encontrar um destino pré-determinado e se localizarem. O sistema de navegação e controle apresentado neste texto é uma tentativa de se construir um terceiro robô agora capaz de encontrar um destino num ambiente não estruturado de forma autônoma. Esse robô seria chamado Emmy III.

Alguns protótipos do robô Emmy III já foram construídos (DESIDERATO & DE OLIVEIRA, 2006), (MARAN, RIBA, COLLETT & DE SOUZA, 2006), (LEAL, DA SILVA & VIEIRA, 2009).

A versão definitiva do robô Emmy III está ainda em construção e compõe-se, basicamente, de uma placa metálica com dimensão de 400 mm de largura e 400 mm de comprimento, três rodas (sendo uma das rodas livre e as outras duas conectadas a servomotores) e os servomotores instalados na base são responsáveis pela movimentação do robô.

Conforme descrito nos capítulos iniciais, o objetivo deste trabalho é apresentar um sistema de navegação e controle de um robô móvel autônomo, e para isso, dividiu-se esse sistema em três módulos: Subsistema de Sensoriamento, Subsistema Planejamento e o Subsistema Mecânico.

No capítulo 5 será apresentado o Subsistema de Sensoriamento. Esse módulo considera o ambiente mapeado dividido em células inicialmente preparado para receber informações de sensores de ultrassom. Mas, com algumas alterações em sua configuração, o

Subsistema de Sensoriamento pode receber informações de outros tipos de sensores.

As informações advindas dos sensores são tratadas por uma Rede Neural Artificial Paraconsistente que possui como saída o Grau de Evidência Favorável da proposição “existe obstáculo na célula” para cada célula analisada. Essas informações são armazenadas num banco de dados.

No capítulo 6 será apresentado o Subsistema de Planejamento. Esse módulo é responsável por gerar a sequência de movimentos para que o robô móvel se desloque até uma posição destino. Serão definidos oito tipos diferentes de movimentos que esse sistema gera. Cada movimento corresponde a um número. Assim, a saída desse módulo é uma sequência numérica, sendo que cada número corresponde a um movimento diferente. A plataforma mecânica que compõe o robô, também chamada de Subsistema Mecânico, deve ser capaz de interpretar esses números e executar os movimentos correspondentes.

O Subsistema Mecânico apesar de não ser implementado nesse trabalho tem como base os trabalhos anteriores, onde foram construídas as plataformas Emmy I e Emmy II. Descreve-se a seguir o Subsistema Mecânico, que fica como sugestão para continuidade deste trabalho.

4.3 Subsistema mecânico

Uma estrutura física parecida com a do robô Emmy II, capaz de suportar os sensores e os outros dispositivos eletrônicos que compõem o Subsistema de Sensoriamento, o Subsistema de Planejamento e o sistema de controle dessa estrutura, compõem o Subsistema Mecânico.

O Subsistema Mecânico deve interpretar a sequência numérica gerada pelo Subsistema de Planejamento e executar seus respectivos movimentos. Além de tratar as informações dos

sensores antes de enviá-las ao Subsistema de Sensoriamento.

Como descrito no capítulo 5, o Subsistema de Sensoriamento utiliza como parâmetros de entrada a distância entre o sensor e o obstáculo (D) e o ângulo entre a plataforma mecânica e o eixo horizontal do ambiente onde o robô se encontra (α). Essas informações são geradas pelo Subsistema Mecânico, pois ele é responsável por tratar as informações dos sensores e controlar a posição da estrutura mecânica no ambiente onde o robô se localiza.

Para funcionar adequadamente, tanto o Subsistema de Sensoriamento quanto o Subsistema de Planejamento necessitam saber exatamente em qual célula o robô se encontra. Essa informação também deve ser gerada pelo Subsistema Mecânico.

Instalando-se o Subsistema de Sensoriamento e Planejamento numa estrutura física parecida com a descrita, ter-se-ia um robô móvel autônomo capaz de encontrar um destino num ambiente não estruturado.

O Subsistema Mecânico proposto baseado na estrutura mecânica do Robô Emmy II para a instalação dos Subsistemas de Sensoriamento e de Planejamento ficam como proposta de continuidade em trabalhos futuros.

CAPÍTULO 5 - SUBSISTEMA DE SENSORIAMENTO

Apresenta-se neste capítulo o Subsistema de Sensoriamento do projeto proposto. O Subsistema de Sensoriamento é construído com base na Lógica Paraconsistente Evidencial Eτ. Portanto, o seu funcionamento se baseia na determinação de graus de evidência relacionados a uma proposição.

Para cada coordenada do ambiente em torno do robô, o Subsistema de Sensoriamento deve ser capaz de determinar um Grau de Evidência Favorável a respeito da proposição: “existe obstáculo na coordenada”.

Os valores desses Graus de Evidência Favorável são armazenados no banco de dados para serem posteriormente utilizados como informações de entrada do Subsistema de Planejamento. Desta forma, o Subsistema de Planejamento, antes de gerar a sequência de ações que a plataforma mecânica deve executar, consulta o banco de dados e considera como obstruídas as coordenadas que possuírem um grau de evidência superior a determinado valor. Assim, a trajetória gerada evita os obstáculos existentes no ambiente, habilitando o robô a se movimentar num ambiente não estruturado.

5.1 Sistemas de sensoriamento tradicionais

Descrevem-se agora algumas técnicas tradicionais de mapeamento de ambientes ainda muito utilizadas na construção de robôs móveis autônomos. Essas técnicas serviram de inspiração para o sistema de mapeamento e controle apresentados neste trabalho.

Em ELFES (1989), apresenta-se uma abordagem de mapeamento de ambiente baseado em técnicas estocásticas. A proposta é dividir o ambiente em células, como mostrado na figura 5.1, e se calcular a probabilidade de existência de obstáculos em cada célula.

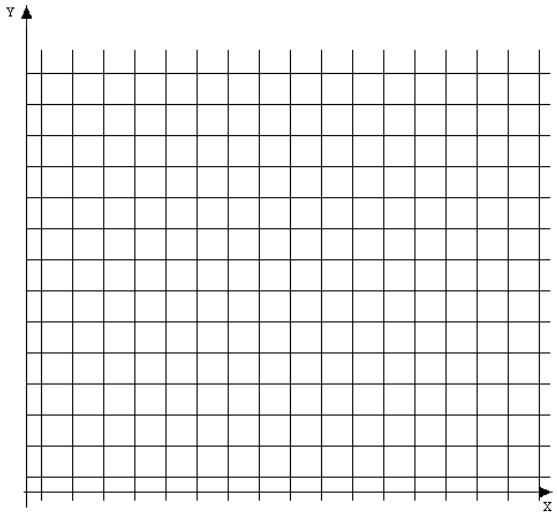


Figura 5.1 Ambiente dividido em células

O sistema de mapeamento e controle de robôs móveis autônomos apresentado neste trabalho também considera o ambiente dividido em células. Então, daqui para frente, neste texto, considera-se o ambiente dividido em células, não mais em coordenadas. Percebe-se que uma célula nada mais é do que a região em volta de uma coordenada.

Mas, ao invés de técnicas estocásticas, baseia-se o sistema de navegação e controle proposto na Lógica Paraconsistente Anotada Evidencial Eτ.

O Subsistema de Sensoriamento proposto manipula informações advindas de sensores de ultrassom. As ideias apresentadas em (BOREINSTEIN & KOREN, 1991), (SHOVAL, ULRICH & BORENSTEIN, 2003) por sua vez baseadas em (ELFES, 1989), serviram de inspiração para o Subsistema de Sensoriamento proposto.

Basicamente, a maneira de tratar as informações advindas de sensores de ultrassom apresentada por (BOREINSTEIN & KOREN, 1991) é a seguinte: sabe-se que as ondas de ultrassom emitidas pelos sensores espalham-se em forma de cone, conforme mostrado na figura 5.2.

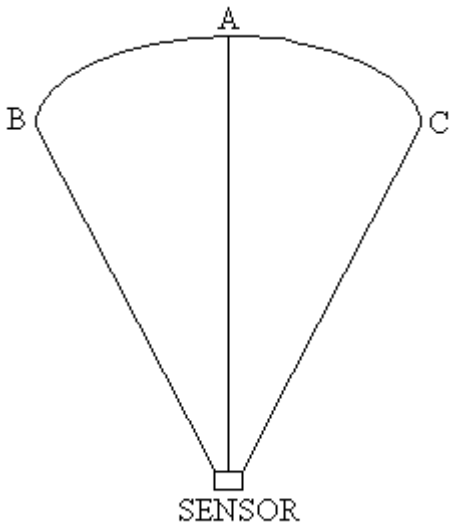


Figura 5.2 Cone formado pelas emissões de ondas de ultrassom por sensores

Assim, caso detecte-se um obstáculo a uma distância exata entre o sensor e o arco BC, o obstáculo pode estar em qualquer ponto do arco BC, sendo que existe uma probabilidade maior que o obstáculo esteja no ponto A. À medida que se desloque do ponto A em direção ao ponto B ou C, essa probabilidade vai diminuindo gradativamente até chegar a um valor próximo de zero nos pontos B e C.

Caso o sensor esteja num ambiente mapeado por células, teríamos uma situação parecida com a mostrada na figura 5.3.

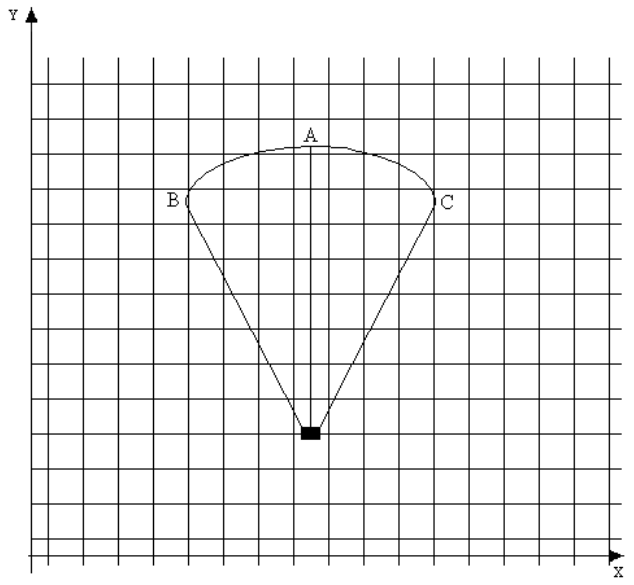


Figura 5.3 Sensor de ultrassom num ambiente dividido em células

O sistema proposto por (BOREINSTEIN & KOREN, 1991) indica uma probabilidade maior de existir obstáculo na célula A. Essa probabilidade decresce gradativamente nas células localizadas sob o arco AC e AB.

O Subsistema de Sensoriamento proposto neste trabalho funciona de forma muito parecida. Mas, para cada célula existe um Grau de Evidência Favorável sobre a proposição "existe obstáculo na célula".

5.2 Subsistema de sensoriamento proposto

O objetivo do Subsistema de Sensoriamento é informar o Subsistema de Planejamento quais células estão obstruídas, portanto, que se deve evitar.

Para isso, o Subsistema de Sensoriamento se utiliza de um conjunto de sensores responsáveis por capturar informações sobre o ambiente em torno do robô. Esse conjunto de sensores está localizado na plataforma mecânica. Assim, o Subsistema Mecânico trata os sinais gerados pelos sensores antes de fornecer as devidas informações ao Subsistema de Sensoriamento.

Esses sensores podem ser dos mais diversos tipos e seus dados deverão ser interpretados como informações relevantes para o Sistema de Navegação e Controle. Como esses dados representam informações que podem ser contraditórias, neste trabalho optou-se por utilizar as Redes Neurais Artificiais Paraconsistentes para tratar as informações geradas pelos sensores.

5.2.1 Estrutura do subsistema de sensoriamento proposto

O objetivo básico do Subsistema de Sensoriamento é informar a situação de cada

célula que compõe o ambiente em torno do robô. Assim, o Subsistema de Sensoriamento deve analisar as informações geradas pelos sensores e armazenar, no banco de dados do sistema, a situação de cada célula analisada. A figura 5.4 mostra graficamente as funções do Subsistema de Sensoriamento.

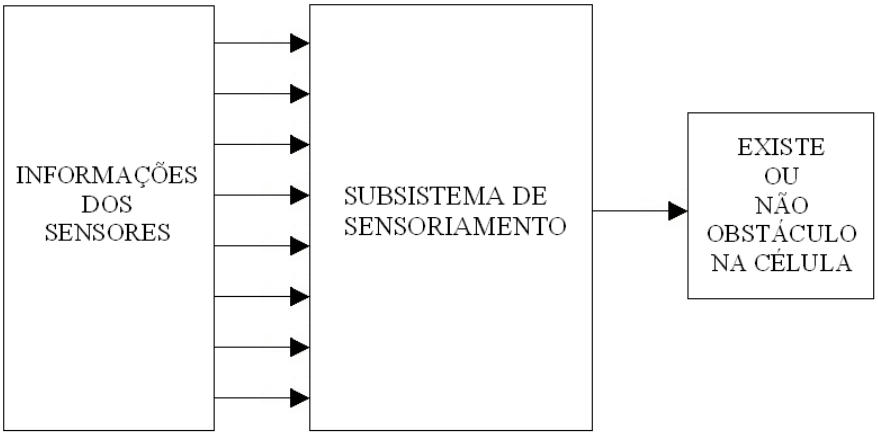


Figura 5.4 Representação gráfica do subsistema de sensoriamento

O Subsistema de Sensoriamento se baseia na Lógica Paraconsistente Anotada Evidencial Eτ. Portanto, as informações geradas pelos sensores são consideradas Graus de Evidência. Assim, para a análise de cada célula, a saída do Subsistema de Sensoriamento é o Grau de Evidência Favorável sobre a proposição "existe obstáculo na célula". A figura 5.5 mostra a representação gráfica do Subsistema de Sensoriamento já utilizando as notações relativas à Lógica Paraconsistente Anotada Evidencial Eτ.

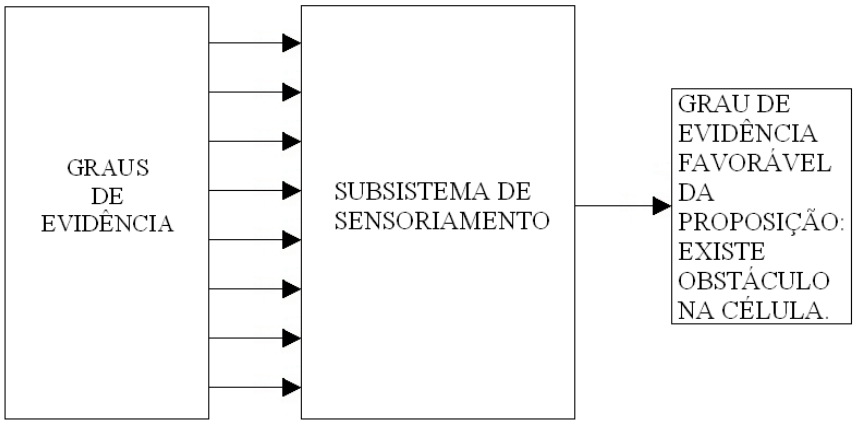


Figura 5.5 Representação gráfica do subsistema de sensoriamento com as notações da Lógica Paraconsistente Anotada Evidencial Eτ

O Subsistema de Sensoriamento divide-se em duas partes. A primeira parte é responsável por analisar as informações dos sensores e gerar três Graus de Evidência. A segunda parte do Subsistema de Sensoriamento é onde se localiza a Rede Neural Artificial Paraconsistente, responsável por analisar os três Graus de Evidências gerados pela primeira parte e criar o Grau de Evidência Favorável da proposição “existe obstáculo na célula”, que deve ser armazenado no banco de dados do sistema. A figura 5.6 mostra a estrutura do Subsistema de Sensoriamento.

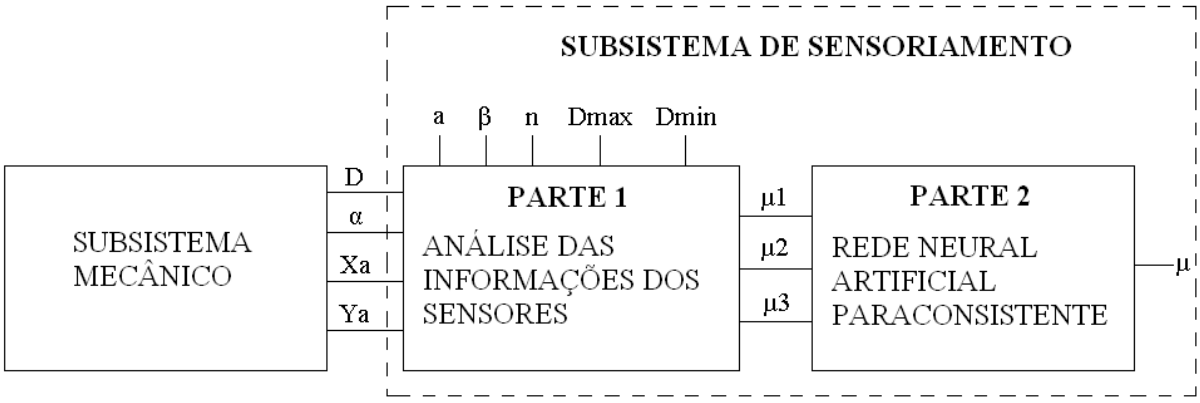


Figura 5.6 Estrutura do subsistema de sensoriamento

Os parâmetros a, β, n, Dmax e Dmin, a serem definidos a seguir, são de configuração. As informações recebidas do Subsistema Mecânico são representadas pelas entradas D, α, Xa e Ya. Os três Graus de Evidência gerados pela parte 1 do Sistema de Sensoriamento são representados por μ1, μ2 e μ3. O Grau de Evidência da proposição “existe obstáculo na célula”, que deve ser armazenado no banco de dados do sistema, é representado pela saída μ.

Encontra-se no Anexo 1 o programa em Linguagem Python do Subsistema de Sensoriamento.

5.2.1.1 Parâmetros de configuração do subsistema de sensoriamento proposto

O Sistema considera as células que compõe o ambiente em torno do robô como sendo quadradas. E o parâmetro “a” define o lado do quadrado que compõe a célula. A figura 5.7 mostra o significado desse parâmetro.

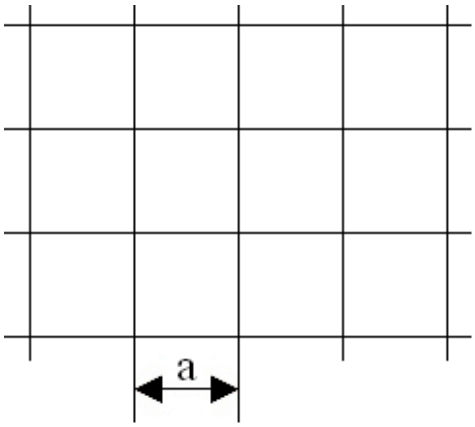


Figura 5.7 Representação do parâmetro “a”

Como o Subsistema de Sensoriamento foi construído para receber informações advindas de sensores de ultrassom, faz-se necessário configurar o subsistema para o tipo exato de sensor utilizado. Assim, o parâmetro “β” define o ângulo de abertura do sensor. Esse ângulo pode variar conforme o fabricante. A figura 5.8 mostra o significado desse parâmetro.

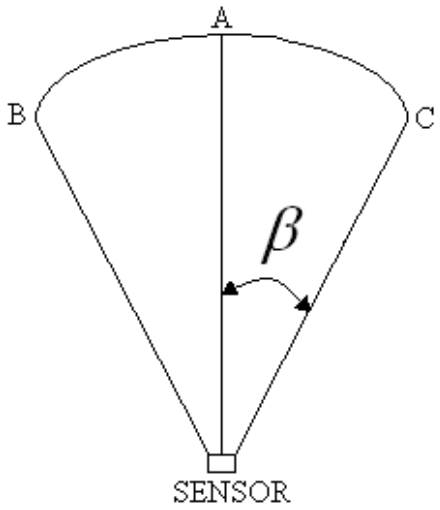


Figura 5.8 Representação do ângulo de abertura do sensor de ultrassom

É interessante observar que o Subsistema de Sensoriamento pode receber informações de um sensor de infravermelho, por exemplo, desde que o ângulo β seja considerado como 0^o.

Numa situação como a representada pela figura 5.9, o Subsistema de Sensoriamento analisa todas as células sob o arco BC. Assim, o parâmetro "n" define quantas vezes o Grau de Evidência é calculado tomando-se parte do arco AC ou AB.

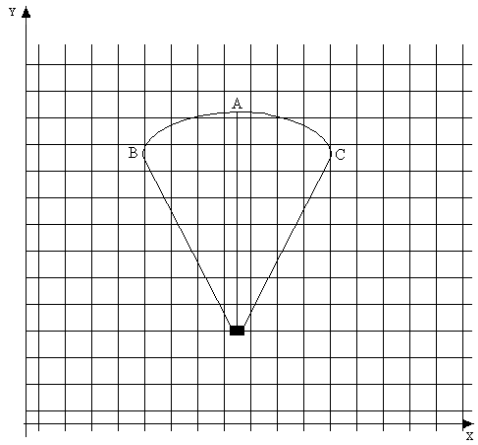


Figura 5.9 Sensor de ultrassom num ambiente dividido em células

O parâmetro Dmax define a distância máxima entre o obstáculo e o sensor que o Subsistema de Sensoriamento considera. E o parâmetro Dmin define a distância mínima entre o obstáculo e o sensor que o Subsistema considera.

5.2.1.2 Dados de entrada do subsistema de sensoriamento proposto

A entrada D representa a distância do obstáculo detectada pelo sensor.

Já a entrada "α" representa o ângulo entre o sensor e o eixo horizontal do ambiente. A figura 5.10 ilustra essa ideia. Essa informação é gerada pelo Subsistema Mecânico, pois, ele recebe os dados dos sensores e controla a localização e a direção da plataforma, possibilitando, assim, a determinação do ângulo α.

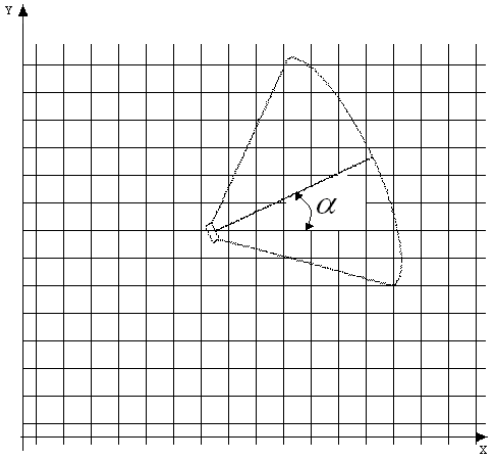


Figura 5.10 Representação da variável *α*

O Subsistema de Sensoriamento deve saber a coordenada onde a plataforma mecânica se encontra para determinar corretamente qual célula deve ser analisada. As entradas Xa e Ya representam essa coordenada. Essa informação também é gerada pelo Subsistema Mecânico.

A parte 1 do Subsistema de Sensoriamento gera como saída os três graus de evidência μ1, μ2 e μ3.

A distância entre o sensor e o obstáculo determina o valor de μ1. Quando a distância entre o sensor e o obstáculo (D) for menor ou igual a Dmin, μ1 vale 1,0. Quando a distância entre o sensor e o obstáculo for maior ou igual a Dmax, μ1 vale 0,0. Caso a distância entre o sensor e o obstáculo seja um valor entre Dmin e Dmax, μ1 será um valor entre 1,0 e 0,0, dependendo da proximidade de D em relação a Dmin.

Analisando a figura 5.9, percebe-se que existe uma evidência forte de que o obstáculo esteja próximo do ponto A. Essa evidência vai enfraquecendo à medida que se aproxime do ponto B ou C. Esse comportamento determina o valor de μ2. Quanto mais próxima a célula analisada estiver do ponto A, maior será o valor de μ2. Consequentemente, quanto mais próxima a célula analisada estiver dos pontos B ou C, menor será o valor de μ2.

Antes de determinar o novo Grau de Evidência Favorável da célula analisada, o Subsistema de Sensoriamento consulta o valor atual armazenado no banco de dados do

sistema. O valor consultado é quanto vale μ3.

5.2.2 Estrutura da rede neural artificial paraconsistente utilizada no subsistema de sensoriamento proposto

A figura 5.11 mostra a estrutura da Rede Neural Artificial Paraconsistente escolhida para compor a segunda parte do Subsistema de Sensoriamento.

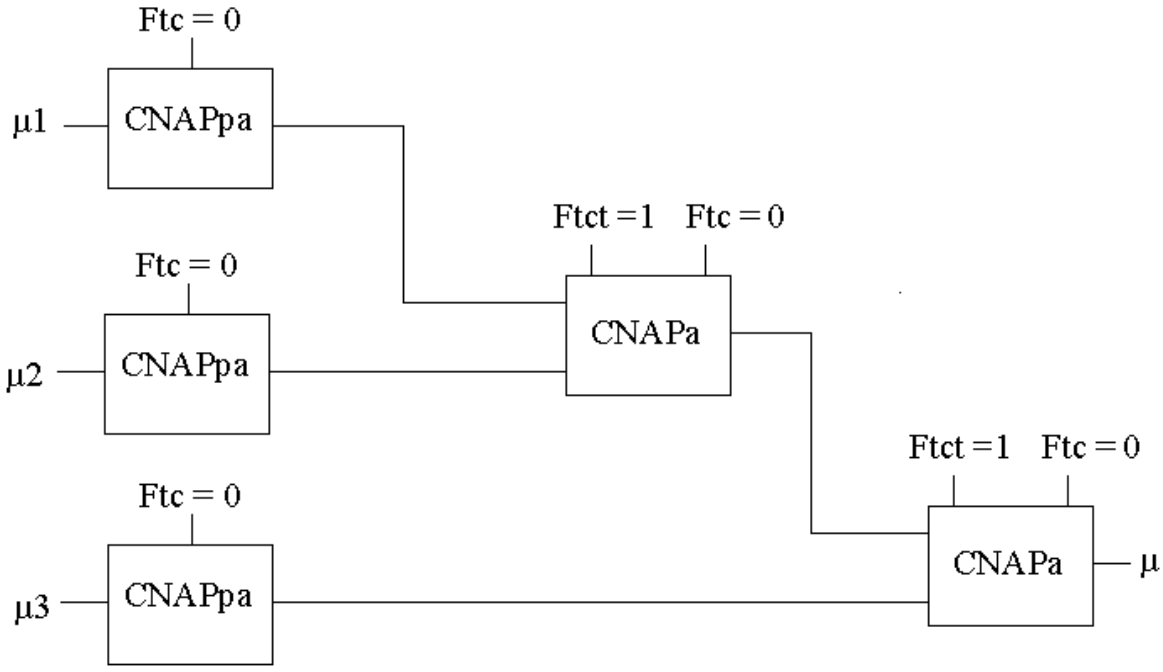


Figura 5.11 Estrutura da rede neural artificial paraconsistente escolhida para compor a segunda parte do subsistema de sensoriamento

Observa-se que μ1, μ2 e μ3 são os três graus de evidência gerados pela primeira parte do Subsistema de Sensoriamento. E μ é a resposta do sistema, ou seja, o valor do Grau de Evidência Favorável da proposição “existe obstáculo na célula”. Portanto, μ é o valor que deve ser armazenado no banco de dados do sistema informando o Grau de Evidência Favorável da Célula Analisada.

A função das Células Neurais Artificiais Paraconsistentes de Passagem – CNAPpa é de, eventualmente, dependendo do valor do Fator de Tolerância à Certeza - Ft_c, limitar o valor

de algum grau de evidência gerado pela primeira parte do Subsistema de Sensoriamento. Como no sistema implementado, o Fator de Tolerância à Certeza - Ft_c vale zero nas três células, não existe qualquer tipo de limitação para μ1, μ2 e μ3.

Configurando-se as Células Neurais Artificiais Paraconsistentes Analítica – CNAPa com $Ft_c = 0$ e $Ft_{ct} = 1$, também se elimina qualquer tipo de limitação às saídas das células.

5.2.3 Resultados dos testes realizados no subsistema de sensoriamento proposto

Apresentam-se a seguir o resultado de alguns testes realizados com o Subsistema de Sensoriamento. Os dados de entradas foram gerados manualmente simulando informações advindas do Subsistema Mecânico. Numa situação real, esses dados seriam gerados pelo Subsistema Mecânico. Cada coordenada analisada pelo Subsistema de Sensoriamento foi nomeada com uma letra. As figuras no final do capitulo mostram graficamente as localizações dessas células.

Teste 1

Configurações Iniciais do Subsistema:

- Distância entre coordenadas (a): 10
- Ângulo de abertura do sensor (β): 30
- Número de células no arco de abertura do sensor (n): 10
- Distância máxima medida pelo sensor (Dmax): 800
- Distância mínima medida pelo sensor (Dmin): 8

Configurações das Células Neurais Artificiais Paraconsistentes:

- Fator de Tolerância à Certeza da 1ª CNAPpa: 0

- Fator de Tolerância à Certeza da 2ª CNAPpa: 0
- Fator de Tolerância à Certeza da 3ª CNAPpa: 0
- Fator de Tolerância à Contradição da 1ª CNAPa: 1
- Fator de Tolerância à Certeza da 1ª CNAPa: 0
- Fator de Tolerância à Contradição da 2ª CNAPa: 1
- Fator de Tolerância à Certeza da 2ª CNAPa: 0

Dados de entrada:

- Distância entre o obstáculo e o sensor (D): 400
- Ângulo do sensor em relação ao eixo x (α): 45
- Coordenada x atual (onde o robô se encontra) (Xa): 0
- Coordenada y atual (onde o robô se encontra) (Ya): 0

Resposta do Subsistema:

- Coordenada Central:

Coordenada A (29,29): μ1 = 0.5 | μ2 = 1 | μ3 = 0 |Saída da RNAP = 0.375

- Coordenadas da Esquerda:

Coordenada B (27,30): μ1 = 0.5 | μ2 = 0.9 | μ3 = 0| Saída da RNAP = 0.35

Coordenada C (26,32): μ1 = 0.5 | μ2 = 0.8 | μ3 = 0| Saída da RNAP = 0.325

Coordenada D (24,33): μ1 = 0.5 | μ2 = 0.7 | μ3 = 0| Saída da RNAP = 0.3

Coordenada E (22,34): μ1 = 0.5 | μ2 = 0.6 | μ3 = 0| Saída da RNAP = 0.275

Coordenada F (20,35): μ1 = 0.5 | μ2 = 0.5 | μ3 = 0| Saída da RNAP = 0.25

Coordenada G (19,36): μ1 = 0.5 | μ2 = 0.4 | μ3 = 0| Saída da RNAP = 0.225

Coordenada H (17,37): μ1 = 0.5 | μ2 = 0.3 | μ3 = 0| Saída da RNAP = 0.2

Coordenada I (15,38): μ1 = 0.5 | μ2 = 0.2 | μ3 = 0| Saída da RNAP = 0.175

Coordenada J (13,39): μ1 = 0.5 | μ2 = 0.1 | μ3 = 0| Saída da RNAP = 0.15

Coordenada K (11,39): μ1 = 0.5 | μ2 = 1.38777878078e-016 | μ3 = 0 | Saída

da RNAP = 0.125

- Coordenadas da Direita:

Coordenada L (30,27): μ1 = 0.5 | μ2 = 0.9 | μ3 = 0| Saída da RNAP = 0.35

Coordenada M (32,26): μ1 = 0.5 | μ2 = 0.8 | μ3 = 0| Saída da RNAP = 0.325

Coordenada N (33,24): μ1 = 0.5 | μ2 = 0.7 | μ3 = 0| Saída da RNAP = 0.3

Coordenada O (34,22): μ1 = 0.5 | μ2 = 0.6 | μ3 = 0| Saída da RNAP = 0.275

Coordenada P (35,20): μ1 = 0.5 | μ2 = 0.5 | μ3 = 0| Saída da RNAP = 0.25

Coordenada Q (36,19): μ1 = 0.5 | μ2 = 0.4 | μ3 = 0| Saída da RNAP = 0.225

Coordenada R (37,17): μ1 = 0.5 | μ2 = 0.3 | μ3 = 0| Saída da RNAP = 0.2

Coordenada S (38,15): μ1 = 0.5 | μ2 = 0.2 | μ3 = 0| Saída da RNAP = 0.175

Coordenada T (39,13): μ1 = 0.5 | μ2 = 0.1 | μ3 = 0| Saída da RNAP = 0.15

Coordenada U (39,11): μ1 = 0.5 | μ2 = 1.38777878078e-016 | μ3 = 0 | Saída da RNAP = 0.125

A figura 5.12 mostra graficamente o resultado do Teste 1 realizado com o Subsistema de Sensoriamento. Percebe-se que a coordenada K e U possuem os mesmos valores.

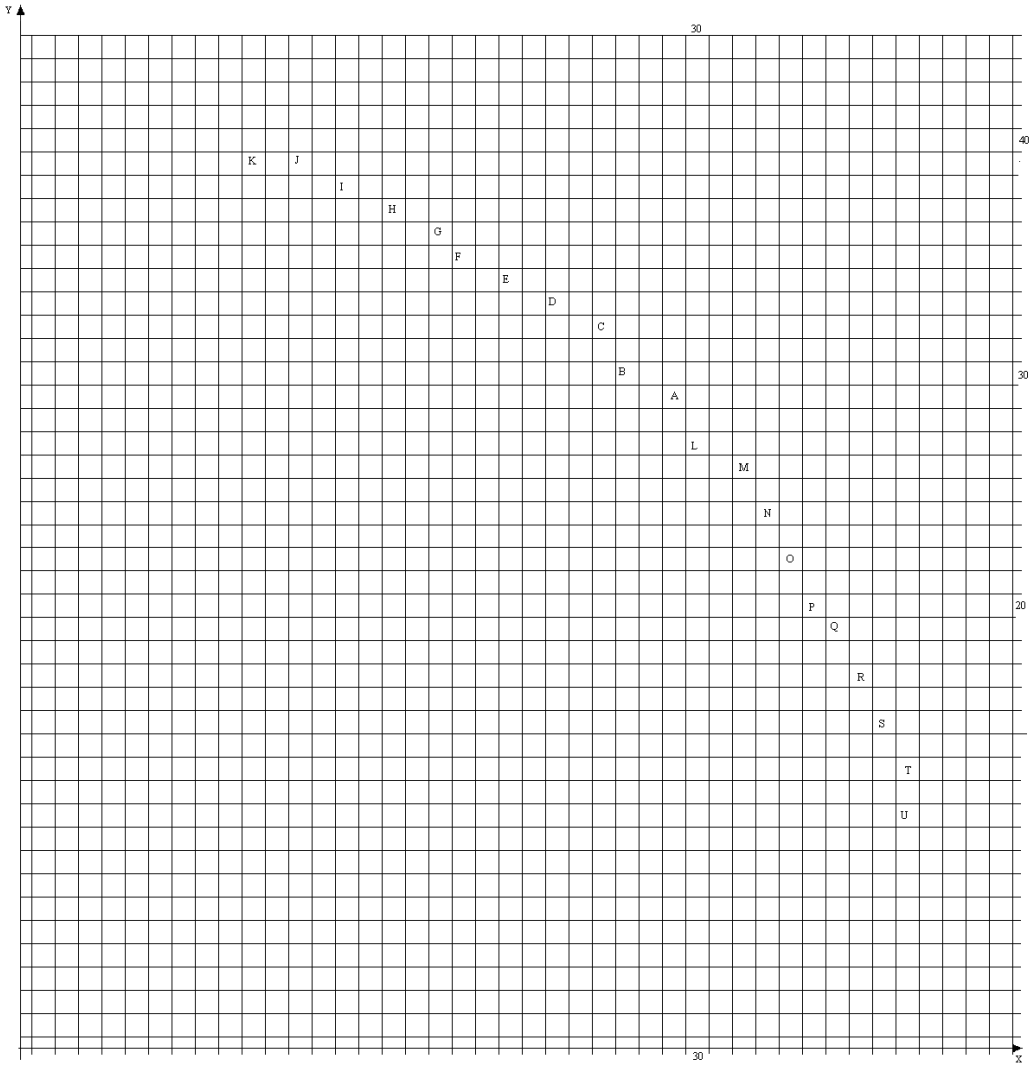


Figura 5.12 Resultado do teste 1 realizado com o subsistema de sensoriamento

Teste 2

O teste 2 utilizou a mesma configuração do teste 1. Além disso, os dados do sensor também são os mesmos do teste 1. Portanto, as células analisadas pelo teste 2 são as mesmas do teste 1. Assim, como μ3 é o valor do Grau de Evidência Resultante armazenado no banco de dados do sistema, observa-se que todos os valores de μ3 no teste 2 equivalem ao valor de saída da respectiva célula no teste 1.

Dados de Entrada:

- Distância entre o obstáculo e o sensor (D): 400
- Ângulo do sensor em relação ao eixo x (α): 45

- Coordenada x atual (onde o robô se encontra) (Xa): 0
- Coordenada y atual (onde o robô se encontra) (Ya): 0

Resposta do Sistema:

- Coordenada Central

Coordenada A (29,29): μ1 = 0.5 | μ2 = 1 | μ3 = 0.38| Saída da RNAP = 0.565

- Coordenadas da Esquerda

Coordenada B (27,30): μ1 = 0.5 | μ2 = 0.9 | μ3 = 035 | Saída da RNAP = 0.525

Coordenada C (26,32): μ1 = 0.5 | μ2 = 0.8 | μ3 = 033 | Saída da RNAP = 0.49

Coordenada D (24,33): μ1 = 0.5 | μ2 = 0.7 | μ3 = 030 | Saída da RNAP = 0.45

Coordenada E (22,34): μ1 = 0.5 | μ2 = 0.6 | μ3 = 028 | Saída da RNAP = 0.415

Coordenada F (20,35): μ1 = 0.5 | μ2 = 0.5 | μ3 = 025 | Saída da RNAP = 0.375

Coordenada G (19,36): μ1 = 0.5 | μ2 = 0.4 | μ3 = 023 | Saída da RNAP = 0.34

Coordenada H (17,37): μ1 = 0.5 | μ2 = 0.3 | μ3 = 020 | Saída da RNAP = 0.3

Coordenada I (15,38): μ1 = 0.5 | μ2 = 0.2 | μ3 = 018 | Saída da RNAP = 0.265

Coordenada J (13,39): μ1 = 0.5 | μ2 = 0.1 | μ3 = 015 | Saída da RNAP = 0.225

Coordenada K (11,39): μ1 = 0.5 | μ2 = 1.38777878078e-016 | μ3 = 0.13 | Saída da RNAP = 0.19

- Coordenadas da Direita

Coordenada L (30,27): μ1 = 0.5 | μ2 = 0.9 | μ3 = 035 | Saída da RNAP = 0.525

Coordenada M (32,26): μ1 = 0.5 | μ2 = 0.8 | μ3 = 033 | Saída da RNAP = 0.49

Coordenada N (33,24): μ1 = 0.5 | μ2 = 0.7 | μ3 = 030 | Saída da RNAP = 0.45

Coordenada O (34,22): μ1 = 0.5 | μ2 = 0.6 | μ3 = 028 | Saída da RNAP = 0.415

Coordenada P (35,20): μ1 = 0.5 | μ2 = 0.5 | μ3 = 025 | Saída da RNAP = 0.375

Coordenada Q (36,19): μ1 = 0.5 | μ2 = 0.4 | μ3 = 023 | Saída da RNAP = 0.34

Coordenada R (37,17): μ1 = 0.5 | μ2 = 0.3 | μ3 = 020 | Saída da RNAP = 0.3

Coordenada S (38,15): μ1 = 0.5 | μ2 = 0.2 | μ3 = 018 | Saída da RNAP = 0.265

Coordenada T (39,13): μ1 = 0.5 | μ2 = 0.1 | μ3 = 015 | Saída da RNAP = 0.225

Coordenada U (39,11): μ1 = 0.5 | μ2 = 1.38777878078e-016 | μ3 = 0.13 | Saída da RNAP = 0.19

A figura 5.13 mostra a localização das células analisadas pelo teste 2.

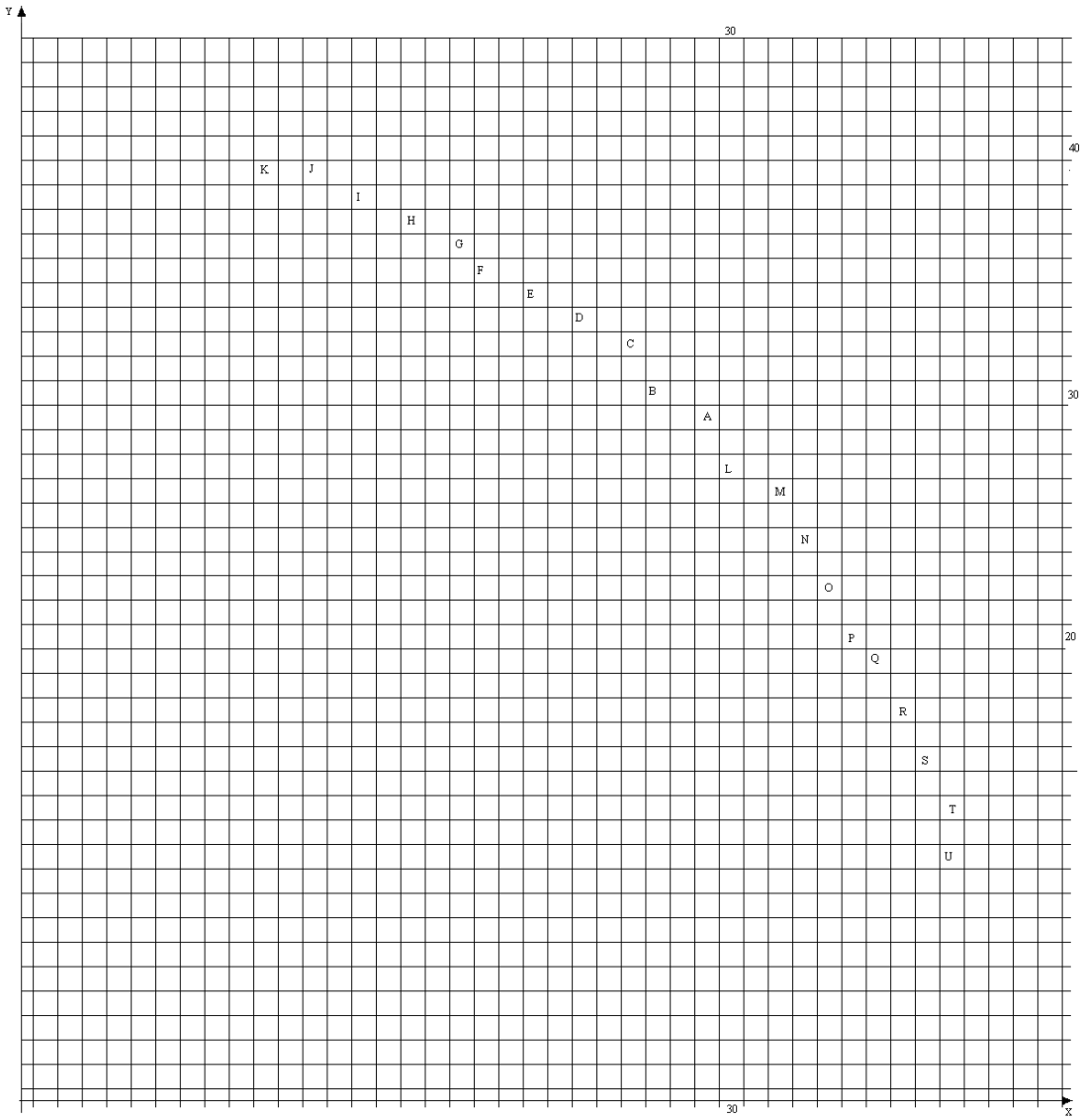


Figura 5.13 Resultado do teste 2 realizado com o subsistema de sensoriamento

Como a configuração e os dados de entrada do teste 2 são os mesmos do teste 1, as respostas dos dois testes se referem às mesmas células. Assim, a figura 5.13 é igual a figura 5.12. Apenas os valores da resposta de cada célula são diferentes porque o valor de μ3 do

teste 2 é igual ao valor de saída do teste 1.

Teste 3

O teste 3 possui a mesma configuração dos testes anteriores. Mas as informações de entrada são diferentes.

Dados do Sensor:

- Distância entre o obstáculo e o sensor: 200
- Ângulo do sensor em relação ao eixo x: 30
- Coordenada x atual (onde o robô se encontra): 0
- Coordenada y atual (onde o robô se encontra): 0

Resposta do sistema:

- Coordenada Central

Coordenada A (18,10): μ1 = 0.75 | μ2 = 1 | μ3 = 0 |Saída da RNAP = 0.438

- Coordenadas da Esquerda

Coordenada B (17,11): μ1 = 0.75 | μ2 = 0.9 | μ3 = 0| Saída da RNAP = 0.413

Coordenada C (17,12): μ1 = 0.75 | μ2 = 0.8 | μ3 = 0| Saída da RNAP = 0.388

Coordenada D (16,13): μ1 = 0.75 | μ2 = 0.7 | μ3 = 0| Saída da RNAP = 0.363

Coordenada E (15,14): μ1 = 0.75 | μ2 = 0.6 | μ3 = 0| Saída da RNAP = 0.338

Coordenada F (15,15): μ1 = 0.75 | μ2 = 0.5 | μ3 = 0| Saída da RNAP = 0.313

Coordenada G (14,15): μ1 = 0.75 | μ2 = 0.4 | μ3 = 0| Saída da RNAP = 0.288

Coordenada H (13,16): μ1 = 0.75 | μ2 = 0.3 | μ3 = 0| Saída da RNAP = 0.263

Coordenada I (12,17): μ1 = 0.75 | μ2 = 0.2 | μ3 = 0| Saída da RNAP = 0.238

Coordenada J (11,17): μ1 = 0.75 | μ2 = 0.1 | μ3 = 0| Saída da RNAP = 0.213

Coordenada K (10,18): μ1 = 0.75 | μ2 = 1.387778780781e-016 | μ3 = 0 | Saída da RNAP = 0.188

- Coordenadas da Direita

Coordenada L (18,10): μ1 = 0.75 | μ2 = 0.9 | μ3 = 0| Saída da RNAP = 0.413

Coordenada M (19,9): μ1 = 0.75 | μ2 = 0.8 | μ3 = 0| Saída da RNAP = 0.388

Coordenada N (19,8): μ1 = 0.75 | μ2 = 0.7 | μ3 = 0| Saída da RNAP = 0.363

Coordenada O (20,7): μ1 = 0.75 | μ2 = 0.6 | μ3 = 0| Saída da RNAP = 0.338

Coordenada P (20,6): μ1 = 0.75 | μ2 = 0.5 | μ3 = 0| Saída da RNAP = 0.313

Coordenada Q (20,5): μ1 = 0.75 | μ2 = 0.4 | μ3 = 0| Saída da RNAP = 0.288

Coordenada R (20,4): μ1 = 0.75 | μ2 = 0.3 | μ3 = 0| Saída da RNAP = 0.263

Coordenada S (20,3): μ1 = 0.75 | μ2 = 0.2 | μ3 = 0| Saída da RNAP = 0.238

Coordenada T (20,2): μ1 = 0.75 | μ2 = 0.1 | μ3 = 0| Saída da RNAP = 0.213

Coordenada U (20,0): μ1 = 0.75 | μ2 = 1.38777878078e-016 | μ3 = 0 | Saída da RNAP = 0.188

A figura 5.14 mostra graficamente o resultado do teste 3 realizado com o Subsistema de Sensoriamento.

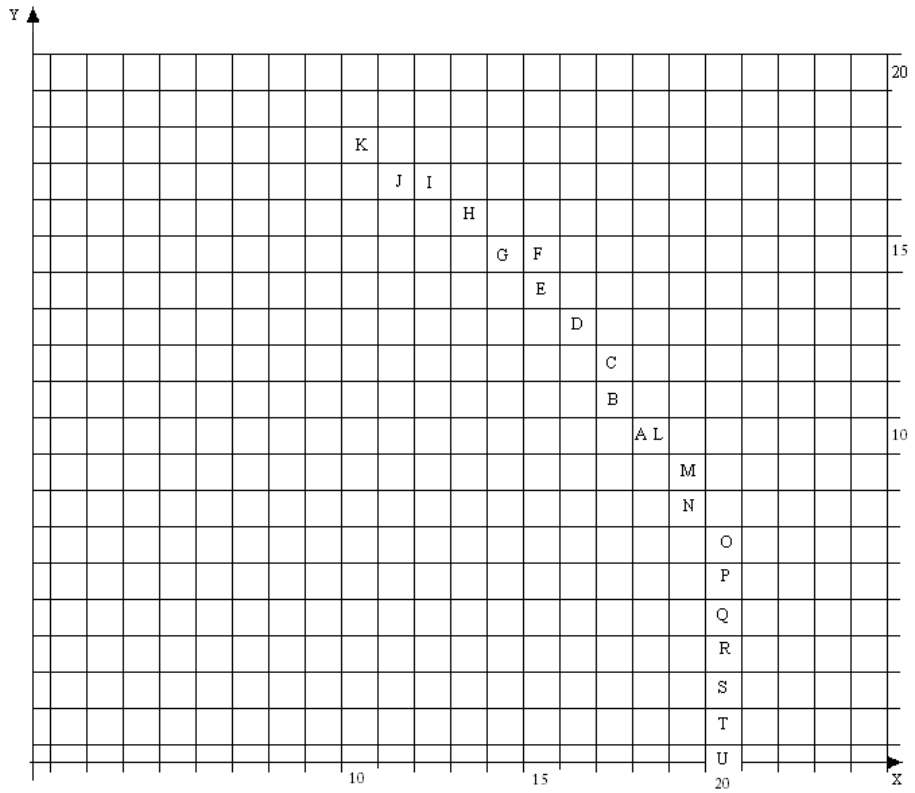


Figura 5.14 Resultado do teste 3 realizado com o subsistema de sensoriamento

Os resultados dos testes realizados mostram que o Subsistema de Sensoriamento proposto funciona adequadamente. Pois, o desenho formado pelas posições das células afetadas pelo Subsistema de Sensoriamento lembra o formato do “cone” gerado pelo sensor de ultrassom. E os valores dos graus de evidência das células decrescem à medida que a célula se posiciona nas extremidades do “cone”.

CAPÍTULO 6 - SUBSISTEMA DE PLANEJAMENTO

6.1 Considerações iniciais

Um robô móvel autônomo dotado de um sistema de navegação deve ser capaz de executar, basicamente, três tarefas. A primeira é capturar informações sobre o ambiente onde se encontra para saber onde existem obstáculos. A segunda tarefa é gerar uma rota que interligue a posição onde ele se encontra a uma posição destino. Essa rota deve desviar dos possíveis obstáculos existentes no ambiente. E a terceira tarefa é executar a rota previamente determinada de tal forma que a estrutura física do robô alcance a posição destino. O sistema de navegação e controle proposto neste trabalho divide as tarefas descritas em três subsistemas: Subsistema de Sensoriamento, Subsistema de Planejamento e Subsistema Mecânico.

No capítulo anterior foi apresentado o Subsistema de Sensoriamento. Ele tem o objetivo de captar informações sobre o ambiente em torno do robô e informar as posições onde se localizam os obstáculos. Esse subsistema gera o Grau de Evidência Favorável da proposição “existe obstáculo na célula” relativo a cada célula que compõe o ambiente em torno do robô. Esses valores são armazenados num banco de dados.

6.2 Subsistema de Planejamento proposto

Neste capítulo descreve-se o Subsistema de Planejamento. O seu objetivo é determinar uma sequência de ações para que o robô saia da posição onde se encontra e, num ambiente não estruturado, encontre a posição destino. O Subsistema de Planejamento proposto baseia-se na Lógica Paraconsistente Anotada Evidencial Eτ. Existem outros sistemas de

planejamento para robôs móveis baseado em outras teorias, como por exemplo, a encontrada em (BARRETO, ARAÚJO & ROSA, 1997), (DU, CHEN & GU, 2005), (KUBOTA, HISAJIMA, KOJIMA & FUKUDA, 2003), (MILLS, WALKER & HIMEBAUGH, 2003), (PRASSLER, RITTER, SCHAEFFER & FIORINI, 2004).

A figura 6.1 mostra a representação do Subsistema de Planejamento.

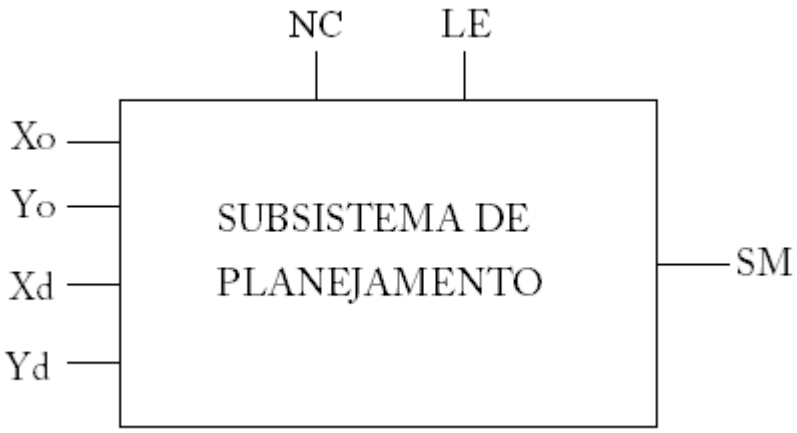


Figura 6.1 Representação do subsistema de planejamento

No Subsistema de Planejamento existem dois parâmetros de configuração, NC e LE. O Parâmetro NC indica o número de células que o sistema deve analisar antes de definir quais ações o robô deve tomar, ou seja, a quantidade de células que compõem a entrada das redes neurais.

Já o parâmetro LE indica o valor limite do Grau de Evidência Favorável para que o sistema considere a célula ocupada. Assim, as células que tiverem um Grau de Evidência Favorável maior que LE, serão consideradas ocupadas.

As coordenadas das células de origem e destino são as entradas do sistema. Assim, Xo e Yo são as coordenadas de origem e X_D e Y_D são as coordenadas de destino.

Como saída, o Subsistema de Planejamento gera uma sequência de movimentos para que a plataforma mecânica a execute. Essa sequência está representada por SM na figura 6.1

Apresenta-se no Anexo 2 o algoritmo completo do Subsistema de Planejamento e no

Anexo 3 o programa em linguagem Python do Subsistema de Planejamento.

O Subsistema de Planejamento, assim como o Subsistema de Sensoriamento, considera o ambiente onde o robô se encontra dividido em células, como mostrado na figura 6.2.

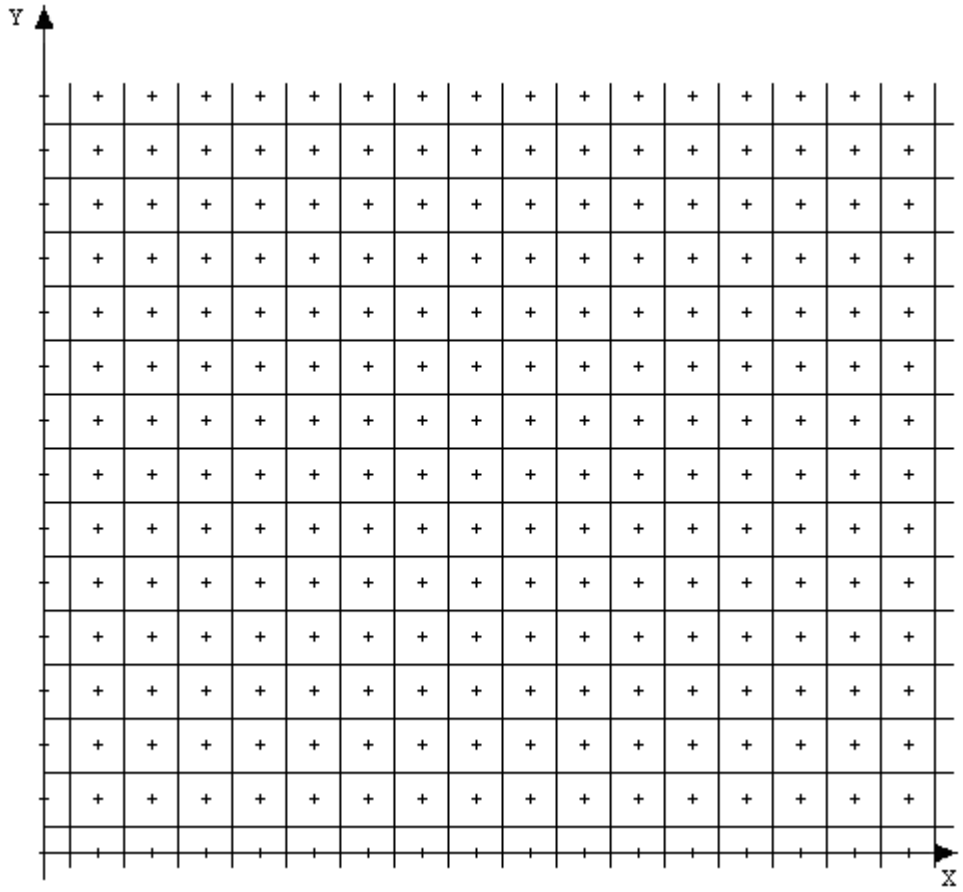


Figura 6.2 Representação do ambiente em torno do robô

O Subsistema de Planejamento deve gerar uma sequência de ações para que a plataforma mecânica deixe a posição onde se encontra e encontre uma posição destino.

O Subsistema Planejamento proposto pode gerar oito tipos diferentes de movimentos. A cada movimento se relaciona um número. Assim, o Subsistema de Planejamento deve informar ao Subsistema Mecânico uma sequência numérica. O Subsistema Mecânico deve interpretar cada número como um movimento diferente. A figura 6.3 mostra a representação do movimento 1.

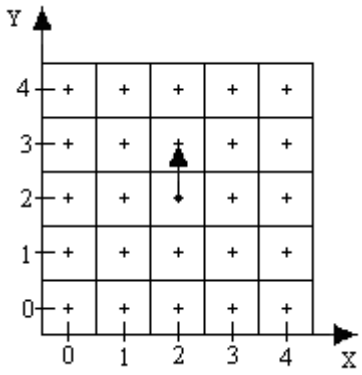


Figura 6.3 Representação do movimento 1

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.3 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 1 da seguinte forma: $X_D = X_O$ e $Y_D = Y_O + 1$.

A figura 6.4 mostra a representação do movimento 2.

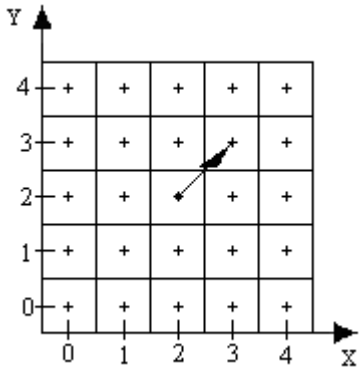


Figura 6.4 Representação do movimento 2

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.4 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 2 da seguinte forma: $X_D = X_O + 1$ e $Y_D = Y_O + 1$.

A figura 6.5 mostra a representação do movimento 3.

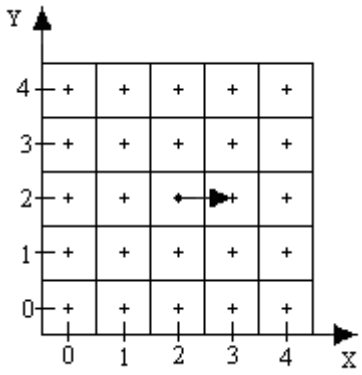


Figura 6.5 Representação do movimento 3

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.5 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 3 da seguinte forma: $X_D = X_O + 1$ e $Y_D = Y_O$.

A figura 6.6 mostra a representação do movimento 4.

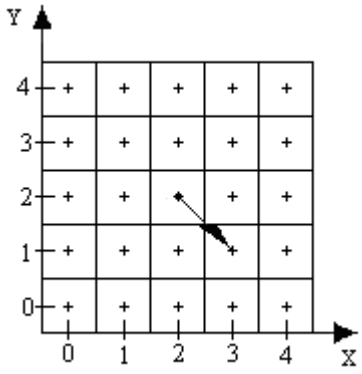


Figura 6.6 Representação do movimento 4

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.6 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 4 da seguinte forma: $X_D = X_O + 1$ e $Y_D = Y_O - 1$.

A figura 6.7 mostra a representação do movimento 5.

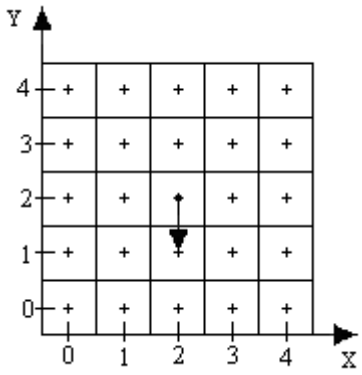


Figura 6.7 Representação do movimento 5

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.7 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 5 da seguinte forma: $X_D = X_O$ e $Y_D = Y_O - 1$.

A figura 6.8 mostra a representação do movimento 6.

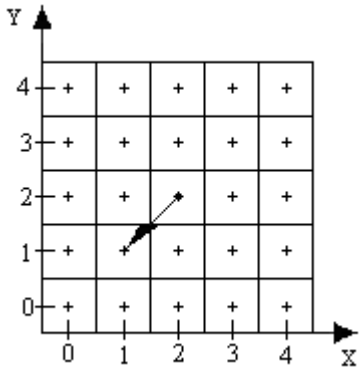


Figura 6.8 Representação do movimento 6

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.8 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 6 da seguinte forma: $X_D = X_O - 1$ e $Y_D = Y_O - 1$.

A figura 6.9 mostra a representação do movimento 7.

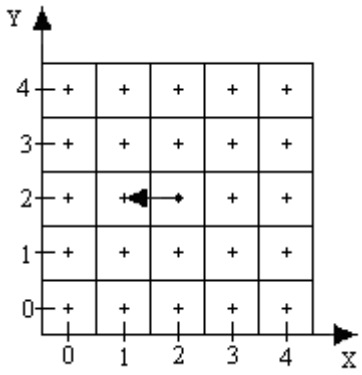


Figura 6.9 Representação do movimento 7

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.9 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento. Pode-se determinar a posição do robô após a execução do movimento 7 da seguinte forma: $X_D = X_O - 1$ e $Y_D = Y_O$.

A figura 6.10 mostra a representação do movimento 8.

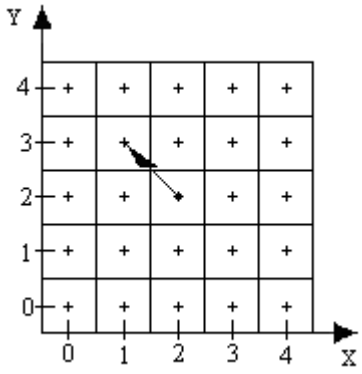


Figura 6.10 Representação do movimento 8

Considerando a coordenada onde o robô se encontra antes da execução do movimento como sendo (X_O, Y_O), ou seja, a coordenada (2, 2) na figura 6.10 e (X_D, Y_D) como sendo a coordenada onde o robô estará após a execução do movimento.Pode-se determinar a posição do robô após a execução do movimento 8 da seguinte forma: $X_D = X_O - 1$ e $Y_D = Y_O + 1$.

A figura 6.11 mostra um exemplo onde o robô localizado na célula (2, 2) deve se

locomover até a célula (14, 5).

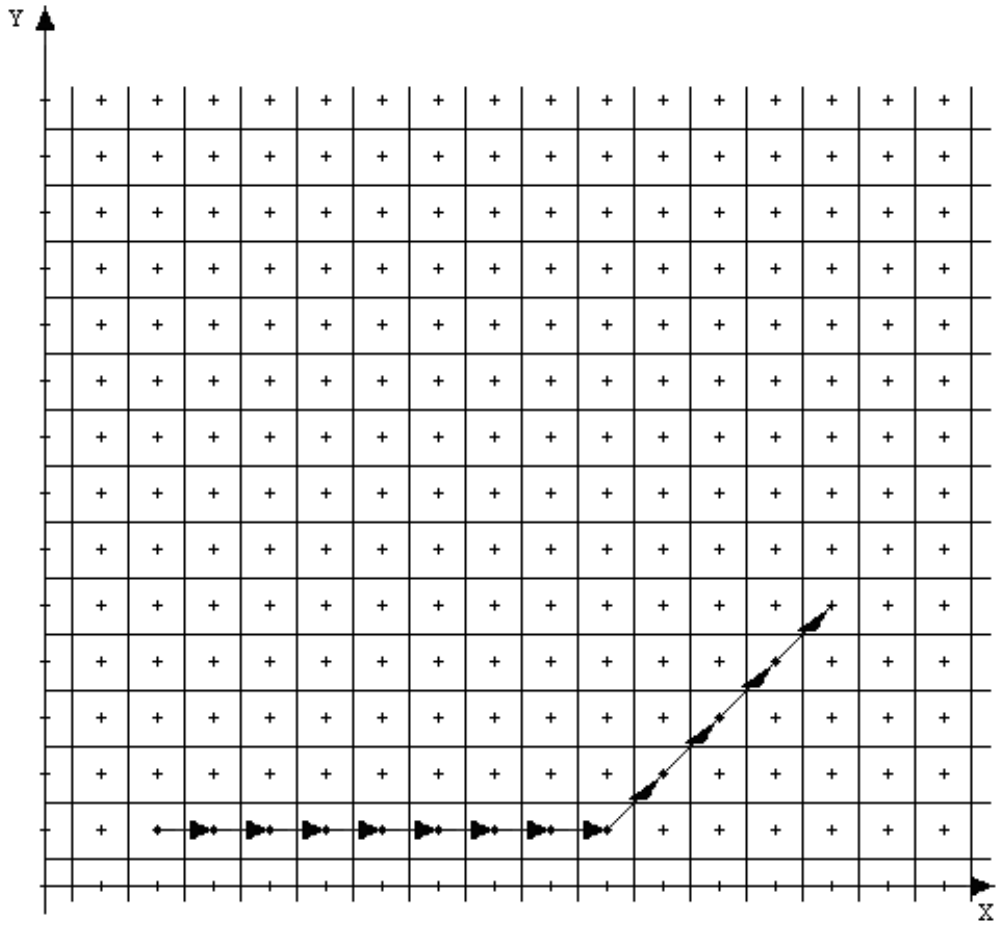


Figura 6.11 Representação da sequência de movimentos geradas pelo Sistema de Planejamento para que robô saia da célula (2, 2) e encontre a célula (14, 5)

Observando a figura 6.11 percebe-se que o Subsistema de Planejamento gerou a seguinte sequência numérica: (3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2) para que a plataforma mecânica deixe a posição origem e encontre a posição destino. O Subsistema Mecânico deve ser capaz de transformar essa sequência numérica em sua respectiva sequência de movimentos.

6.3 A Rede de Petri como base do subsistema de planejamento proposto

O Subsistema de Planejamento proposto é fruto da modelagem do ambiente em torno do robô e dos seus possíveis movimentos com as Redes de Petri (DE MORAES & CASTRUCCI, 2001). Assim, define-se agora alguns conceitos das Redes de Petri antes de se

continuar a descrição do Subsistema de Planejamento.

Segundo (DE MORAES & CASTRUCCI, 2001) define-se formalmente uma Rede de Petri como uma quintupla (P, T, A, W, m0) em que:

- $P = \{p_1, ..., p_n\}$ é um conjunto finito de posições ou lugares.
- $T = \{t_1, ..., t_m\}$ é um conjunto finito de transições.
- A é um conjunto finito de arcos pertencente ao conjunto $(P \times T) \cup (T \times P)$, em que $(P \times T)$ representa o conjunto dos arcos orientados de p_i para t_j, também designados por (p_i, t_j), e $(T \times P)$ representa o conjunto dos arcos orientados de t_i, para p_j ou (t_i, p_j).
- W é a função que atribui um peso w (um número inteiro) a cada arco.
- m0 é um vetor cuja i-ésima coordenada define o número de marcas (tokens) na posição p_i, no início da evolução da rede.
- Os conjuntos T e P são disjuntos, isto é, $T \cap P = 0$.
- $N = |P|$ é a cardinalidade do conjunto P, o número de posições da Rede de Petri.
- $M = |T|$ é o número de transições da Rede de Petri.

Pode-se definir uma Rede de Petri de forma informal como um grafo bipartido orientado composto por três elementos: lugares ou posições, transições e arco orientado. Os lugares ou posições são conectados a outros lugares ou posições através de transições. A figura 4.12 mostra os símbolos representativos da Rede de Petri.

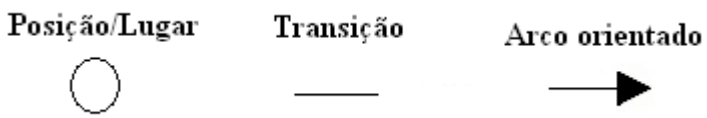


Figura 6.12 Símbolos representativos da Rede de Petri

Os lugares modelam condições e as transições modelam eventos.

A seguir encontram-se algumas outras definições importantes.

- Pré-set de transição: ${}^{\bullet}t = \{\forall p_i \in P \mid A \in (p_i, t)\}$, ou seja, o pré-set de t, ${}^{\bullet}t$, é o conjunto das posições em P a partir das quais existe arco para a transição t.
- Pós-set de transição: $t^{\bullet} = \{\forall p_i \in P \mid A \in (t, p_i)\}$, ou seja, o pós-set de t, $t^{\bullet}$, é o conjunto das posições em P para quais existe arco oriundo da transição t.
- Pré-set de posição: ${}^{\bullet}p = \{\forall t_j \in T \mid A \in (t_j, p)\}$, ou seja, o pré-set de p, ${}^{\bullet}p$, é o conjunto das transições T a partir das quais existe arco para a posição p.
- Pós-set de posição: $p^{\bullet} = \{\forall t_j \in T \mid A \in (p, t_j)\}$, ou seja, o pós-set de p, $p^{\bullet}$, é o conjunto das transições T para quais existe arco oriundo da posição p.

Chama-se de execução da Rede de Petri a movimentação das marcas (fichas) pela rede de acordo com certas regras; ocorre em duas fases: habilitação e disparo da transição.

Uma transição $t_j \in T$ numa Rede de Petri é habilitada por uma marcação m se ocorre que, para todo $p_i \in {}^{\bullet}t, m(p_i) \geq w(p_i, t_j)$, isto é, (marcação em p_i) $\geq$ (peso do arco de p_i a t_j).

Uma transição é disparada por meio de duas operações:

- Remover fichas das posições de pré-set (tantas fichas quanto for o peso do arco correspondente).
- Depositar em cada uma das posições do pós-set tantas fichas quanto for o peso do arco correspondente.

O vetor de marcação m é um vetor cuja i-ésima coordenada define o número de marcas (fichas) na posição p_i.

6.4 Modelagem do subsistema de planejamento

Consideram-se como posições da Rede de Petri as células que compõem o ambiente em volta do robô e as transições são os possíveis movimentos que o robô pode realizar.

A figura 6.13 mostra o grafo representativo da Rede de Petri que modela o ambiente em torno do robô e os movimentos passíveis de realização. A posição que contém marca representa a célula onde o robô se encontra.

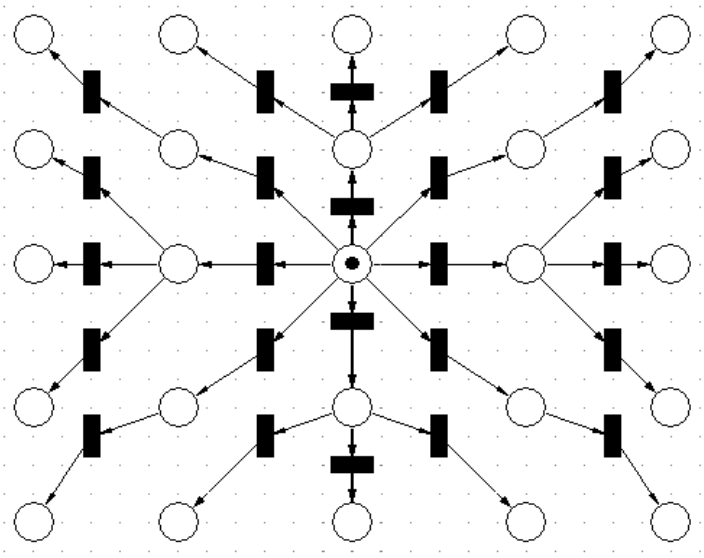


Figura 6.13 Grafo representativo da Rede de Petri que modela o subsistema de planejamento

Considerando que o vetor de marcação representa o estado da Rede de Petri, podemos mostrar através de um grafo os estados da Rede de Petri e os possíveis movimentos do robô. Na figura 6.14 os nós indicam os estados da Rede de Petri e os arcos os possíveis movimentos do robô.

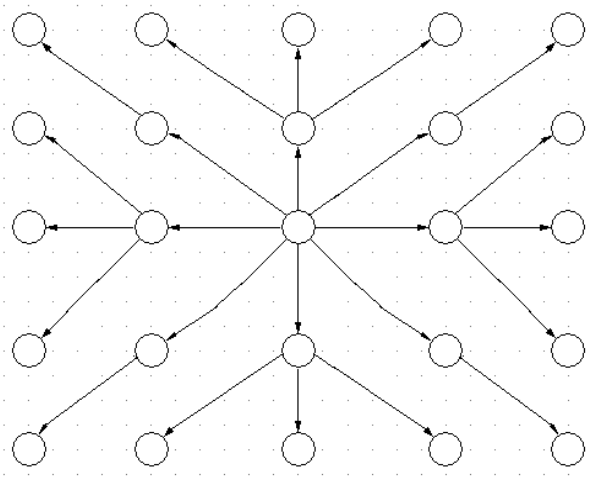


Figura 6.14 Grafo representativo dos estados da Rede de Petri que modela o planejador

Um planejador determina uma sequência de estados para que um sistema saia de um estado inicial e encontre um estado final. Observa-se na figura 6.14 que o planejador poderia ter que experimentar muitos caminhos até encontra um caminho que ligasse o estado inicial ao estado final desejado.

Mas, se simplesmente invertermos os sentidos dos arcos representados na figura 6.14, fazendo com que o planejador gere a sequência de ações para que o robô saia da posição final e encontre a posição inicial, teremos um caminho único. A figura 6.15 mostra um grafo representando o planejador que gera a sequência de ações para que o robô saia do estado destino e encontre o estado inicial.

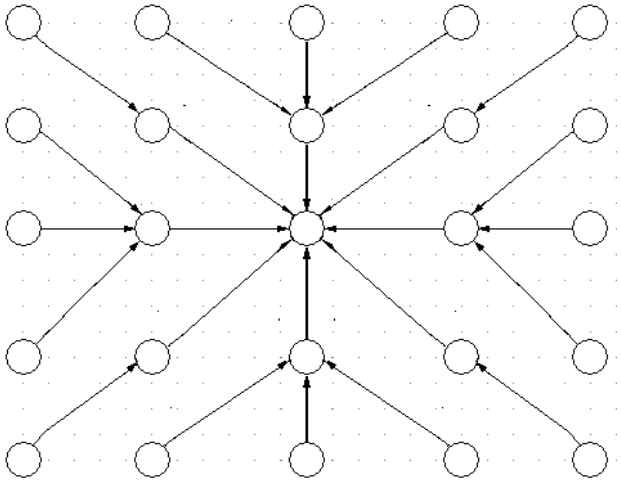
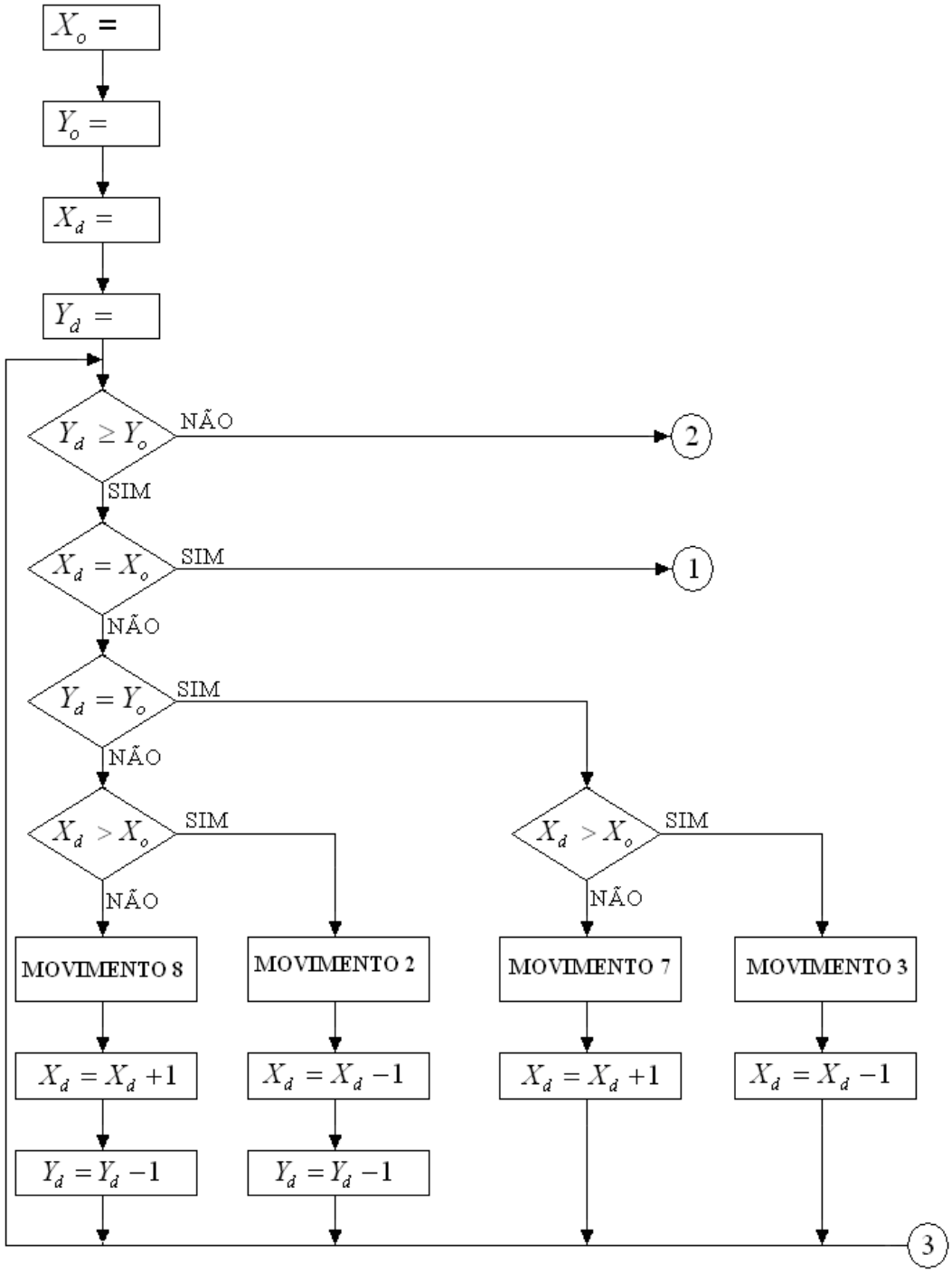


Figura 6.15 Grafo representativo do planejador que gera as ações para que o robô saia do destino e encontre a origem

Desta forma, se o robô sempre executar a última ação determinada pelo planejador e em seguida refazer o planejamento, ele será capaz de alcançar a célula destino.

6.5 Relação entre o Subsistema de planejamento proposto e as Redes de Petri

A modelagem com a Rede de Petri permitiu a construção do algoritmo planejador apresentado nesta tese. O algoritmo planejador mostrado na figura 6.16 é capaz de gerar a sequência de ações para que o robô deixe sua posição origem e encontre uma posição destino num ambiente sem obstáculo. Na figura 6.16, (X_o, Y_o) representa a coordenada onde se localiza a célula origem, ou seja, a célula onde o robô se encontra, e (X_d, Y_d) representa a coordenada da célula destino.


X_o =
Y_o =
X_d =
Y_d =
$Y_d \geq Y_o$
NÃO
2
SIM
$X_d = X_o$
SIM
1
NÃO
$Y_d = Y_o$
SIM
NÃO
$X_d > X_o$
SIM
NÃO
$X_d > X_o$
SIM
NÃO
MOVIMENTO 8
MOVIMENTO 2
MOVIMENTO 7
MOVIMENTO 3
$X_d = X_d + 1$
$X_d = X_d - 1$
$X_d = X_d + 1$
$X_d = X_d - 1$
$Y_d = Y_d - 1$
$Y_d = Y_d - 1$
3

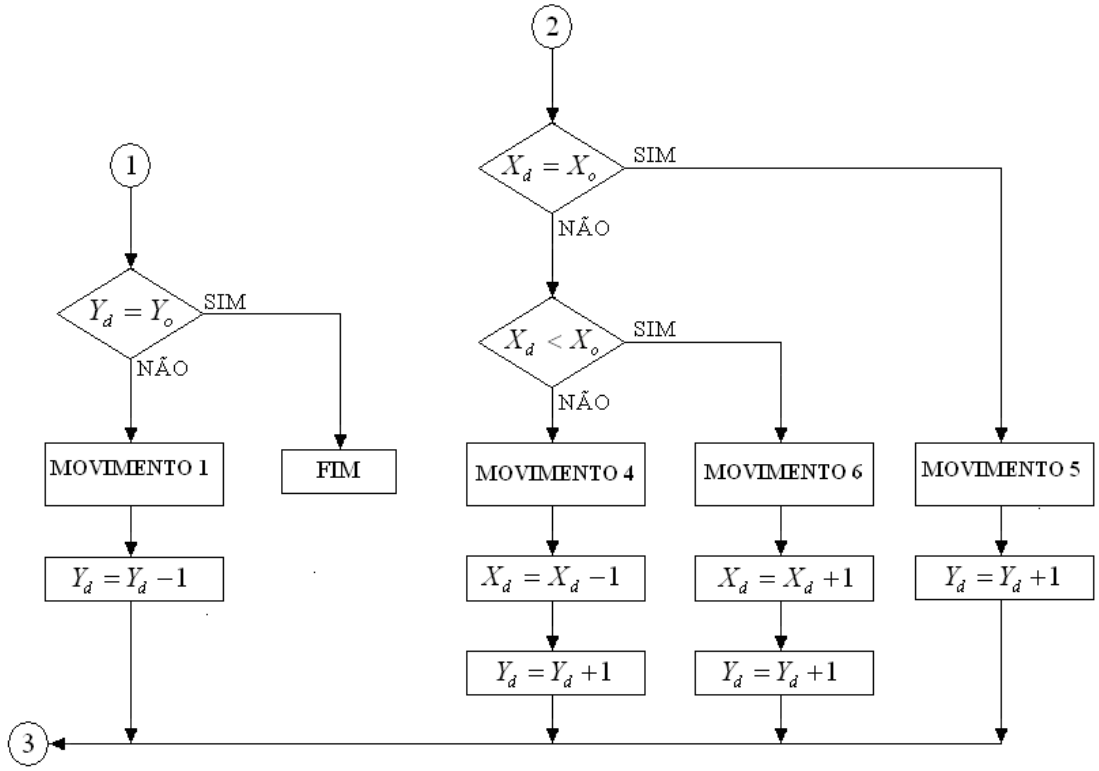


Figura 6.16 Algoritmo simplificado do subsistema de planejamento

Apresentam-se agora alguns resultados alcançados através do algoritmo mostrado na figura 6.16. Enfatiza-se que este algoritmo funciona apenas num ambiente sem obstáculo.

A figura 6.17 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (0, 8).

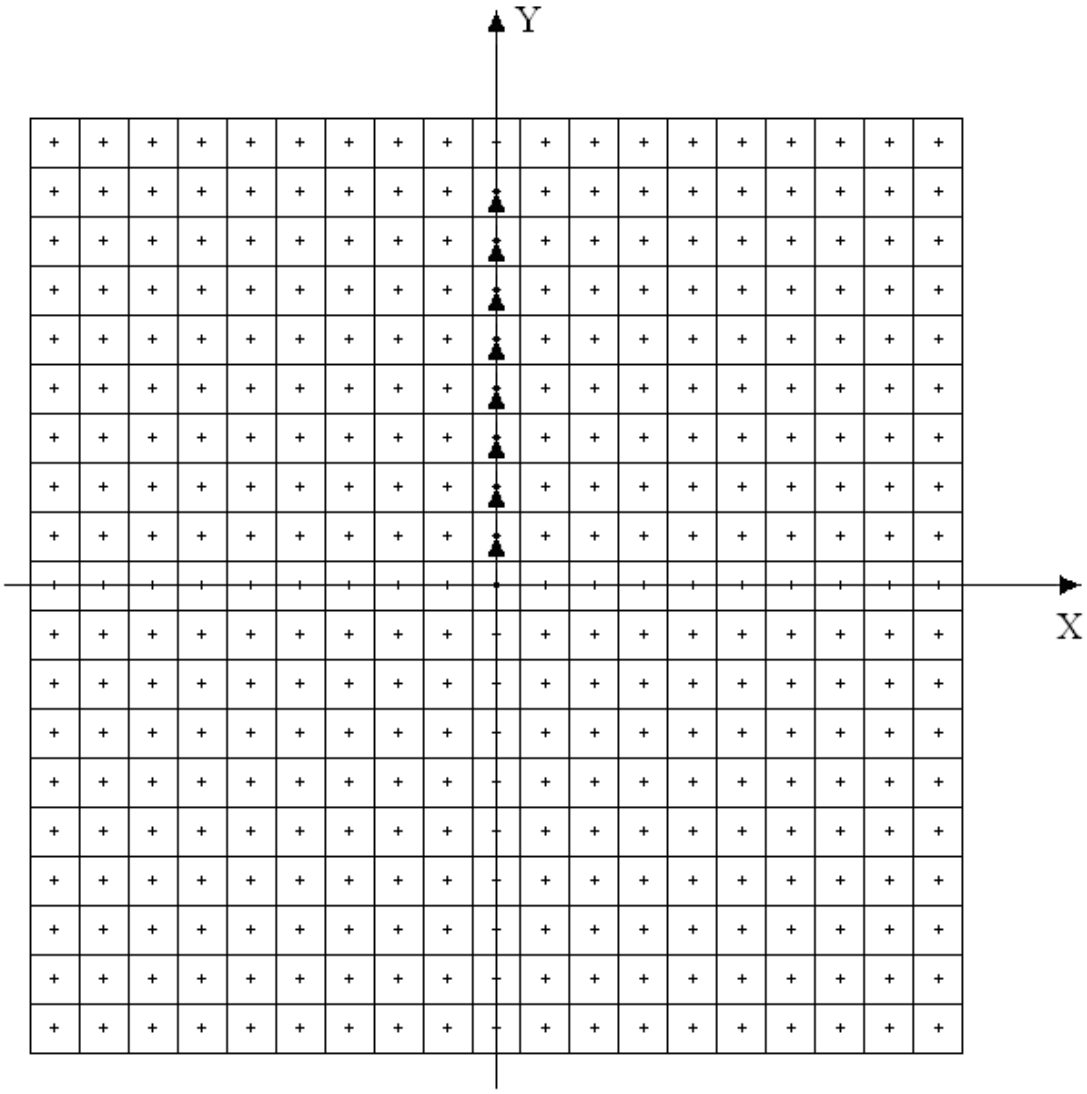


Figura 6.17 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (0, 8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (1, 1, 1, 1, 1, 1, 1, 1), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.18 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (8, 8).

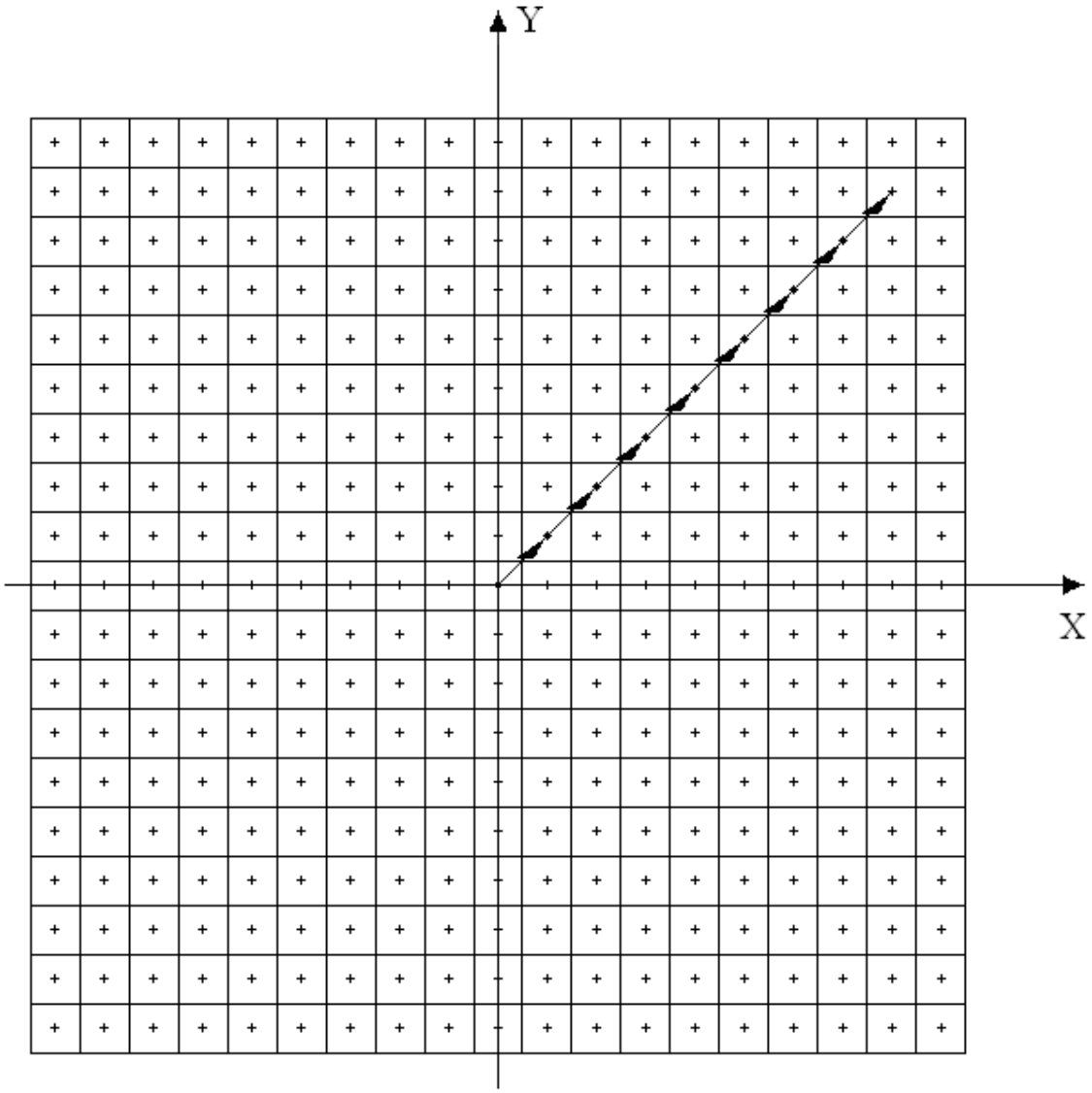


Figura 6.18 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (8, 8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (2, 2, 2, 2, 2, 2, 2, 2), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.19 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (8, 0).

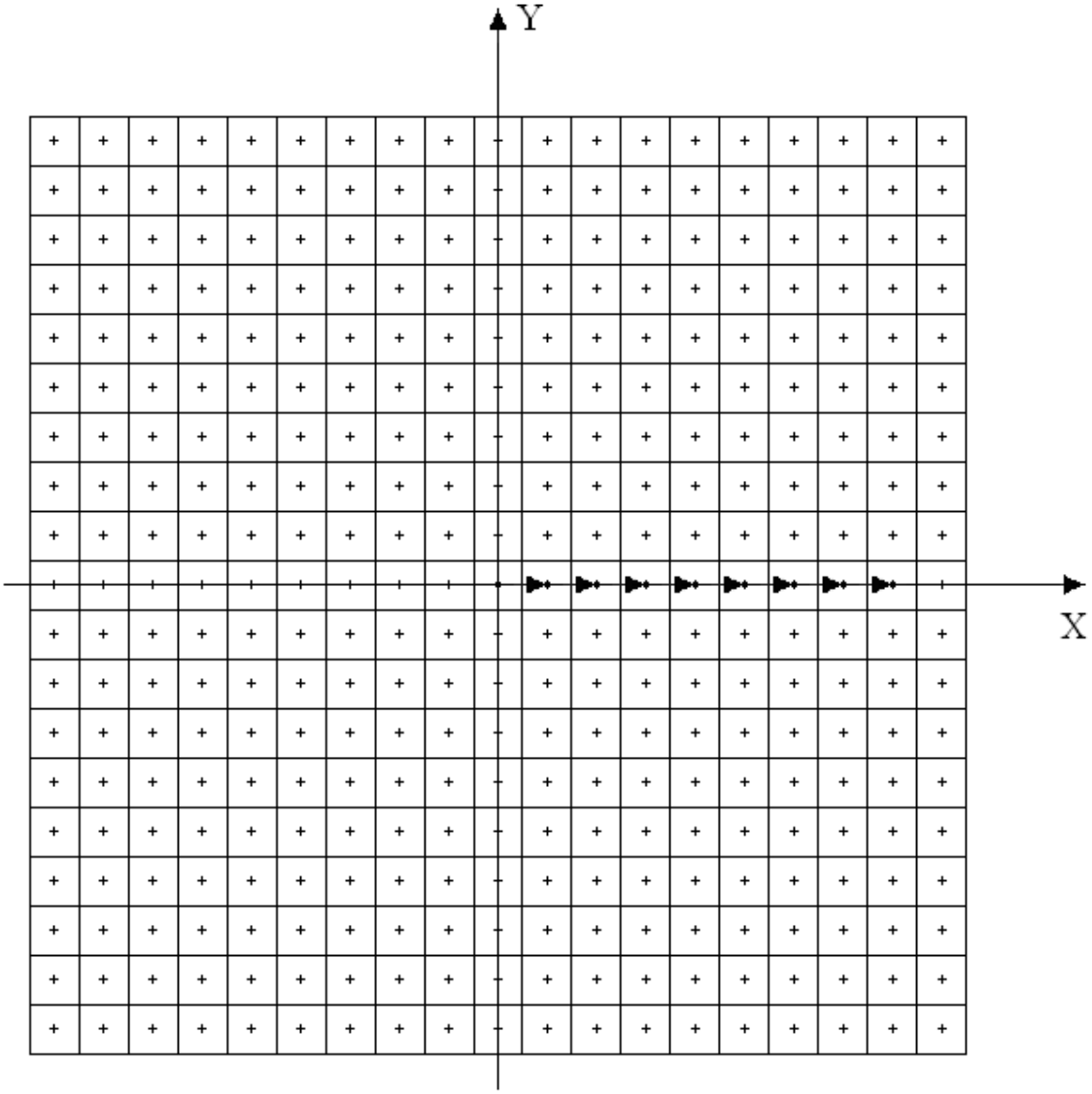


Figura 6.19 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (8, 0)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (3, 3, 3, 3, 3, 3, 3, 3), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.20 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (8, -8).

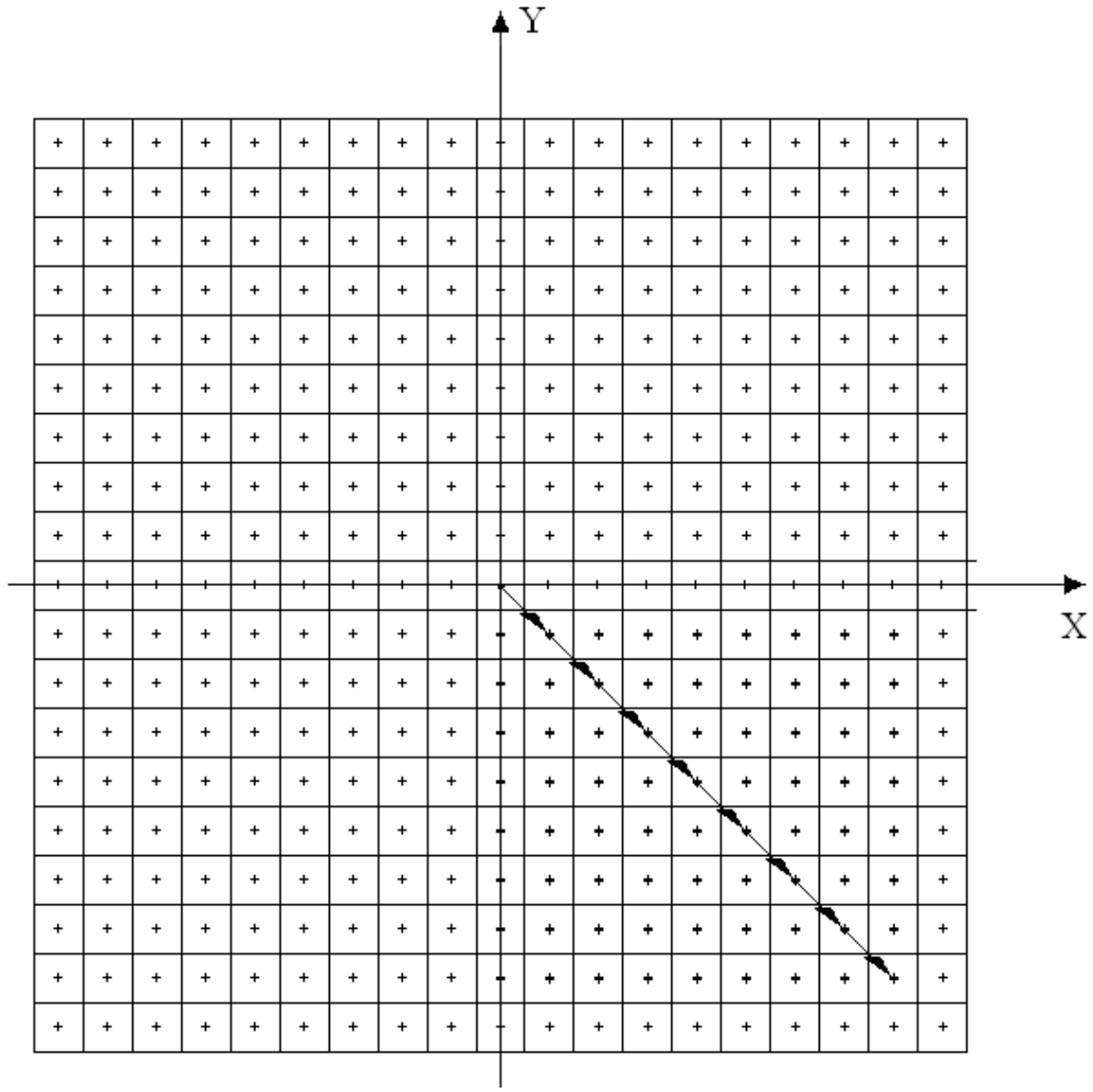


Figura 6.20 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (8, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (4, 4, 4, 4, 4, 4, 4, 4), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.21 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (0, -8).

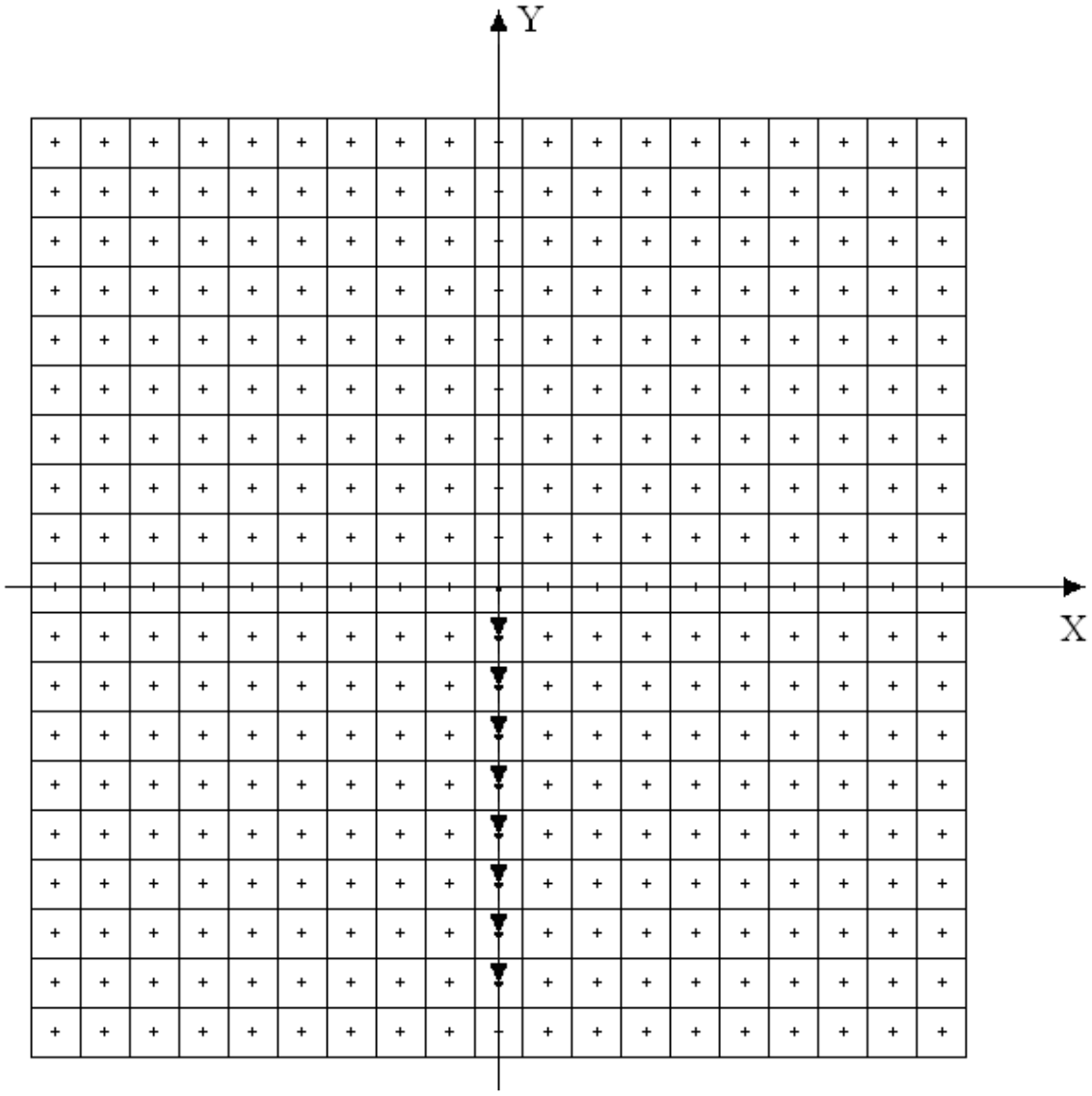


Figura 6.21 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (0, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (5, 5, 5, 5, 5, 5, 5, 5), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.22 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-8, -8).

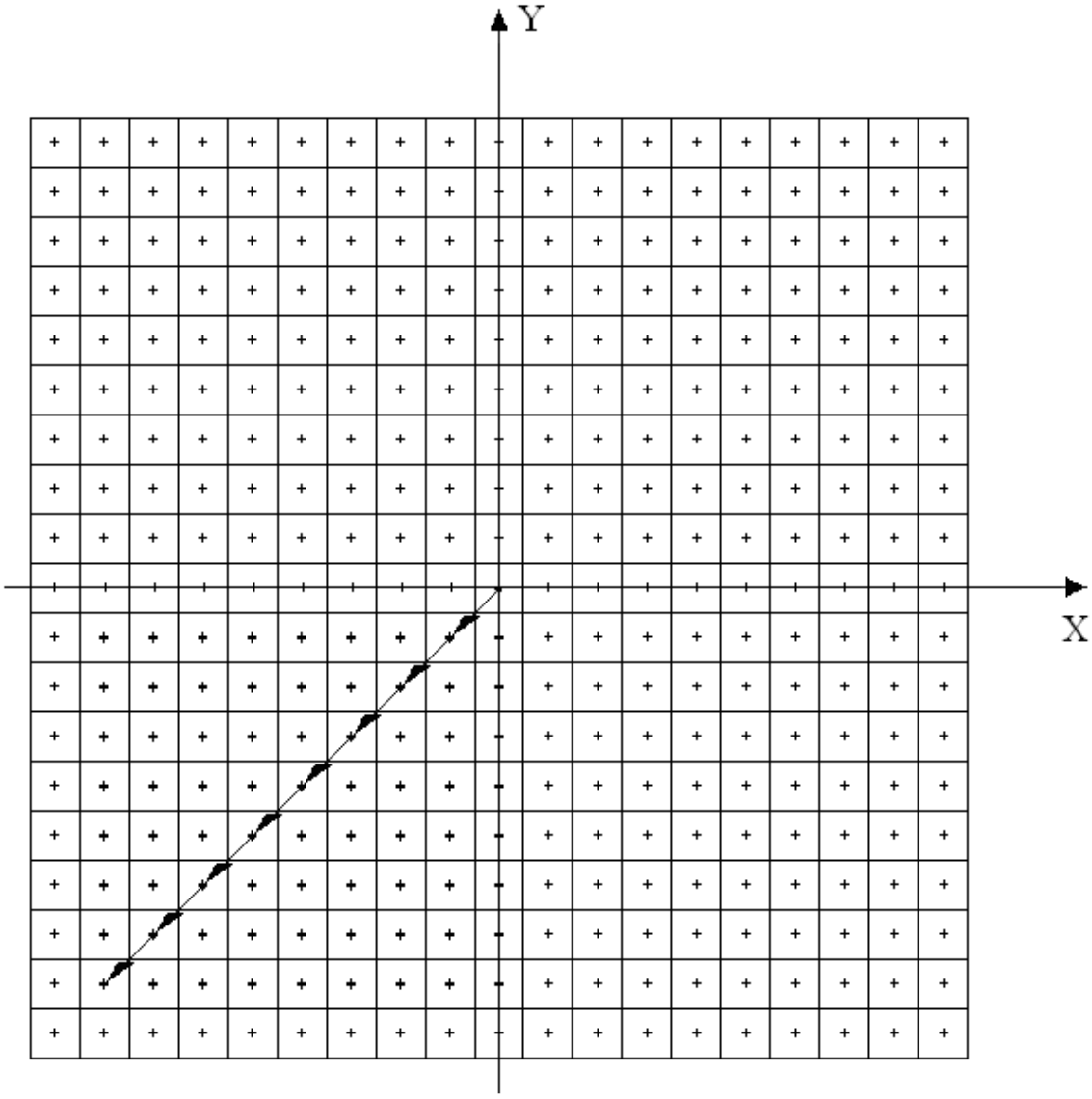


Figura 6.22 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-8, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (6, 6, 6, 6, 6, 6, 6, 6), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.23 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-8, 0).

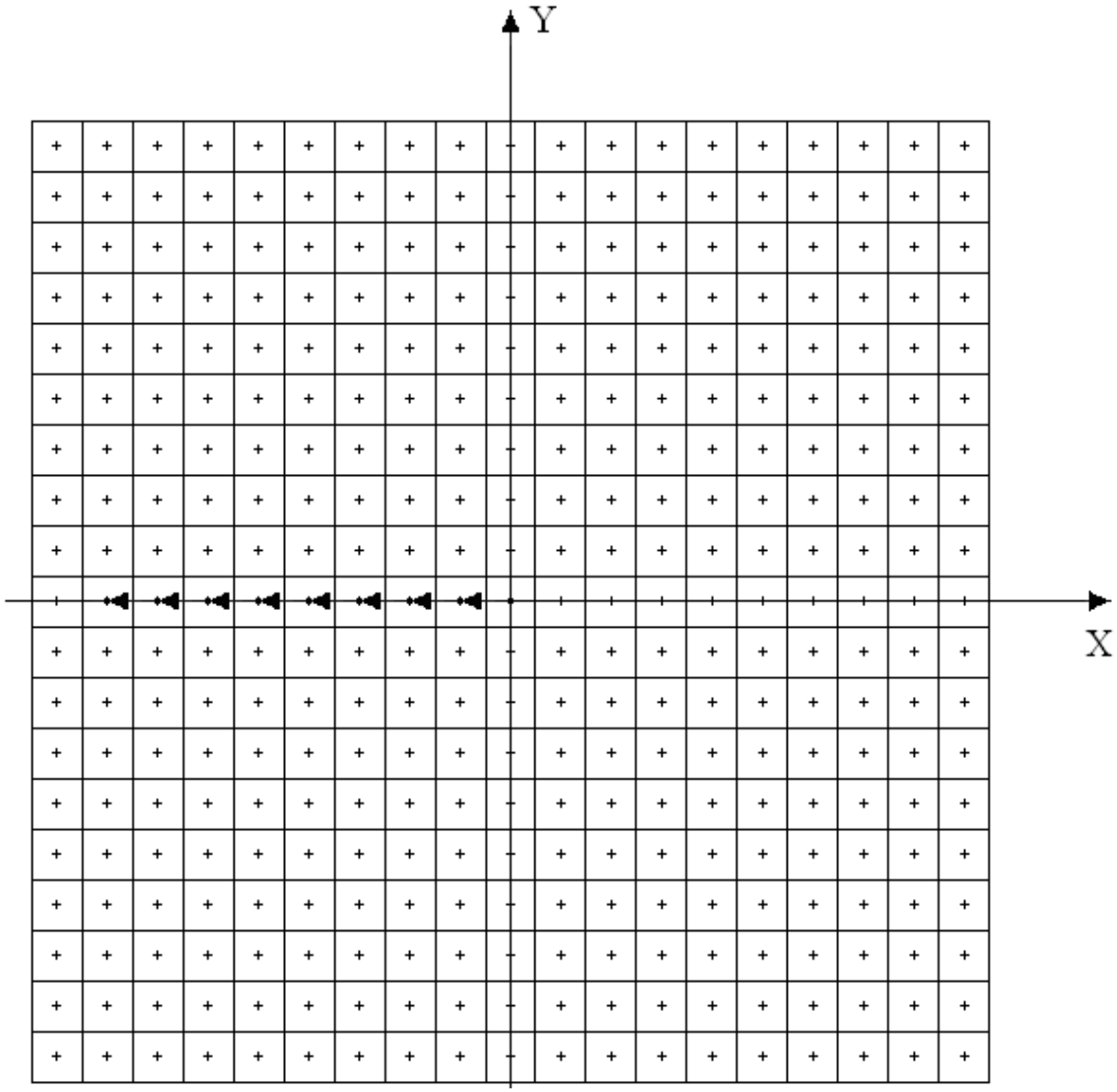


Figura 6.23 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-8, 0)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (7, 7, 7, 7, 7, 7, 7, 7), sendo o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.24 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-8, 8).

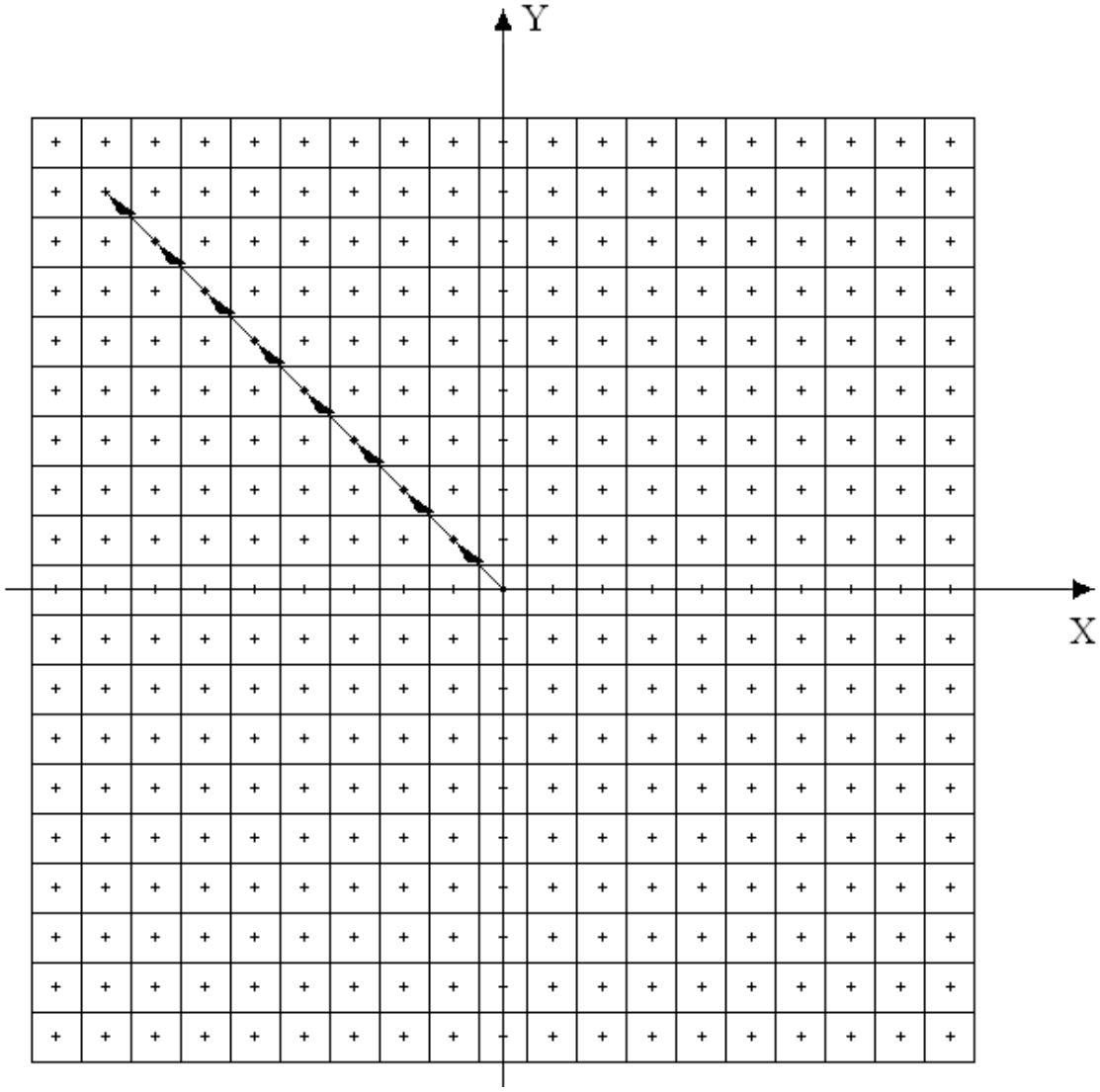


Figura 6.24 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-8, 8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (8, 8, 8, 8, 8, 8, 8, 8), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica.

A figura 6.25 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (4, 8).

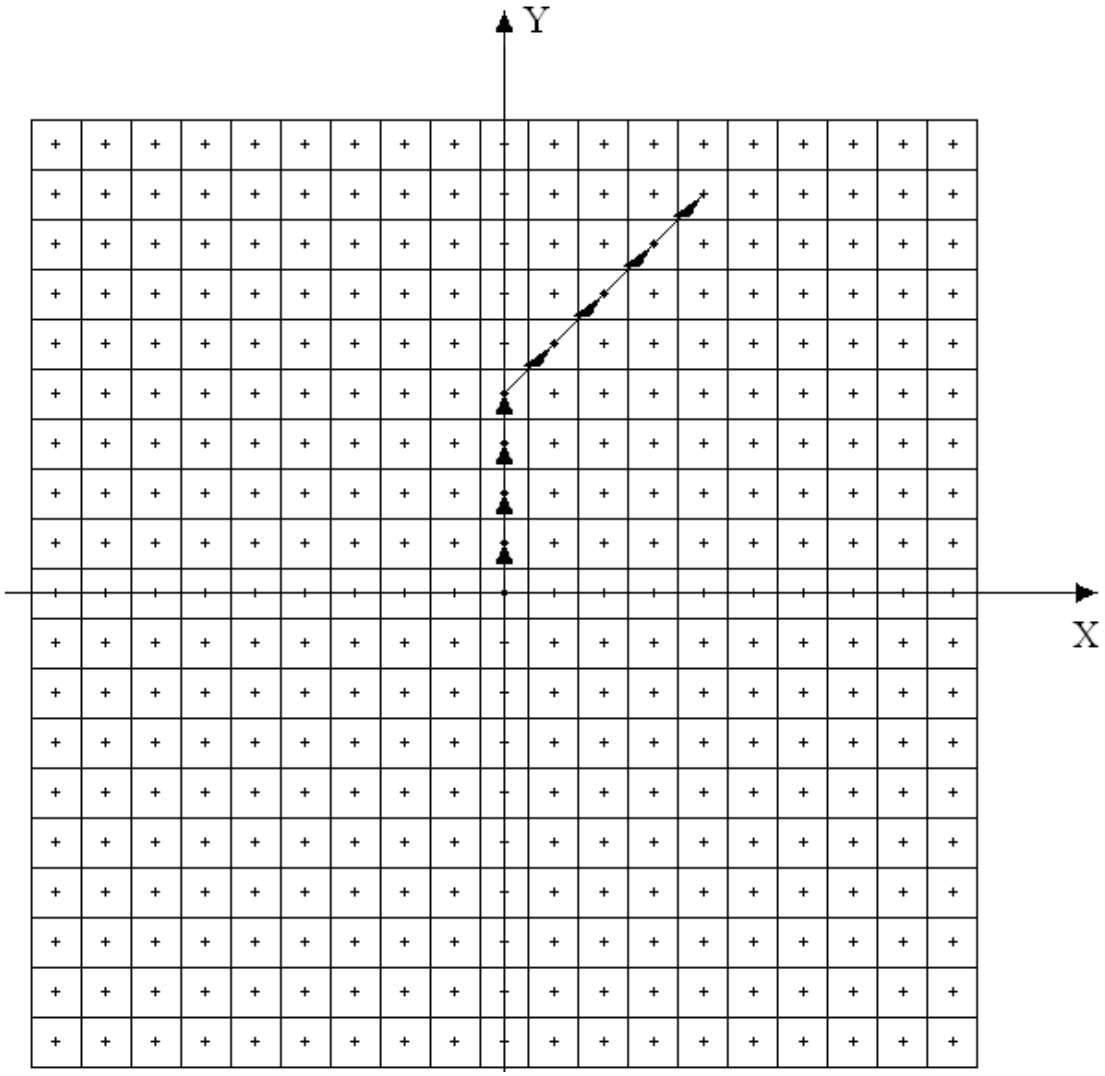


Figura 6.25 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (4, 8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (2, 2, 2, 2, 1, 1, 1, 1), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 1.

A figura 6.26 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (8, 4).

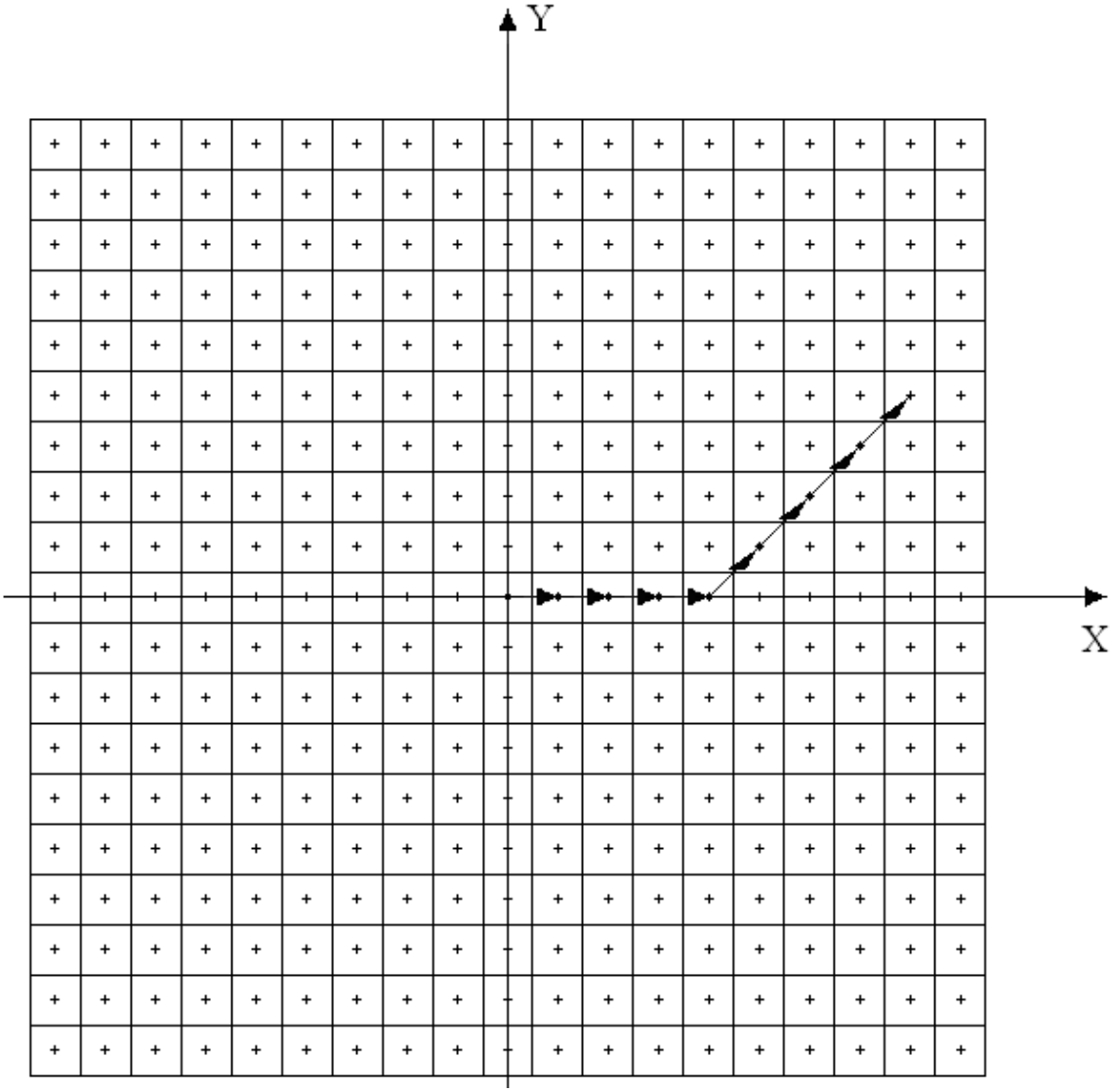


Figura 6.26 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (8, 4)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (2, 2, 2, 2, 3, 3, 3, 3), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 3.

A figura 6.27 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (8, -4).

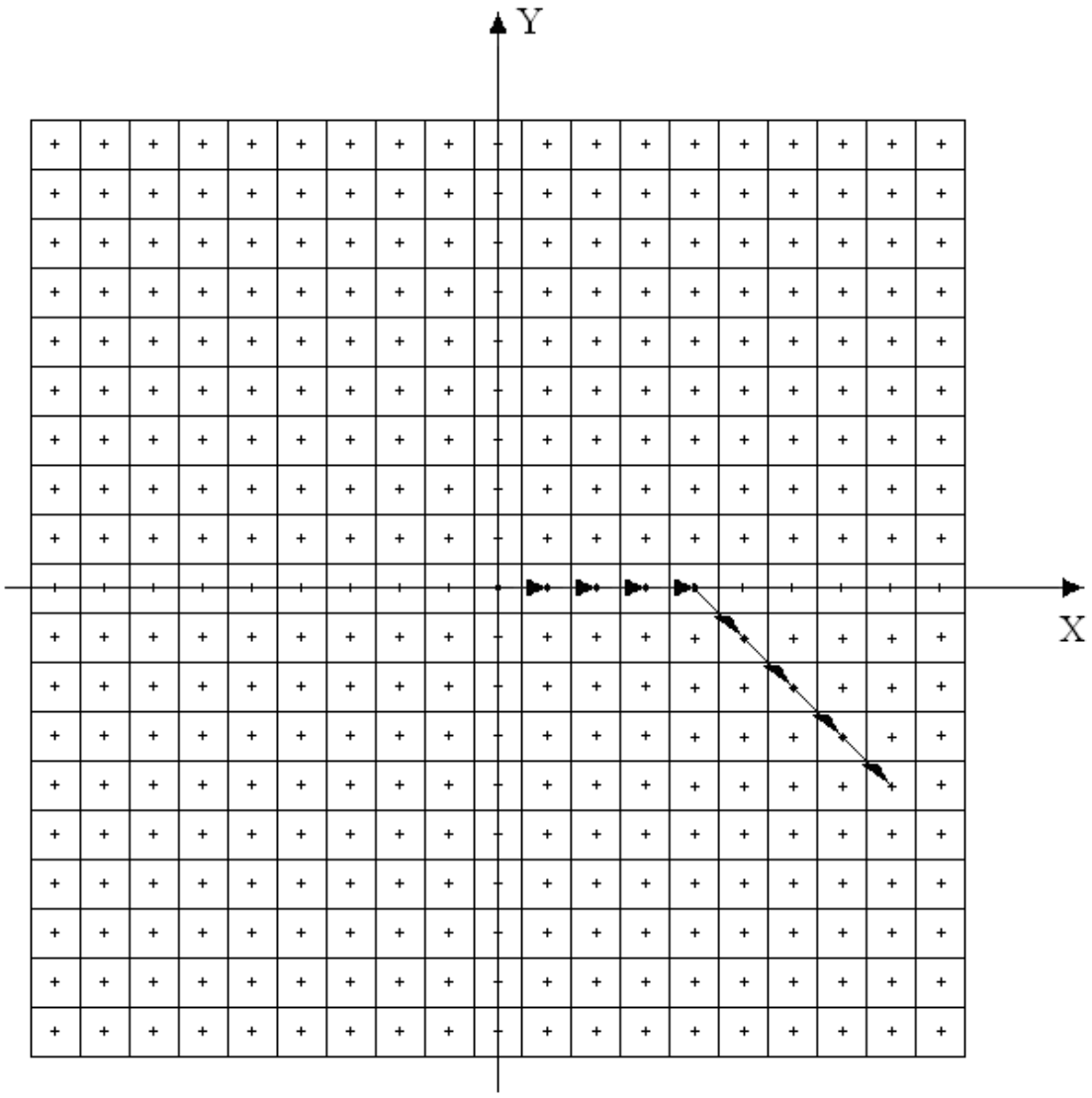


Figura 6.27 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (8, -4)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (4, 4, 4, 4, 3, 3, 3, 3), sendo o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 3.

A figura 6.28 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (4, -8).

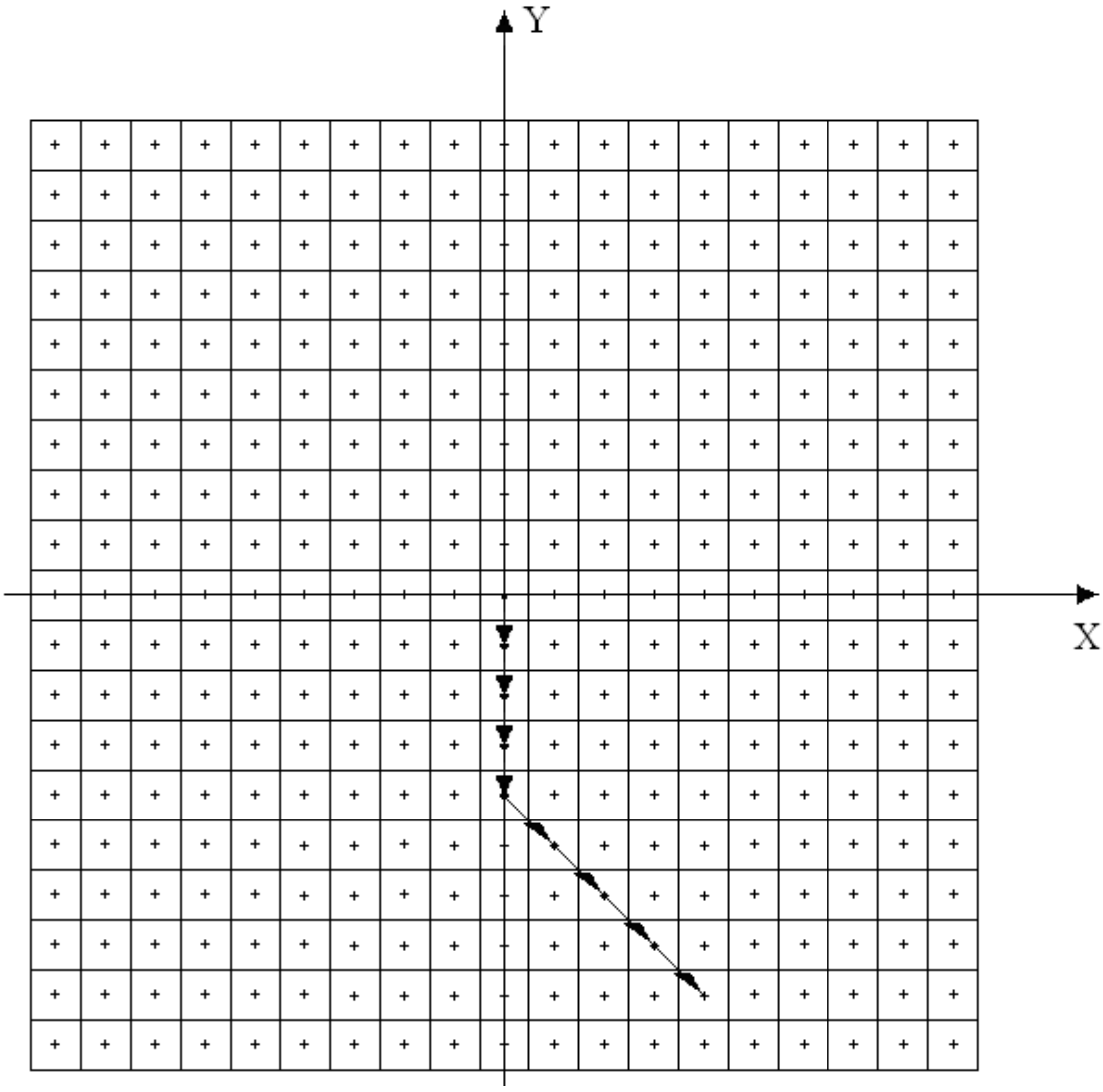


Figura 6.28 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (4, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (4, 4, 4, 4, 5, 5, 5, 5), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 5.

A figura 6.29 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-4, -8).

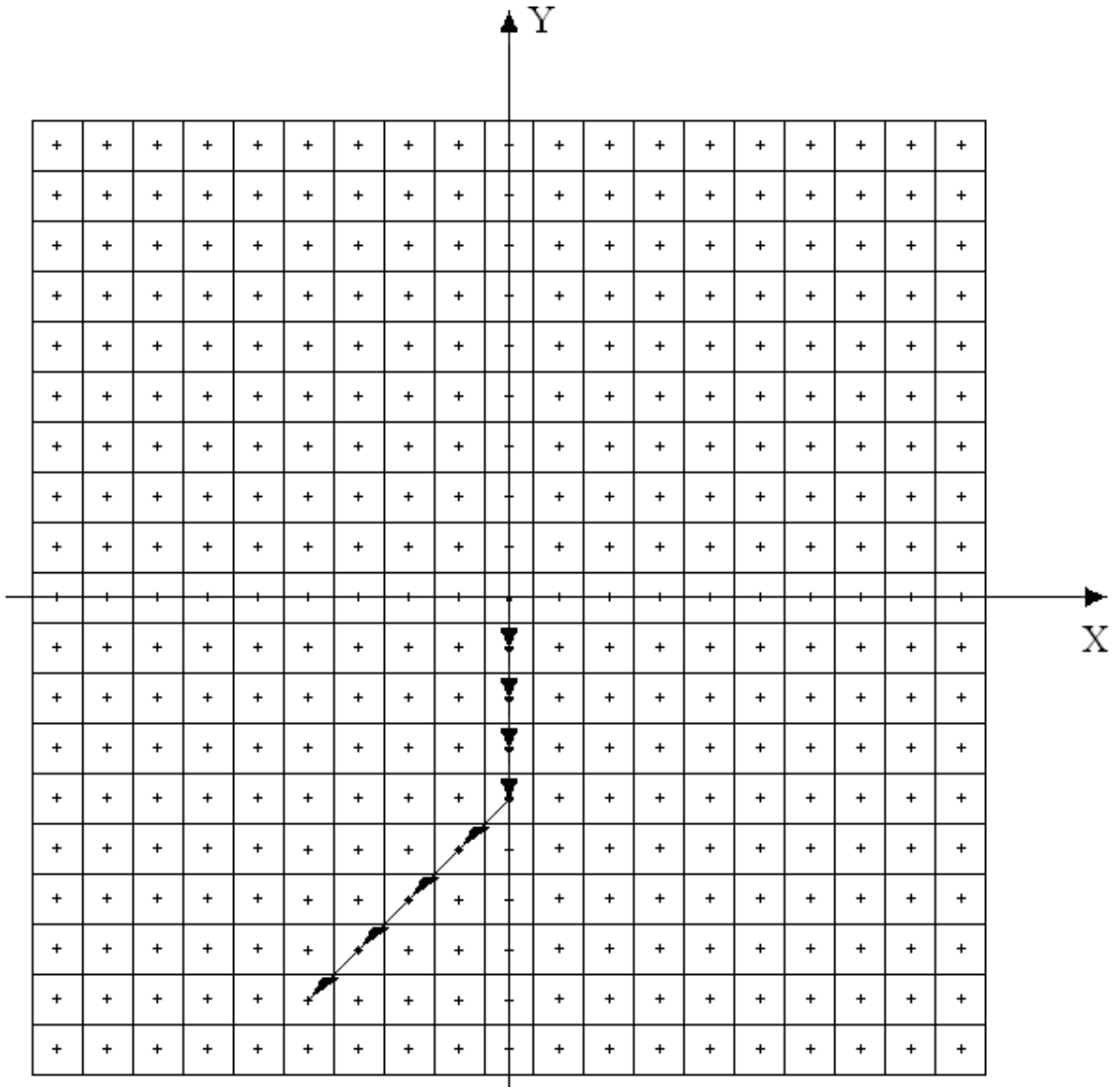


Figura 6.29 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-4, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (6, 6, 6, 6, 5, 5, 5, 5), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 5.

A figura 6.30 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-8, -4).

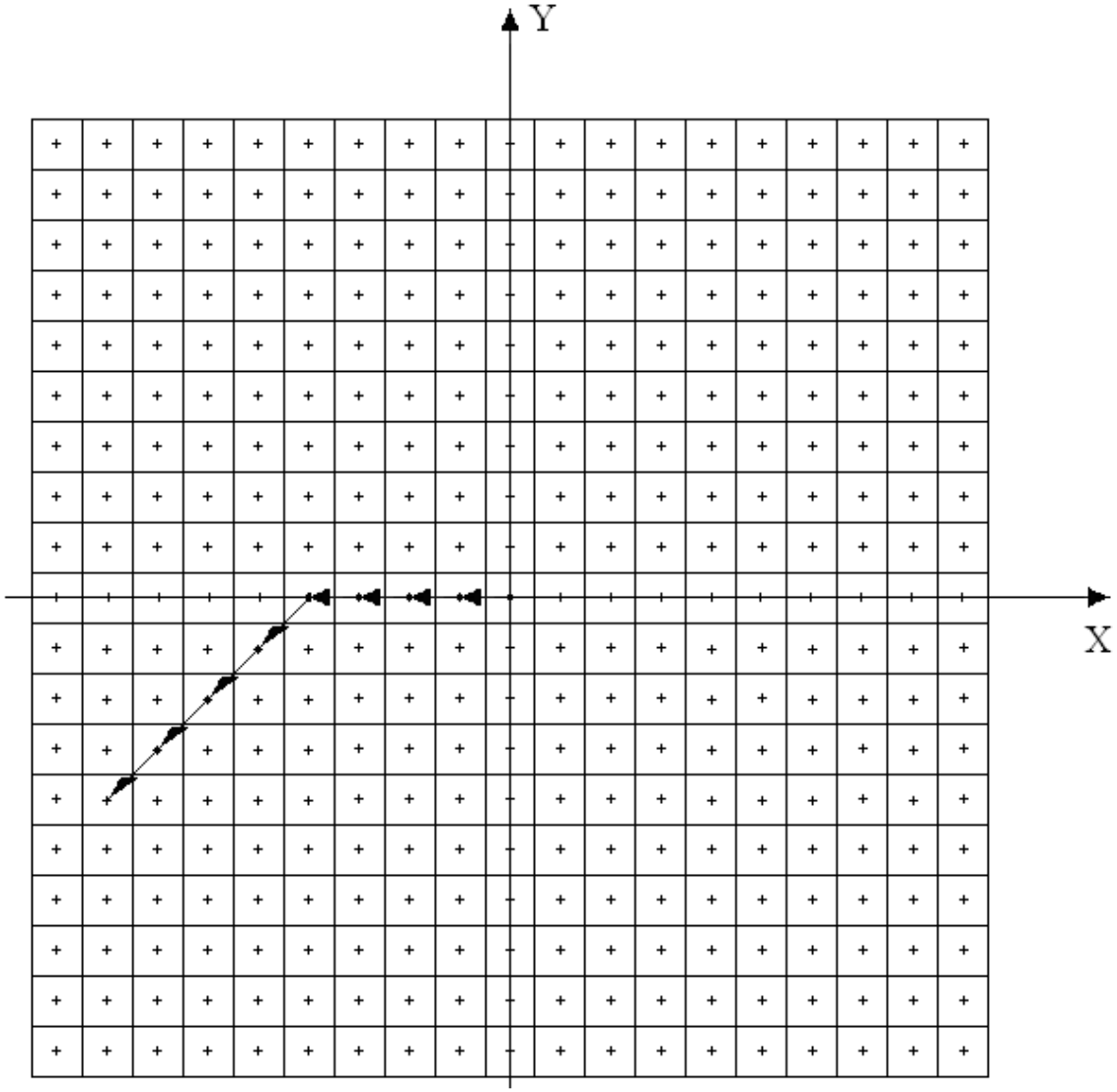


Figura 6.30 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-4, -8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (6, 6, 6, 6, 3, 3, 3, 3), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 3.

A figura 6.31 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-8, 4).

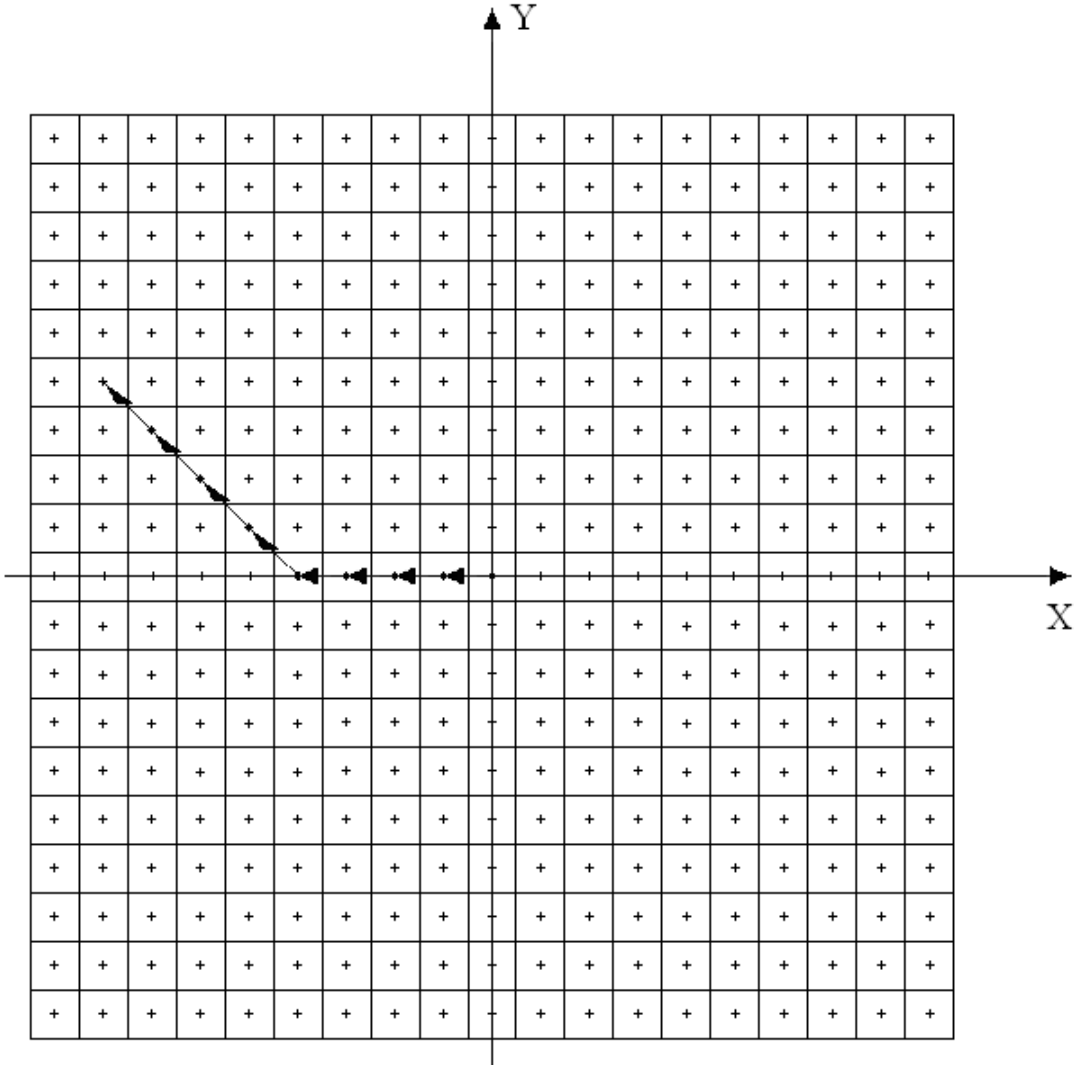


Figura 6.31 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-8, 4)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (8, 8, 8, 8, 3, 3, 3, 3), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 3.

A figura 6.32 mostra a representação dos movimentos quando o robô está na coordenada (0, 0) e deseja ir até a coordenada (-4, 8).

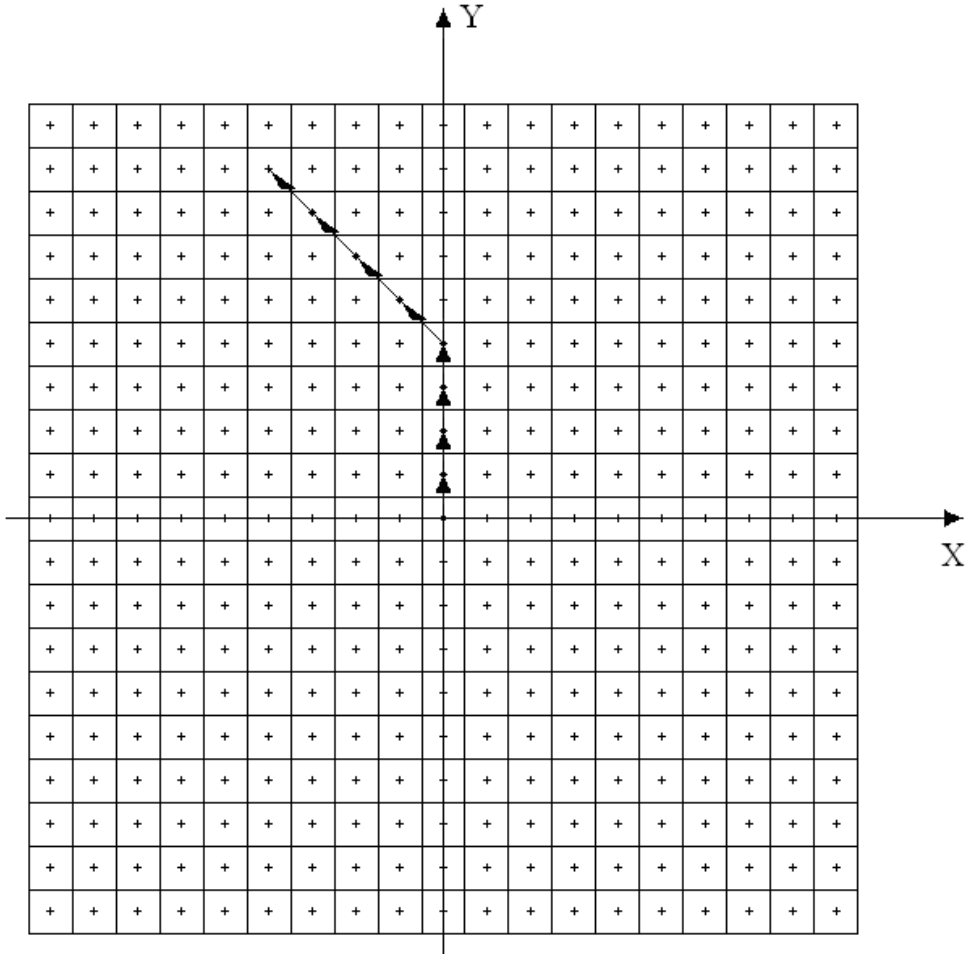


Figura 6.32 Representação dos movimentos quando a coordenada origem é (0, 0) e a coordenada destino é (-4, 8)

O planejador gera a sequência de ações para que se saia do destino e se alcance a origem. Assim, a sequência numérica gerada pelo planejador é (8, 8, 8, 8, 1, 1, 1, 1), sendo que o último movimento gerado pelo planejador deve ser o primeiro a ser executado pela plataforma mecânica. Portanto, o primeiro movimento executado pelo robô deve ser o movimento 1.

Pelo descrito até aqui o planejador é capaz de gerar a sequência de ações para que o robô deixe a sua posição de origem e alcance a posição destino num ambiente livre de obstáculos.

Descreve-se, a partir de agora, as ações determinadas pelo Subsistema de Planejamento quando o robô se encontra num ambiente com obstáculos.

O Grau de Evidência Favorável da proposição “existe obstáculo na célula” de cada célula que compõe o ambiente em torno do robô está armazenado no banco de dados do sistema. Antes do Sistema de Planejamento determinar quais ações o robô deve realizar, ele consulta no banco de dados as condições das células pelas quais o robô deve passar. As células que possuírem um valor de Grau de Evidência Favorável acima de um determinado valor são consideradas ocupadas.

O Subsistema de Planejamento determina qual o maior Grau de Evidência Favorável existente nas células analisadas. E caso esse valor seja maior que o valor do Grau de Evidência Favorável limite, o planejador escolhe outro movimento.

O Subsistema de Planejamento utiliza uma Rede Neural Artificial Paraconsistente compostas por Células Neurais Artificiais Paraconsistentes de Conexão Lógica Simples no Processo de Maximização (OU) para determinar qual o maior valor de Grau de Evidência Favorável das posições analisadas.

A figura 6.33 mostra a arquitetura da Rede Neural Artificial Paraconsistente utilizada pelo Subsistema de Planejamento para determinar qual o maior valor de Grau de Evidência Favorável existente nas células analisadas.

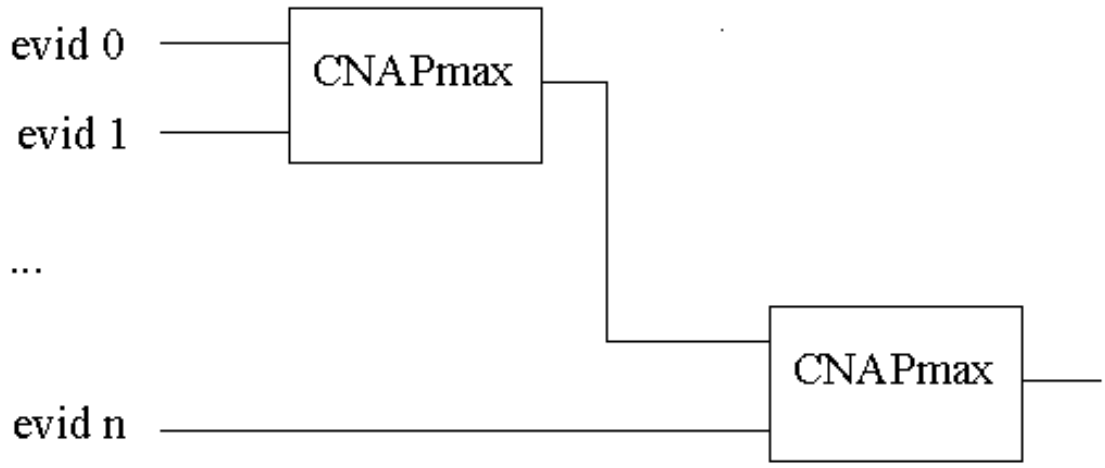


Figura 6.33 Rede neural artificial paraconsistente utilizada pelo subsistema de planejamento para determinar qual o maior valor de grau de evidência

As entradas da Rede Neural Artificial Paraconsistente mostradas na figura 4.33 são os Graus de Evidências Favoráveis das posições analisadas pelo Subsistema de Planejamento. E na saída encontra-se o maior valor de Grau de Evidência Favorável encontrado.

6.6 Subsistema de planejamento para um ambiente não estruturado

Para se descrever como o Subsistema de Planejamento[1] decide quando existe obstáculo no ambiente, utilizam-se vários exemplos, nos quais as células em branco têm Grau de Evidência Favorável igual a zero e as células negras têm Grau de Evidência Favorável igual a 1.

Analisando o exemplo 1 mostrado na figura 6.34, onde não há obstáculos, vê-se que a posição origem é a célula (0, 4) e a célula destino é a (8, 4). A sequência numérica determinada pelo Subsistema de Planejamento é (3, 3, 3, 3, 3, 3, 3, 3).

[1] Enfatiza-se que o planejador gera as ações do destino para a origem. Portanto, o último movimento determinado pelo planejador é o primeiro executado pelo robô.

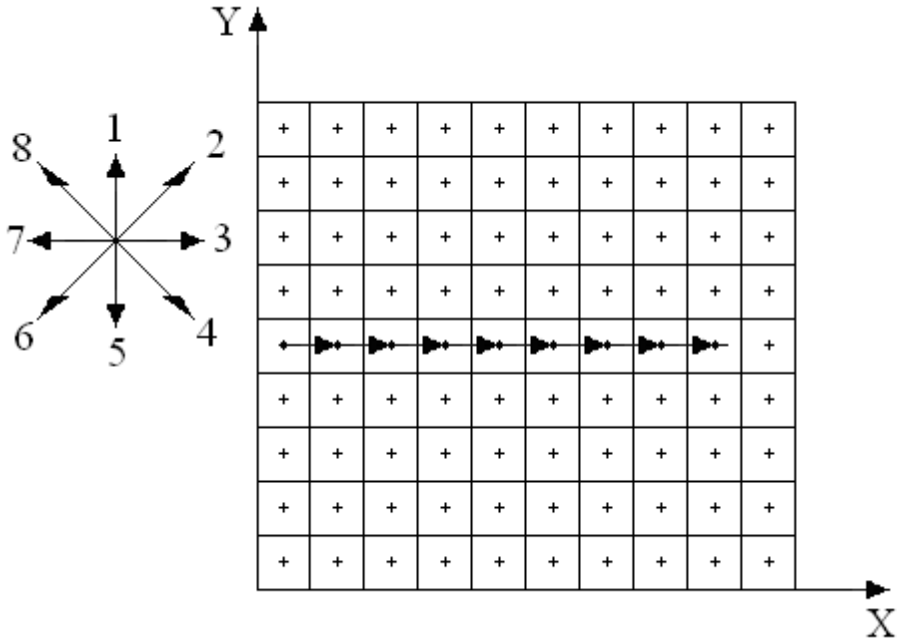


Figura 6.34 Exemplo 1: não existem obstáculos no caminho do robô

O Subsistema de Planejamento gera as ações do destino para a origem. Assim, antes de determinar o primeiro movimento 3, o Subsistema de Planejamento consulta no banco de dados as condições das células localizadas na direção do primeiro movimento. Portanto, as condições das células (7, 4), (6, 4) e (5, 4) seriam consultadas no banco de dados, caso o sistema estivesse configurado para analisar 3 células. Então, as condições das células (7, 4), (6, 4) e (5, 4) seriam a entrada da Rede Neural Artificial Paraconsistente mostrada na figura 6.33. Como no exemplo em análise não existem obstáculos, a saída da rede é 0,0, o que permite ao planejador escolher o movimento 3.

A figura 6.35 mostra o exemplo 2. Observa-se que na célula (5, 4) existe um obstáculo, ou seja, o Grau de Evidência Favorável desta célula é maior do que um valor de Evidência Favorável determinado.

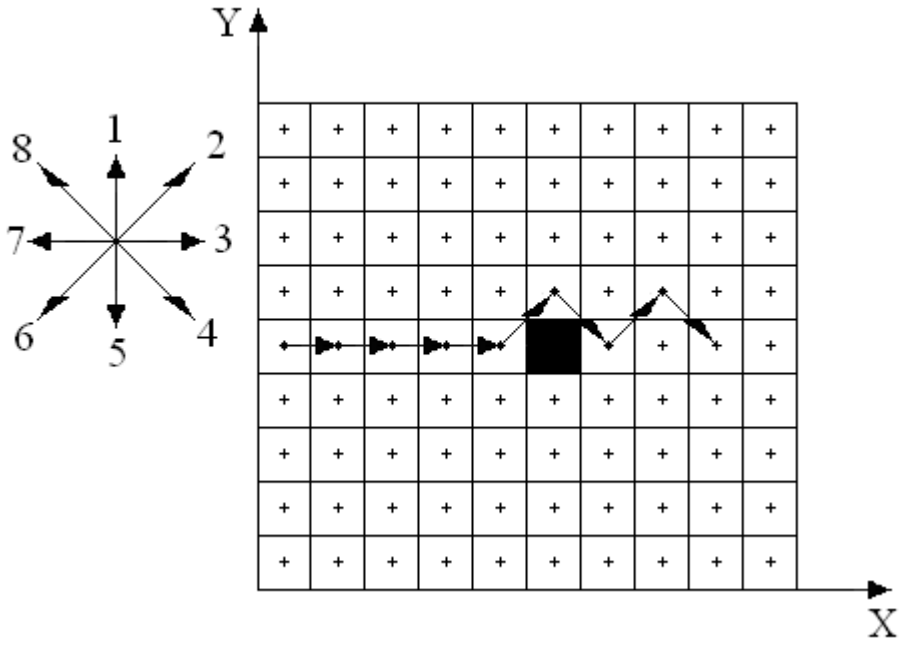


Figura 6.35 Exemplo 2: existe obstáculo na célula (5, 4)

Se não existisse obstáculo na célula (5, 4) o planejador determinaria uma sequência de movimentos 3 como resposta. Mas, as condições das células (7, 4), (6, 4) e (5, 4) são analisadas pela rede neural. Como na célula (5, 4) existe um obstáculo, o planejador não pode escolher como primeira ação o movimento 3. Numa situação como essa, o planejador pode escolher o movimento 2 ou 4.

Antes de decidir qual movimento escolher, as células que estão na direção 4 são analisadas pela rede neural, ou seja, as células (7, 5), (6, 6) e (5, 7). Também as células que estão na direção do movimento 2 são analisadas pela rede neural, no caso as células (7, 3), (6, 2) e (5, 1). Como as respostas para as duas análises é zero, o planejador pode escolher tanto o movimento 2 quanto o movimento 4. Numa situação como essa o planejador escolhe o movimento que leva a posição mais próxima da origem.

Observa-se que a plataforma iria para a posição (7, 3) se fosse escolhido o movimento 2. E iria para a posição (7, 5) se fosse escolhido o movimento 4. Percebe-se que as posições (7, 3) e (7, 5) estão exatamente à mesma distância da posição origem do robô, (0, 4). Nestas

condições, o planejador escolhe o movimento de número maior, no caso, movimento 4.

A figura 6.36 mostra o exemplo 3.

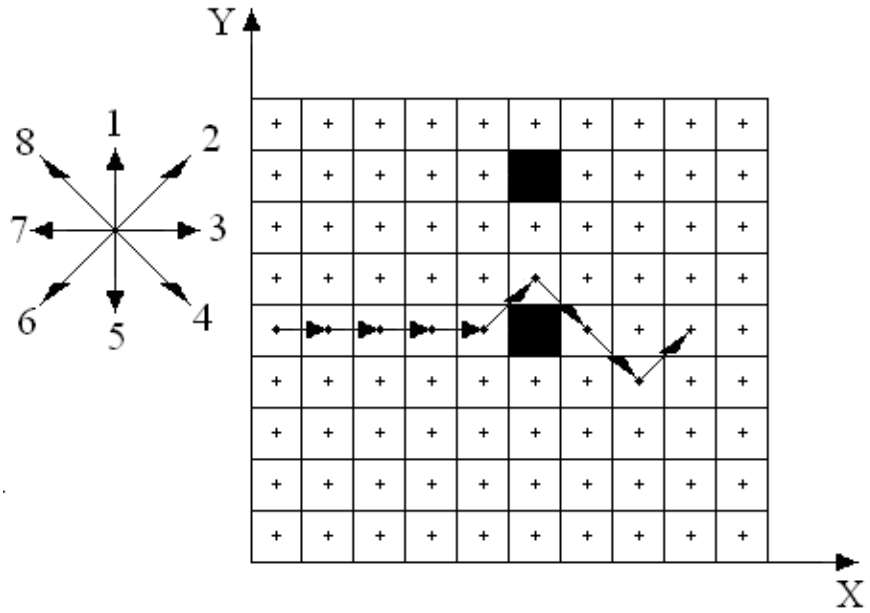


Figura 6.36 Exemplo 3: existem obstáculos nas células (5, 4) e (5, 7)

Observa-se que existe um obstáculo na direção do movimento 3. Assim, o planejador analisa as condições da direção 4 e 2. Como a direção do movimento 4 está obstruída, o planejador escolhe o movimento 2 como primeira ação.

A figura 6.37 mostra o exemplo 4.

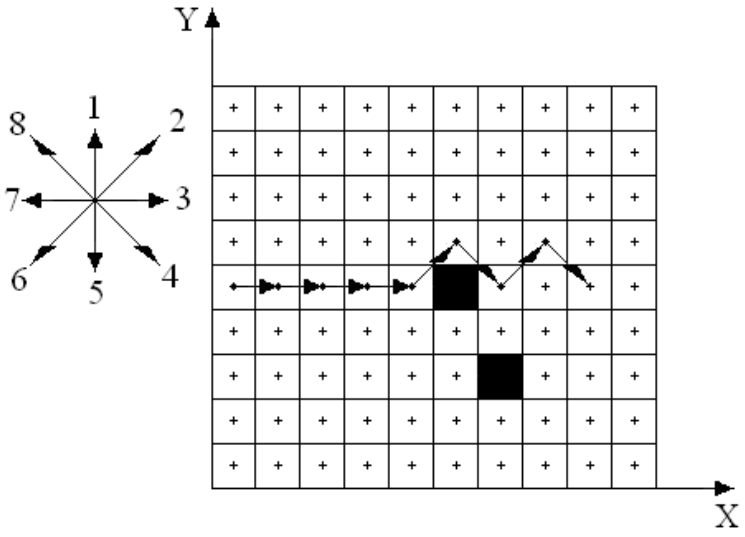


Figura 6.37 Exemplo 4: existem obstáculos nas células (5, 4) e (6, 2)

Nessa situação a primeira tentativa seria o movimento 3. Mas a existência de obstáculo na célula (5, 4) não permite a execução desse movimento. Assim, o planejador verifica as condições das células na direção do movimento 4 e 2. A existência de obstáculo na célula (6, 2) obriga o Subsistema de Planejamento escolher o movimento 4.

A figura 6.38 mostra o exemplo 5.

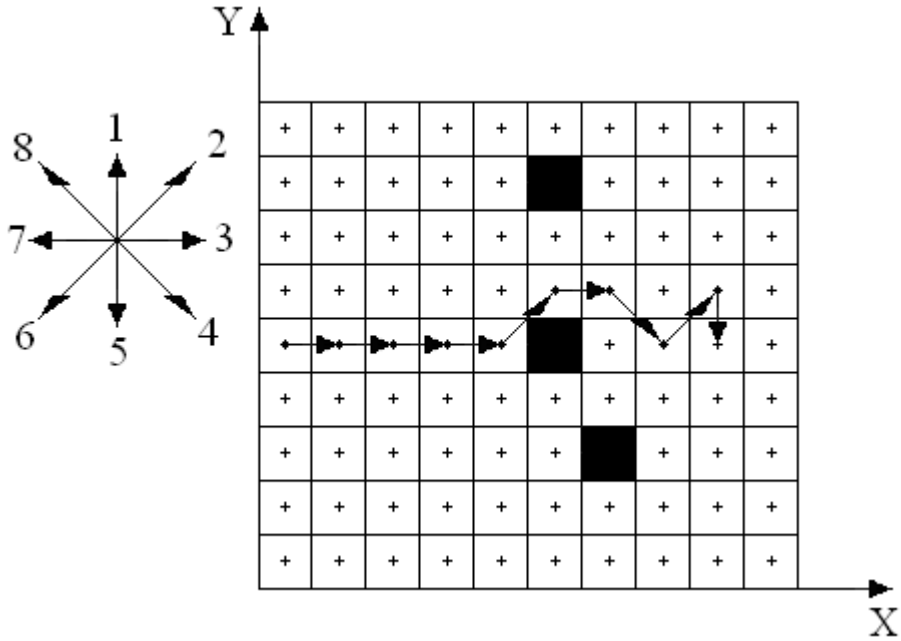


Figura 6.38 Exemplo 5: existem obstáculos nas células (5, 4), (5, 7) e (6, 2)

No exemplo 5 a primeira opção, que seria o movimento 3, não é possível devido a existência de obstáculo na célula (5, 4). As duas próximas opções de movimentos, 4 e 2, também não são possíveis devido a existência de obstáculos nas células (5, 7) e (6, 2). Assim, o Subsistema de Planejamento verifica as condições nas direções do movimento 5 e 1.

Como o sistema está configurado para analisar três células, as células (8, 5) (8, 6) e (8, 7), localizadas na direção do movimento 5, são analisadas pela rede neural. As células (8, 3) (8, 2) e (8, 1), localizadas na direção do movimento 1, também são analisadas pela rede neural. Como a rede não detecta obstáculos, pode-se optar tanto pelo movimento 5 quanto pelo movimento 1. Numa situação como essa, o Subsistema escolhe o movimento que leva a

uma posição mais próxima da origem. Como tanto o movimento 1 quanto o movimento 5 levam a posições equidistantes da origem, o subsistema escolhe o movimento de maior valor, no caso, movimento 5.

A figura 6.39 mostra o exemplo 6.

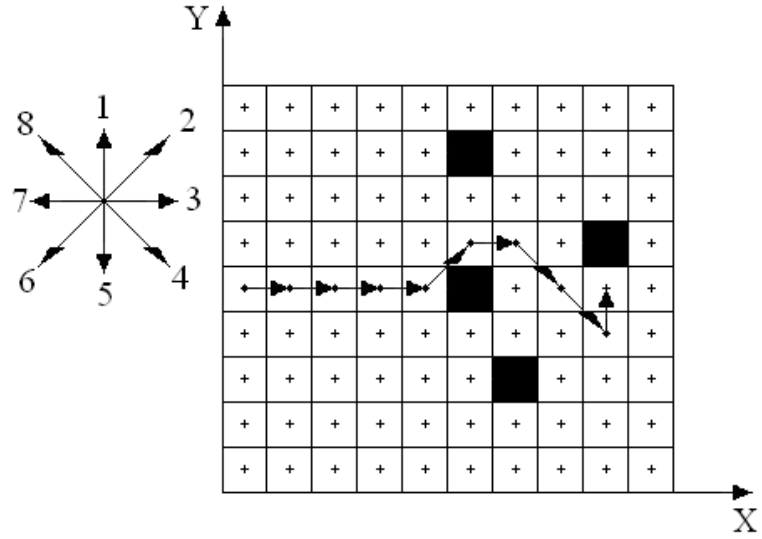


Figura 6.39 Exemplo 6: existem obstáculos nas células (5, 4), (5, 7), (6, 2) e (8, 5)

Nessa situação, o Subsistema de Planejamento fica impossibilitado de selecionar os movimentos 3, 4, 2 e 5. Sendo, portanto, a próxima opção o movimento 1.

A figura 6.40 mostra o exemplo 7.

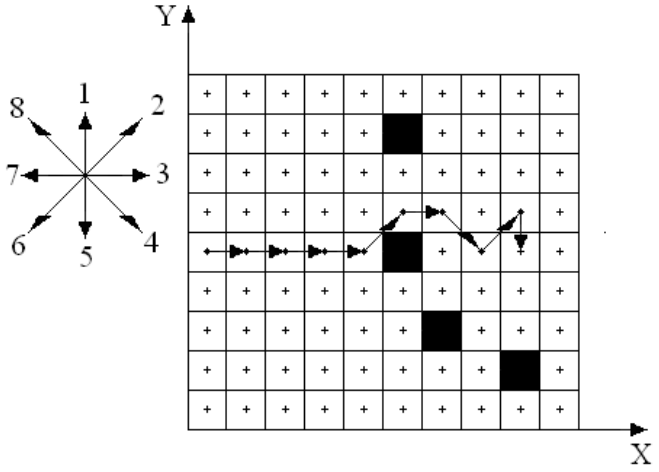


Figura 6.40 Exemplo 7: existem obstáculos nas células (5, 4), (5, 7), (6, 2) e (8, 1)

Agora, o Subsistema de Sensoriamento fica impossibilitado de selecionar os movimentos 3, 4, 2 e 1. Sendo, portanto, a próxima opção o movimento 5.

A figura 6.41 mostra o exemplo 8.

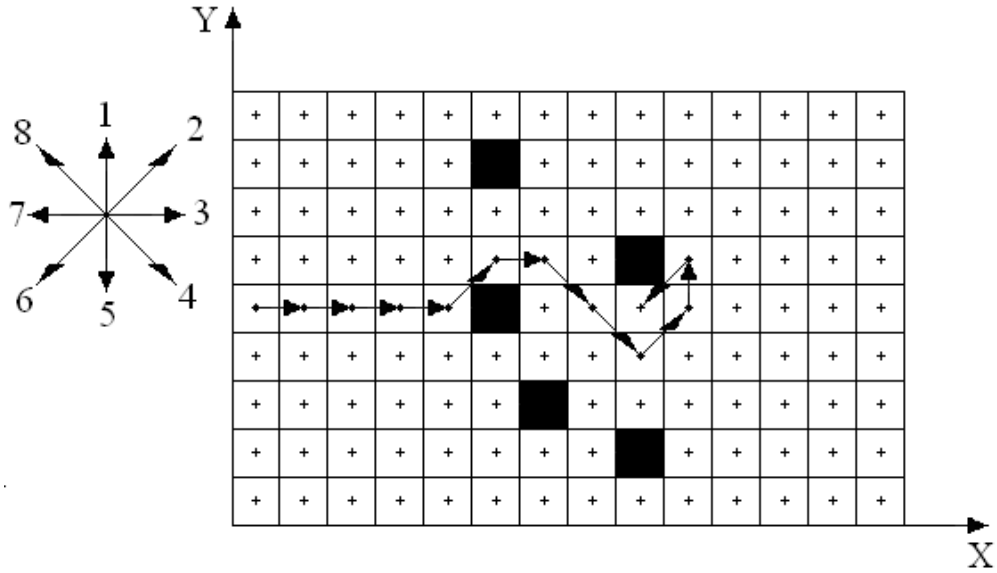


Figura 6.41 Exemplo 8: existem obstáculos nas células (5, 4), (5, 7), (6, 2), (8,1) e (8, 5)

No exemplo 8, os movimentos 3, 4, 2, 5 e 1 não são possíveis de serem realizados devido a existência de obstáculos. Assim, o Subsistema de Planejamento verifica as condições na direção do movimento 6 e 8. Portanto, o Grau de Evidência Favorável das células (9, 5), (10, 6) e (11, 7) e das células (9, 3), (10, 2) e (11, 1), relativas a direção do movimento 6 e 8, respectivamente, são as entradas da rede neural.

Como a rede neural não detecta obstáculos nas células que estão na direção dos movimentos 6 e 8, o Subsistema Planejamento pode escolher tanto o movimento 6 quanto o movimento 8. Ele escolheria o movimento que levaria a posição mais próxima da origem. Como tanto o movimento 6 quanto o movimento 8 levam a posições equidistantes da origem, o planejador escolhe o movimento de maior valor, no caso, movimento 6.

A figura 6.42 mostra o exemplo 9.

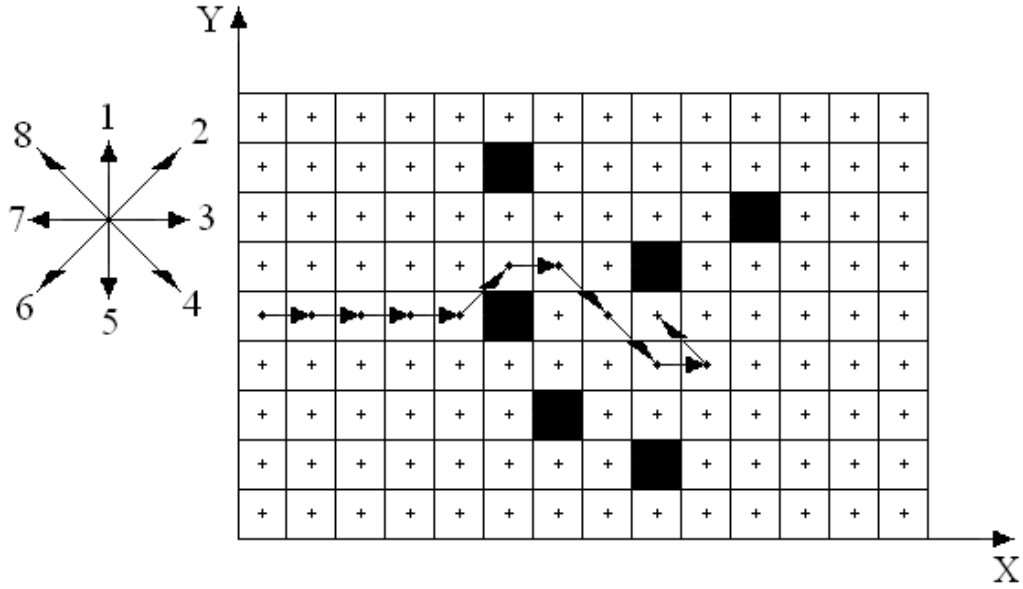


Figura 6.42 Exemplo 9: existem obstáculos nas células (5, 4), (5, 7), (6, 2), (8,1), (8, 5) e (10, 6)

No exemplo 9, o Subsistema de Planejamento fica impossibilitado de selecionar os movimentos 3, 4, 2, 5, 1 e 6. Sendo, portanto, a próxima opção o movimento 8.

A figura 6.43 mostra o exemplo 10.

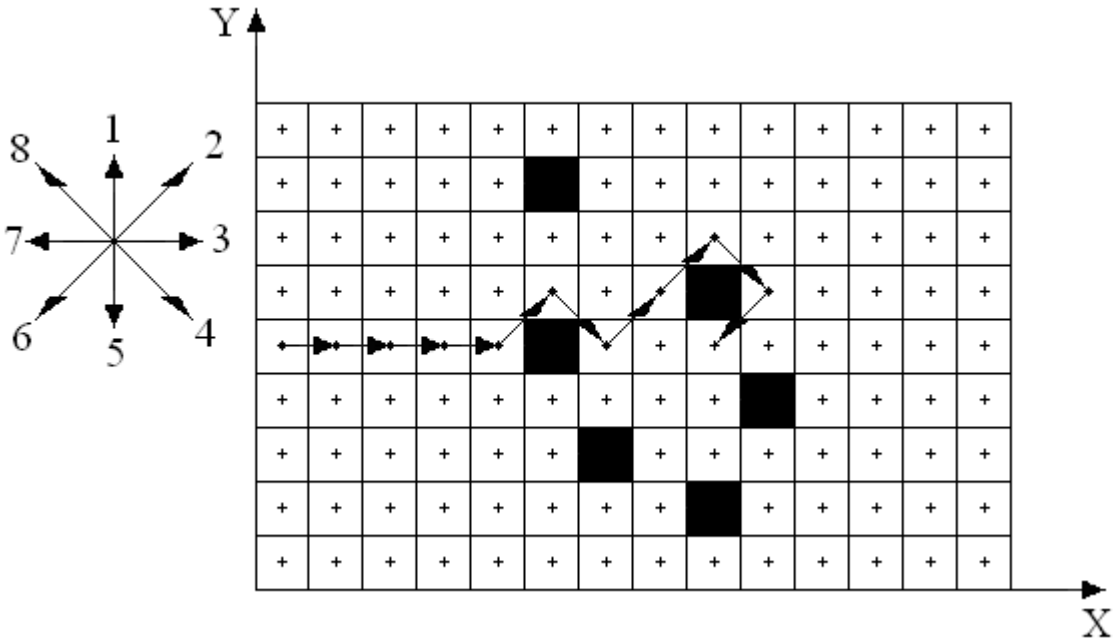


Figura 6.43 Exemplo 10: existem obstáculos nas células (5, 4), (5, 7), (6, 2), (8,1), (8, 5) e (9, 3)

Nesse exemplo, o Subsistema de Planejamento fica impossibilitado de selecionar os

movimentos 3, 4, 2, 5, 1 e 8. Sendo, portanto, a próxima opção o movimento 6.

A figura 6.44 mostra o exemplo 11.

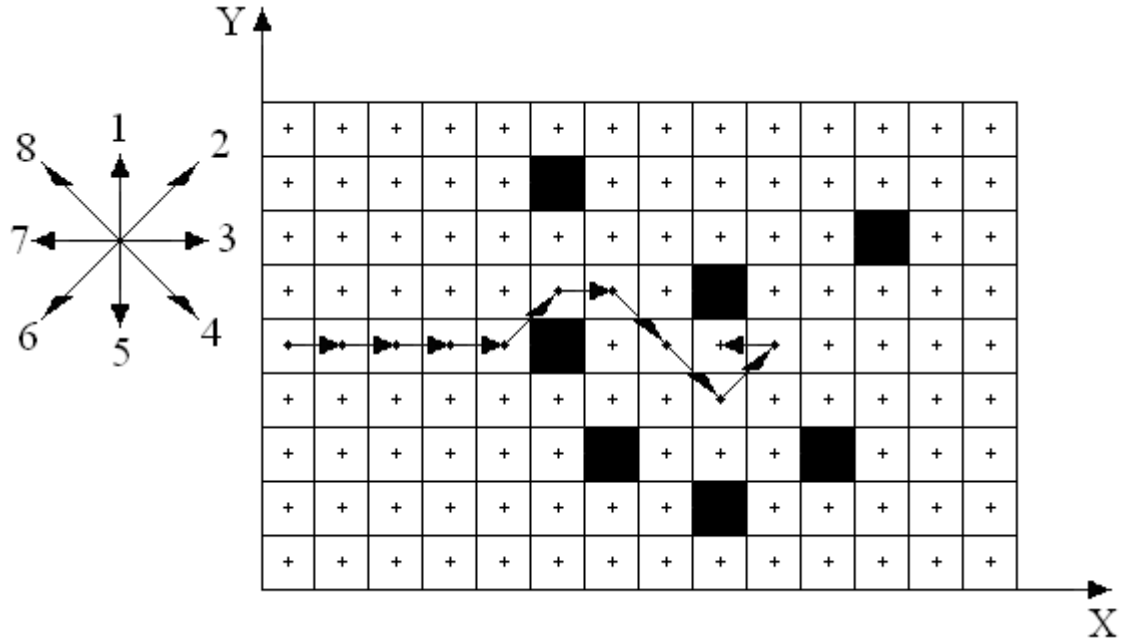


Figura 6.44 Exemplo 11: existem obstáculos nas células (5, 4), (5, 7), (6, 2), (8,1), (8, 5), (10, 2) e (11, 6)

Nessa situação o único primeiro movimento possível é o movimento 7. Esse movimento é exatamente o oposto ao movimento 3 que seria realizado caso não houvesse obstáculos no ambiente.

O Subsistema de Planejamento também considera como obstruídas todas as células que tenham sido utilizadas em trajetórias anteriores. Isso acontece para evitar que o robô execute sempre uma mesma sequência de movimentos, ficando de certa forma preso, sempre passando por um mesmo caminho, e nunca alcançando a posição destino.

Na figura 6.45, as células de cor cinza possuem um alto valor de Grau de Evidência Favorável pelo fato de já terem sido utilizadas numa trajetória anterior.

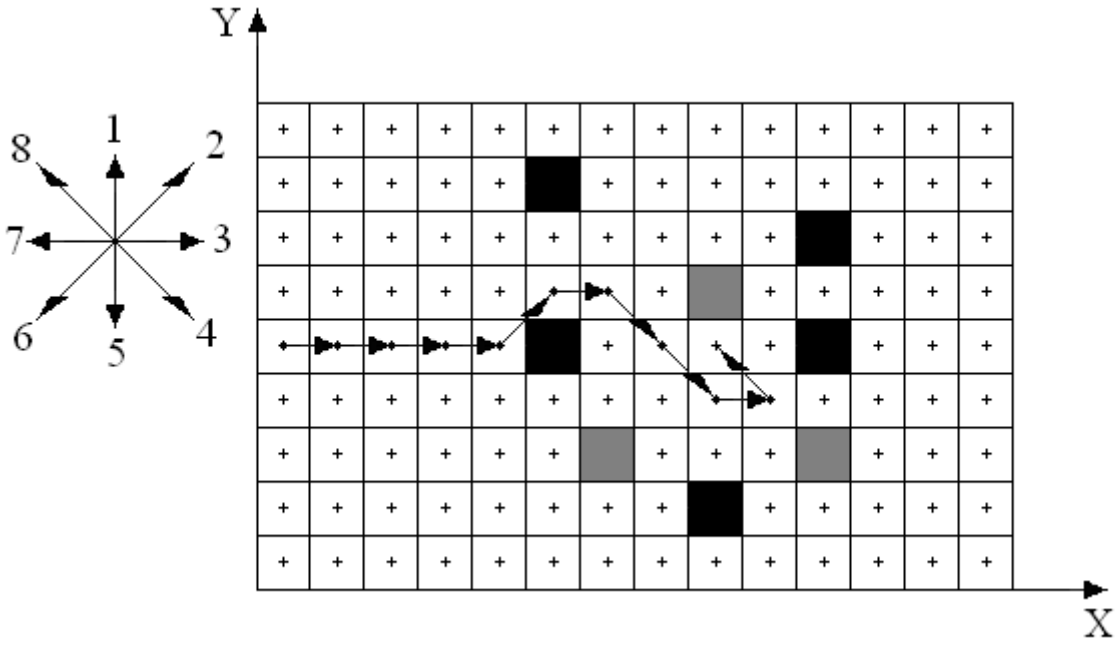


Figura 6.45 Exemplo 12: células que já foram utilizadas em trajetórias anteriores

Assim, numa situação como a mostrada na figura 6.45, onde o robô fica cercado por obstáculos, o Subsistema de Planejamento gera uma trajetória que passe por uma célula que tenha valor alto de Grau de Evidência Favorável pelo fat de já ÿer sido utilizada numa trajetória anterior.

Em todos oo exemplts mostrados até aqui se considerou as células que compõe o ambiente como totalmente livres de obstáculos, ou seja, com Grau de Evidência Favorável igual a 0,0, sendo representadas pela cor branca. Ou se considerou as células como totalmente obstruídas, ou seja, com Grau de Evidência Favorável igual a 1,0, sendo representadas pelas cores preta e cinza. Mas existe a possibilidade da célula possuir valor de Grau de Evidência Favorável entre 0,0 e 1,0.

Do exemplo 2 até o exemplo 12 foram apresentadas situações onde existem obstáculos no caminho que o robô deve seguir. Essa situação acontece quando a rede neural, mostrada na figura 6.33, composta por Células Neurais Artificiais Paraconsistentes de Conexão Lógica Simples no Processo de Maximização (OU), detecta obstáculo. A existência de obstáculo

numa célula se caracteriza por ela possuir um Grau de Evidência Favorável acima de um determinado valor.

Caso a análise realizada pela rede neural, composta por Células Neurais Artificiais Paraconsistentes de Conexão Lógica Simples no Processo de Maximização (OU), não detecte obstáculo nas posições que compõe o ambiente, estas são analisadas por uma rede neural composta por Células Neurais Artificiais Paraconsistentes Analíticas com a estrutura mostrada na figura 6.46.

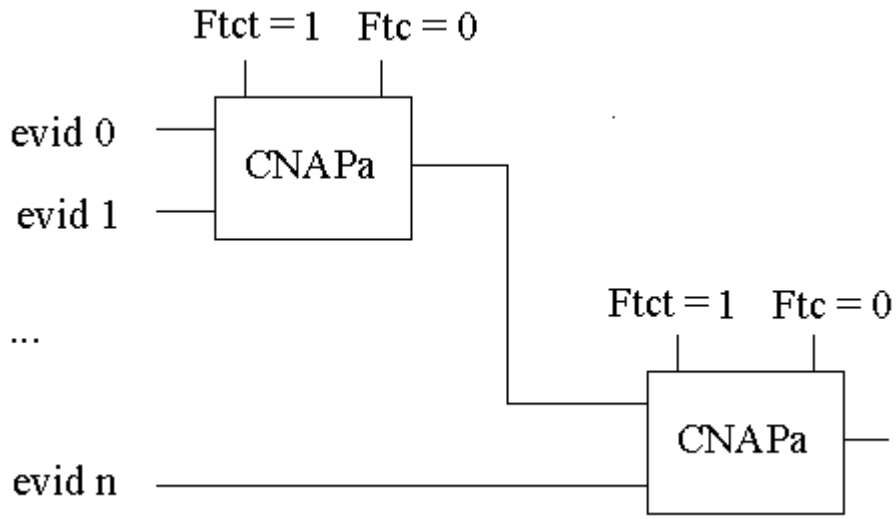


Figura 6.46 Rede neural artificial paraconsistente utilizada pelo subsistema de planejamento para determinar qual o valor de grau de evidência de uma direção

A saída da rede neural apresentada na figura 6.46 é o Grau de Evidência Resultante das células analisadas.

A figura 6.47 mostra uma situação onde o planejador analisa duas possibilidades de caminhos a seguir.

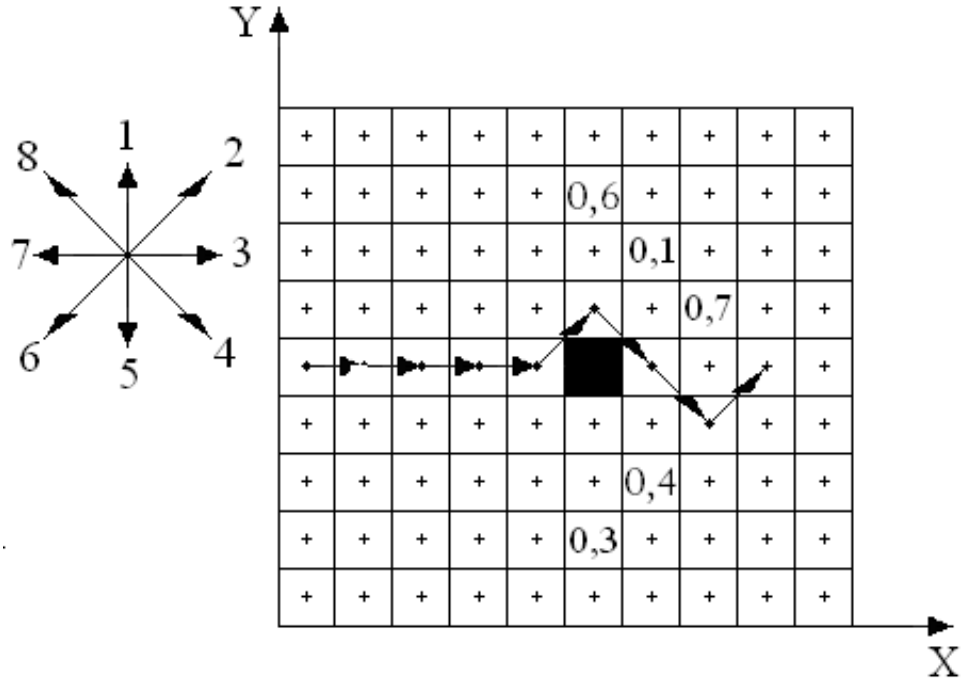


Figura 6.47 Exemplo 13: existência de células que possuem grau de evidência com valor intermediário

Observa-se que o primeiro movimento escolhido é aquele cuja direção possui o menor Grau de Evidência Resultante.

6.7 Testes realizados com o subsistema de planejamento

Apresentam-se três testes realizados com o Subsistema de Planejamento. Nas três situações o Subsistema de Planejamento estava configurado da mesma forma, apenas a localização dos obstáculos em cada teste era diferente. O subsistema foi configurado para que as redes neurais analisassem três células e o valor limite para uma célula não ser considerada ocupada era 0,9. O banco de dados com as condições de cada célula foi preenchido manualmente.

Teste 1

A figura 6.48 mostra o ambiente utilizado no teste 1.

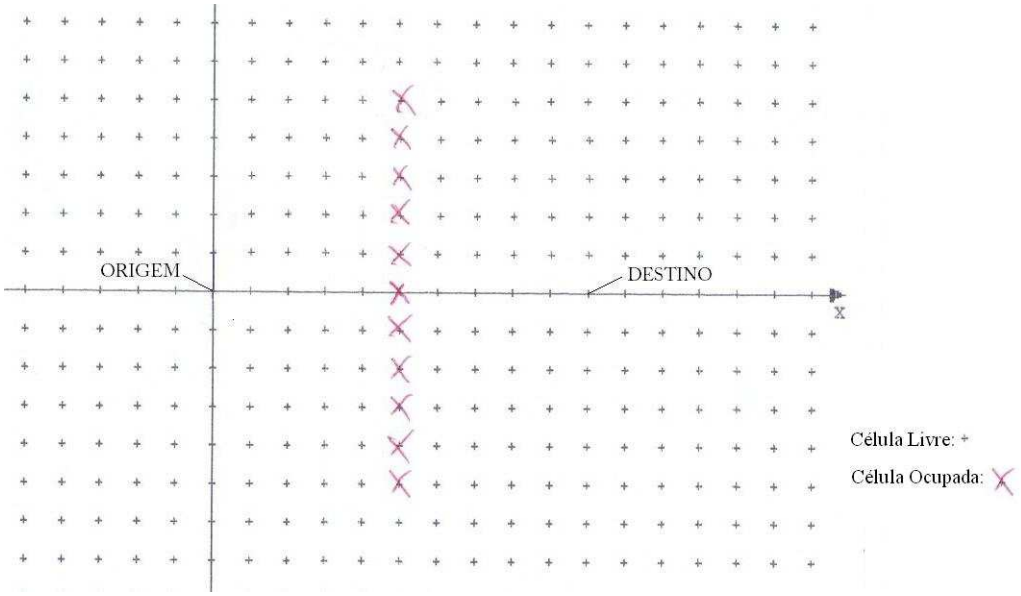


Figura 6.48 Ambiente do teste 1 do subsistema de planejamento

Observa-se que a origem é a coordenada (0, 0) e o destino é a coordenada (10, 0). As coordenadas marcadas com “x” são consideradas obstruídas.

A figura 6.49 mostra o resultado do teste 1.

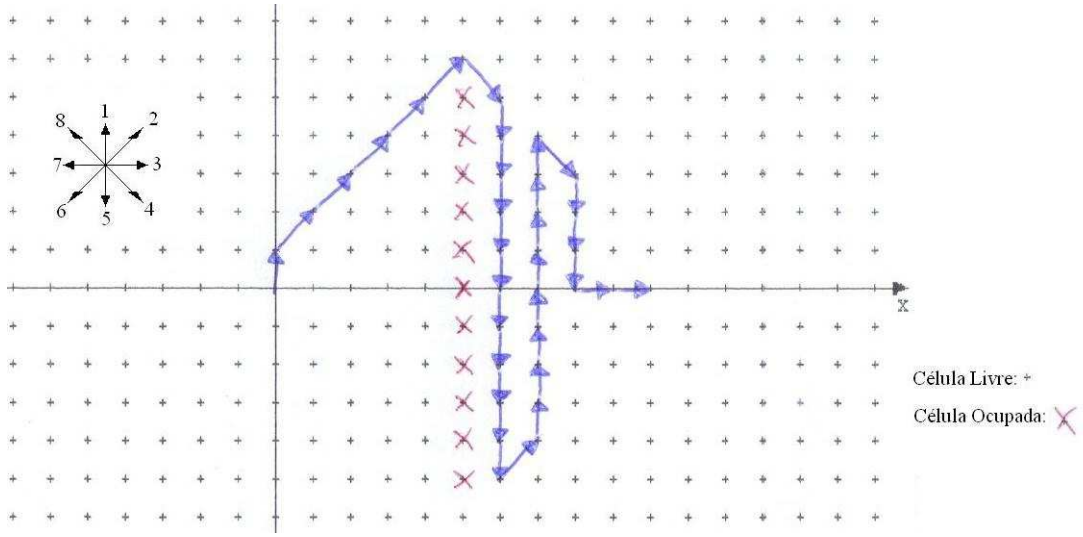


Figura 6.49 Resultado do teste 1 do subsistema de planejamento

Teste 2

A figura 6.50 mostra o ambiente utilizado no teste 2.

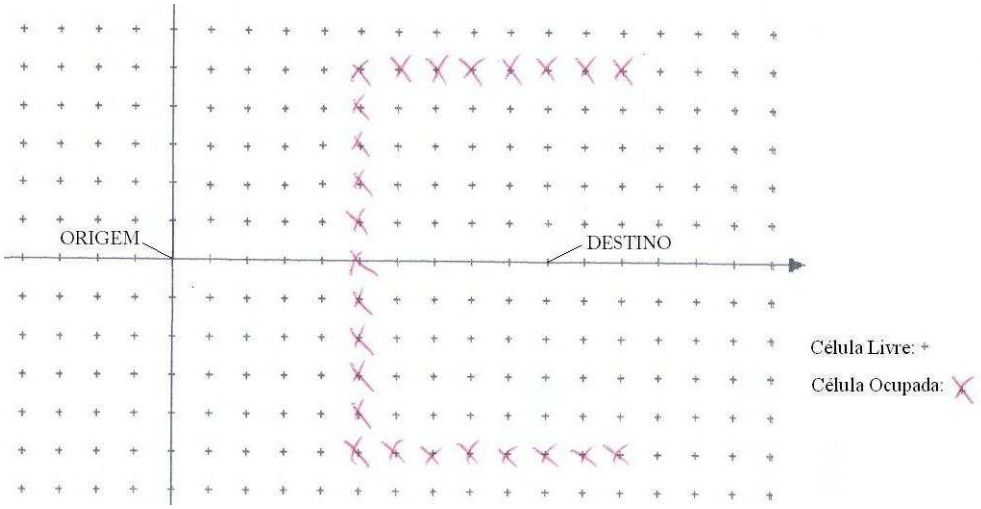


Figura 6.50 Ambiente do teste 2 do subsistema de planejamento

As figuras 6.51, 6.52, 6.53 e 6.54 mostram o resultado do teste 2.

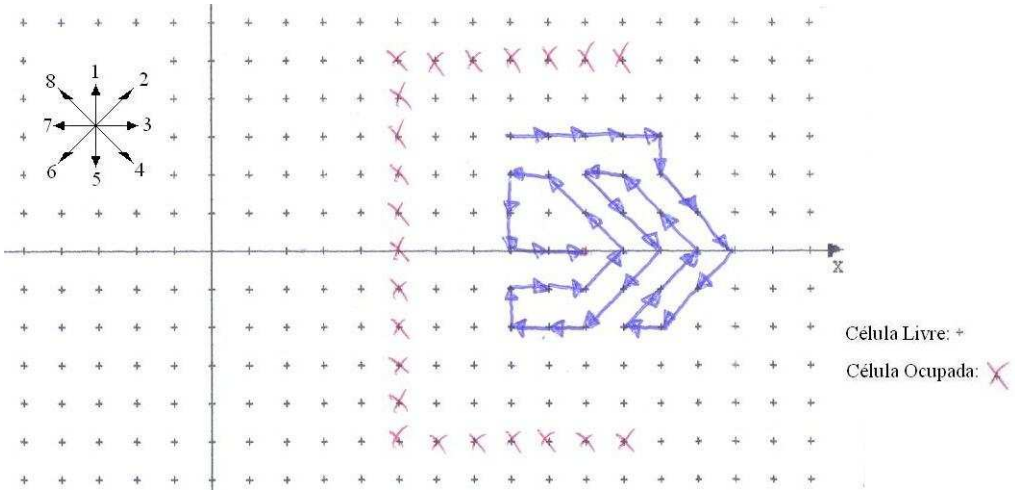


Figura 6.51 Resultado do teste 2

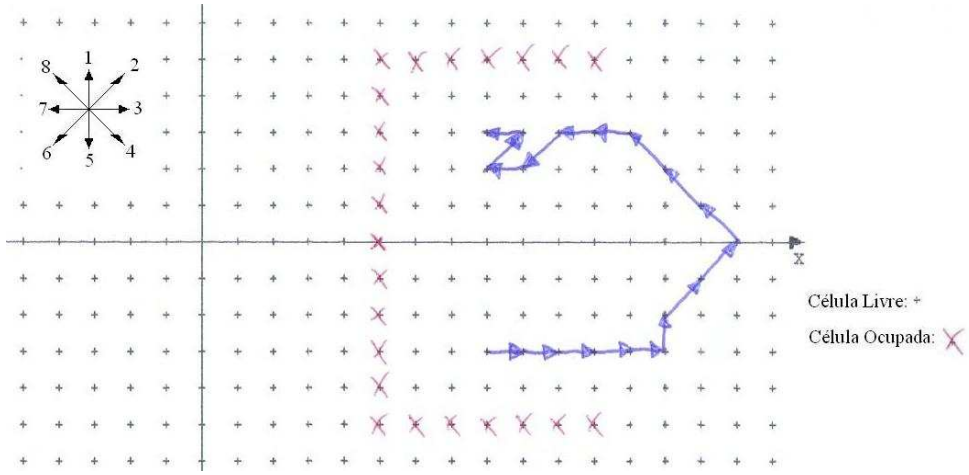


Figura 6.52 Resultado do teste 2

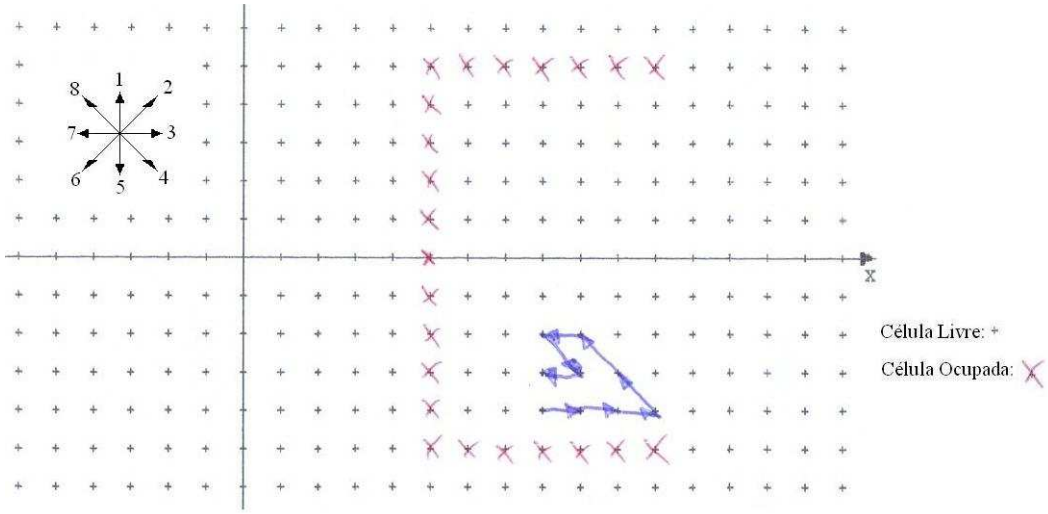


Figura 6.53 Resultado do teste 2

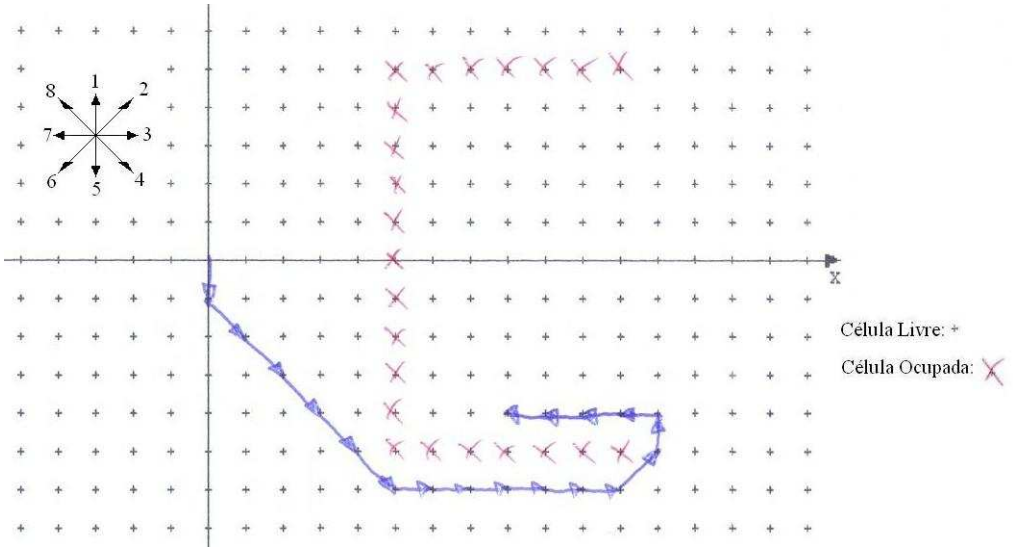


Figura 6.54 Resultado do teste 2

Teste 3

A figura 6.55 mostra o ambiente utilizado no teste 3.

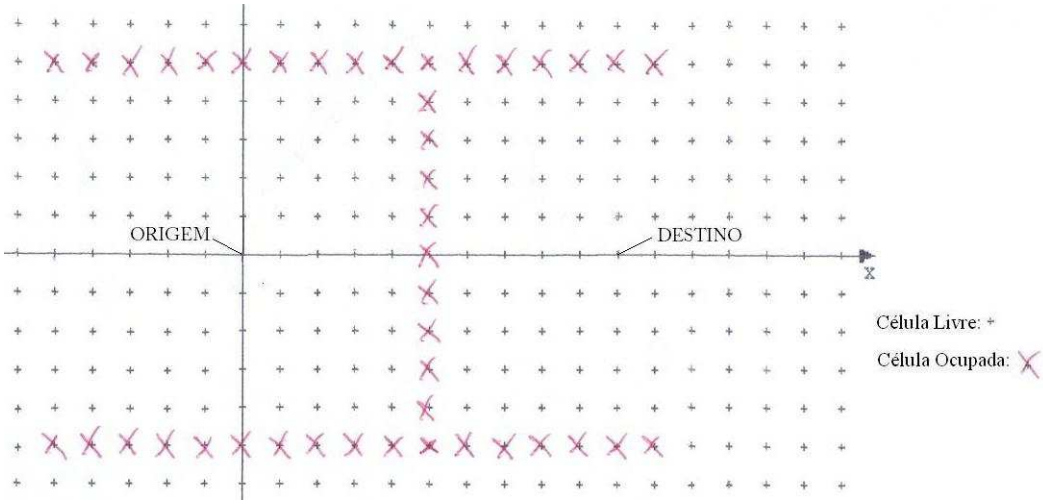


Figura 6.55 Ambiente do teste 3 do subsistema de planejamento

As figuras 6.56, 6.57, 6.58 e 6.59 mostram o resultado do teste 3.

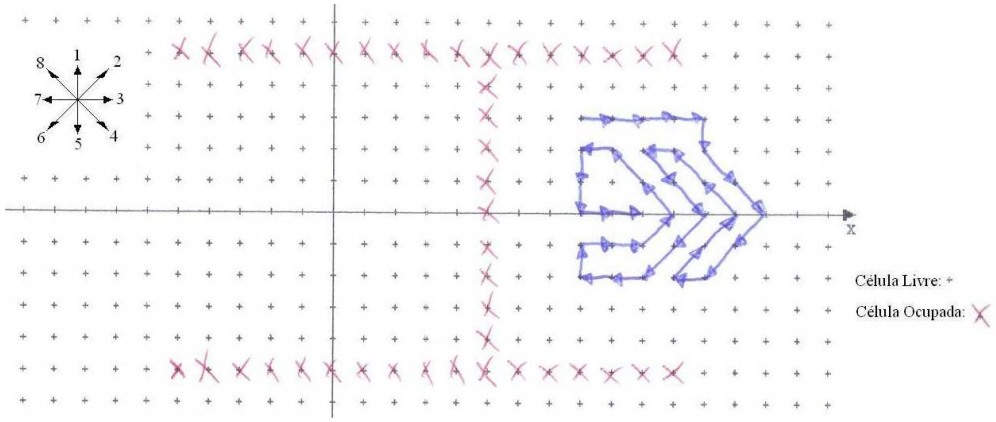


Figura 6.56 Resultado do teste 3

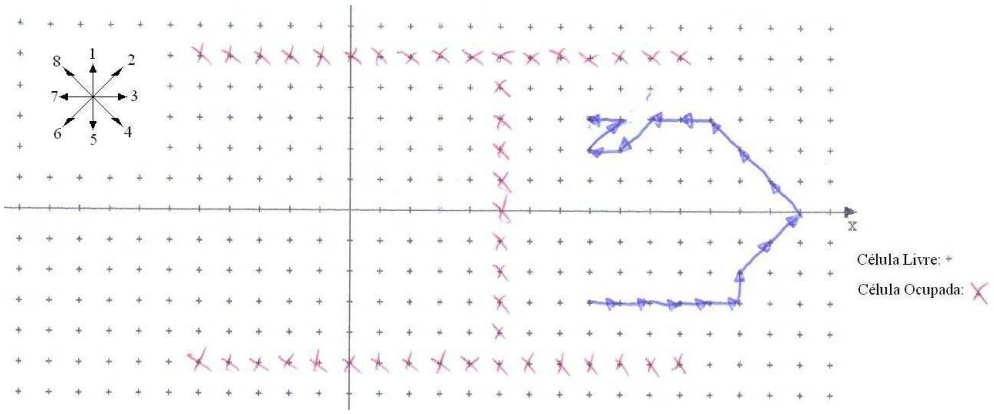


Figura 6.57 Resultado do teste 3

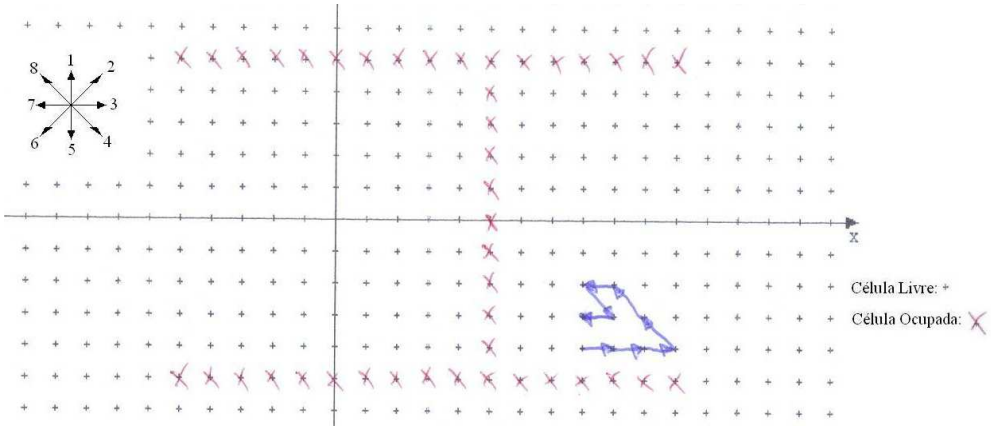


Figura 6.58 Resultado do teste 3

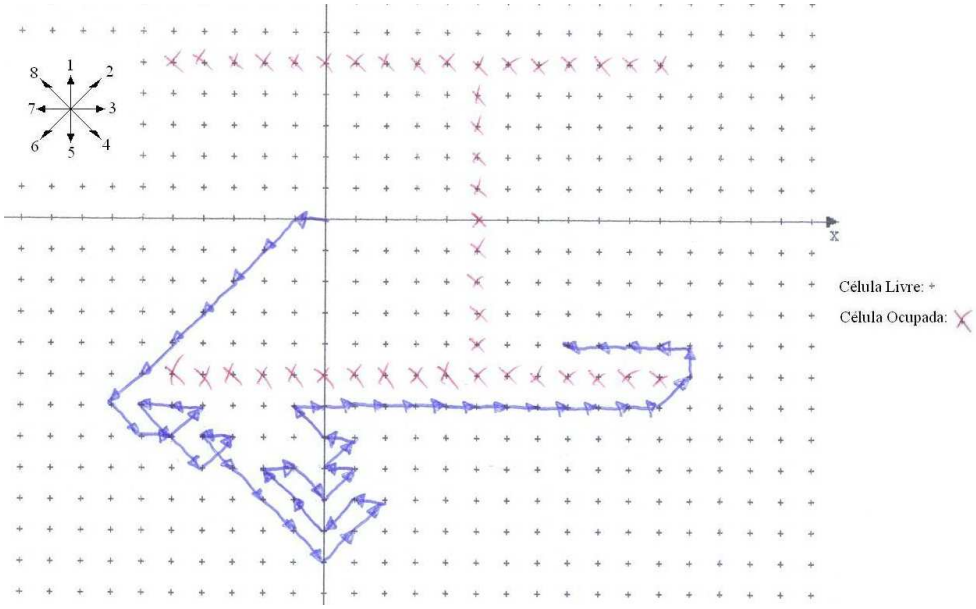


Figura 6.59 Resultado do teste 3

Analisando os testes realizados, percebe-se que o planejador gera muito mais ações que o necessário para que o robô saia da posição inicial e encontre a posição destino. Isso não é exatamente um problema, pois, pode-se fazer com que a plataforma mecânica execute o último movimento solicitado pelo Subsistema de Planejamento (o planejador gera as ações do destino para a origem) e em seguida solicite outro planejamento. Repetindo essas ações até que o robô chegue ao destino, tem-se uma trajetória próxima da ideal conectando o ponto de origem ao ponto destino. Baseado nesta ideia, foi desenvolvido uma versão do planejador cuja resposta é o que seria a trajetória realmente executada pela plataforma mecânica.

As figuras 6.60, 6.61 e 6.62 mostram os resultados otimizados dos testes 1, 2 e 3.

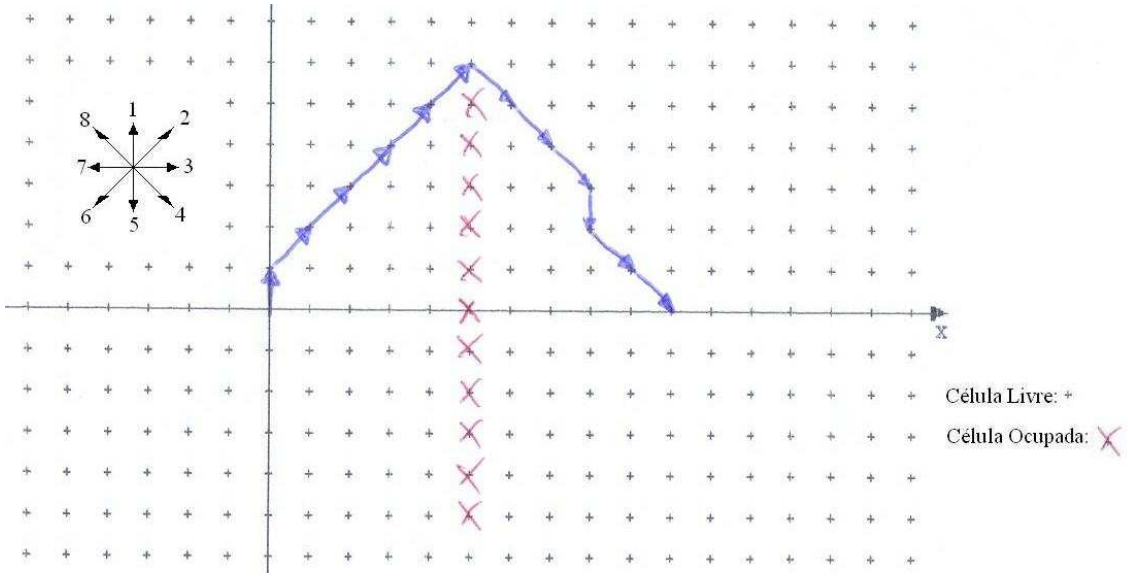


Figura 6.60 Resultado otimizado do teste 1

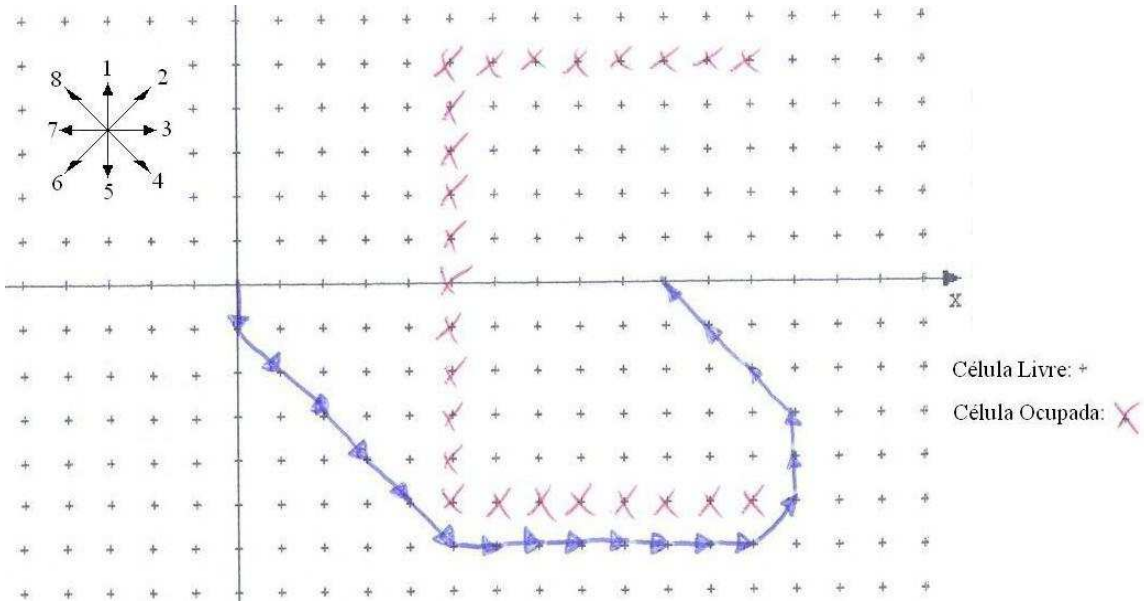


Figura 6.61 Resultado otimizado do teste 2

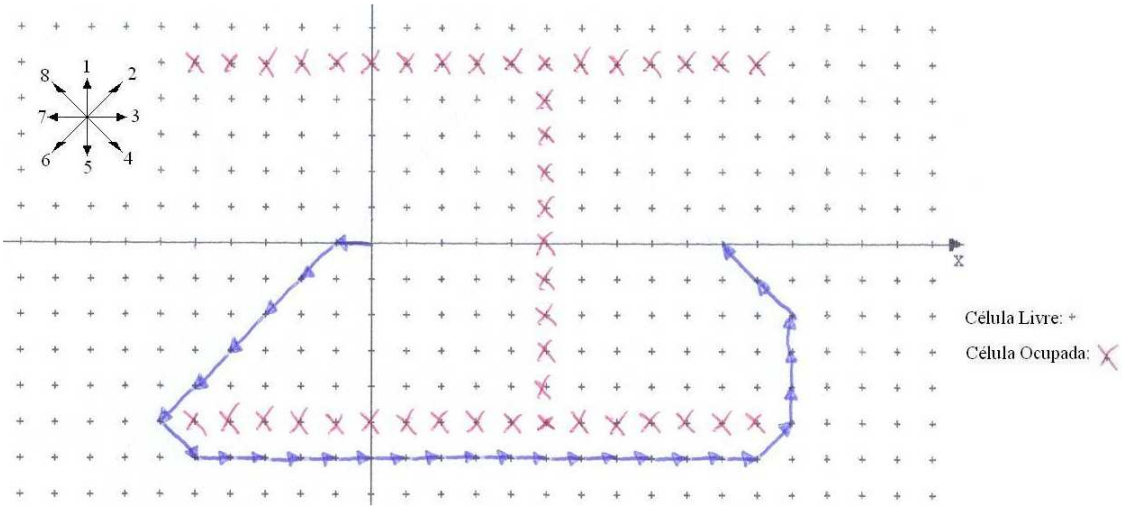


Figura 6.62 Resultado otimizado do teste 3

A trajetória realizada pela plataforma mecânica é satisfatória. Pois, conforme observado nos resultados apresentados acima, a plataforma se desloca da origem até o destino com um número relativamente baixo de movimentos.

CAPÍTULO 7 - CONCLUSÃO

Neste trabalho propôs-se um Sistema de Navegação e Controle para um Robô Móvel Autônomo baseado na Lógica Paraconsistente Anotada Evidencial Eτ através da utilização das Redes Neurais Artificiais Paraconsistentes. A utilização desse tipo de lógica permite ao sistema lidar melhor com situações difusas, inconsistentes e paracompletas.

O sistema proposto compõe-se por três módulos: Subsistema de Sensoriamento, Subsistema de Planejamento e Subsistema Mecânico.

O Subsistema de Sensoriamento recebe as informações advindas dos sensores e armazena num banco de dados as informações relativas a cada célula que compõe o ambiente em torno do robô.

Nesse subsistema existe uma Rede Neural Artificial Paraconsistente composta por Células Neurais Artificiais Paraconsistentes Analíticas e por Células Neurais Artificiais Paraconsistentes de Passagem. Esse tipo de estrutura permite a construção de um sistema de sensoriamento eficiente e relativamente simples.

Vários testes foram realizados com o Subsistema de Sensoriamento e constatou-se que o funcionamento foi satisfatório. Os testes consistiram da aplicação na entrada do sistema de sinais que simulavam informações advindas de sensores. Em todos os casos a resposta foi a esperada.

Um próximo passo seria a ligação de sensores na entrada desse sistema para se verificar o funcionamento do mesmo em situações reais.

O Subsistema de Planejamento é responsável por gerar a sequência de ações que o robô deve executar para encontrar a posição destino. Esse sistema consulta o banco de dados que é atualizado pelo Subsistema de Sensoriamento para saber onde existem obstáculos no

ambiente em torno do robô. Com essa informação ele gera a sequência de movimentos que o robô deve executar para alcançar um destino num ambiente não estruturado.

No Subsistema de Planejamento também existem Redes Neurais Artificiais Paraconsistentes. Além de uma estrutura similar à utilizada pelo Subsistema de Sensoriamento, ele também se utiliza de uma estrutura composta por Células Neurais Artificiais Paraconsistentes de Conexão Lógica Simples no Processo de Maximização (OU). Isso permitiu a simplificação da construção do Subsistema de Planejamento.

Os diversos testes realizados com o Subsistema de Planejamento foram satisfatórios. Os testes consistiam em preencher o banco de dados manualmente, criando um ambiente previamente conhecido. Assim, dado um ponto de partida e um ponto de chegada, o sistema sempre gerou uma sequência de passos interligando o ponto de partida ao ponto de chegada.

O grau de dificuldade de cada teste realizado com o Subsistema de Planejamento foi aumentado gradativamente, chegando a situações de grande dificuldade. Em todos os casos o sistema foi capaz de encontrar um caminho ligando o ponto de origem ao ponto destino. Também em todos os casos o caminho encontrado foi satisfatório e, em muitas situações, o caminho encontrado foi o ótimo.

Para que o robô se movimente de forma autônoma num ambiente não estruturado ainda necessita-se construir o Subsistema Mecânico, ou seja, a plataforma mecânica capaz de carregar todos os dispositivos que constituem os Subsistemas de Sensoriamento e Planejamento. Além disso, essa plataforma deve executar a sequência de ações solicitadas pelo Sistema de Planejamento e informar aos outros subsistemas a sua localização exata. O Subsistema Mecânico também deve tratar os sinais gerados pelos sensores antes de repassar essas informações para o Subsistema de Sensoriamento.

Esses três módulos funcionando de forma independente e interligada formam um robô móvel autônomo capaz de encontrar um destino pré-determinado num ambiente não estruturado.

A contribuição deste trabalho é a proposta de um sistema de navegação e controle de um robô móvel autônomo, com as características descritas acima, baseado na lógica paraconsistente anotada Eτ.

Uma interface gráfica para o Subsistema de Planejamento capaz de mostrar, em uma tela de computador, a rota gerada pelo planejador para que o robô saia de uma posição inicial e encontre uma posição destino está atualmente em desenvolvimento.

O sistema proposto possui aplicações promissoras e merece uma continuidade de estudos com a implantação dos Subsistemas de Sensoriamento e de Planejamento em uma plataforma mecânica compondo o Subsistema Mecânico, completando assim o conjunto para aplicações em robótica autônoma.

REFERÊNCIAS BIBLIOGRÁFICAS

ABE, J. M. **Fundamentos da Lógica Anotada.** 1992. 135 fl. Tese (Doutorado). Universidade de São Paulo, São Paulo, 1992.

ABE, J. M. Some Aspects of Paraconsistent Systems and Applications. **Logique et Analyse**, v. 157, p. 83-96, 1997.

ABE, J. M. Annotated logics $Q\tau$ and model theory, in Logic, Artificial Intelligence, and Robotics. In: 2^{ND} CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC'2001. In: ABE, J.M.; DA SILVA FILHO, J. I. [edts.] **Frontiers in Artificial Intelligence and Its Applications**. Amsterdam/Tokyo: IOS Press/Ohmsha, v. 71, p. 1-12, 287, 2001.

ABE, J. M. & AKAMA, S. A Logical System for Reasoning with Fuzziness and Inconsistencies. In: IASTED INTERNATIONAL CONFERENCE ON ARTITFICIAL INTELLIGENCE AND SOFT COMPUTING (ASC'99). **Anais...** Honolulu, Hawaii, USA, 1999, p. 221-225.

ABE, J. M.; ÁVILA, B. C.; NAKAMATSU, K. Paraconsistent Annotated Logic Programming – Paralog. **International Journal of Computing Anticipatory Systems**, v. 6, p. 51-65, 1999.

ABE, J. M. & DA SILVA FILHO, J. I. Inconsistency and Electronic Circuits. In: INTERNATIONAL ICSC SYMPOSIUM ON ENGINEERING OF INTELLIGENT SYSTEMS (EIS'98). v 3. ALPAYDIN, E. [ed.]. **Artificial Intelligence**, ICSC Academic Press International Computer Science Conventions, Canada/Switzerland, p. 191-197, 1998.

ABE, J. M. & DA SILVA FILHO, J. I. A Para-Analyser Method to Increase Robot Availability Through Maintenance. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL LOGISTIC 2001. **Anais...** 9-12 July 2001, Okinawa, Japan, p. 327-337, 2001.

ABE, J. M. & DA SILVA FILHO, J. I. Simulating Inconsistencies in a Paraconsistent Logic Controller. In : FIFTH INTERNATIONAL CONFERENCE ON COMPUTING ANTICIPATORY SYSTEMS, CASYS'2001. CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, Liège, Belgium, 2001b.

ABE, J. M. & J. I. DA SILVA FILHO, Manipulating Conflicts and Uncertainties in Robotics, **Journal of Multiple-Valued Logic and Soft Computing**, Philadelfia, USA, Old City Publishing, v. 9, n. 2, p. 147-169, 2003.

ABE, J. M.; LAMBERT-TORRES, G.; DA SILVA FILHO, J. I.; TORRES, C. R.; MARTINS, H. G. Paraconsistent Autonomous Mobile Robot Emmy III. In: 6TH CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC'2007. **Proceedings of the VI Congress of Logic Applied to Technology.** Santos, São Paulo, 2007.

ABE, J. M.; TORRES, C. R.; LAMBERT-TORRES, G.; NAKAMATSU, K.; KONDO, M. Intelligent Paraconsistent Logic Controller and Autonomous Mobile Robot Emmy II. **Lecture Notes in Computer Science**, v. 4252, p. 851-857, 2006.

ABE, J. M., TORRES, C. R., LAMBERT-TORRES, G., NAKAMATSU, K., KONDO, M. Intelligent Paraconsistent Logic Controller and Autonomous Mobile Robot Emmy II. In: 10th International Conference on Knowledge-Based, Intelligent Information & Engineering Systems, KES2006, 2006, Bournemouth. Proceedings of the 10th International Conference on Knowledge-Based, Intelligent Information & Engineering Systems. Bournemouth - UK : KES Pub., 2006b.

ABE, J. M.; TORRES, C. R.; LAMBERT-TORRES, G.; DA SILVA FILHO, J. I.; MARTINS, H. G. Paraconsistent Autonomous Mobile Robot Emmy III. In: TORRES, G. L.; J. M. ABE; J. I. DA SILVA FILHO; H. G. MARTINS. [Orgs.]. **Advances in Technological Applications of Logic and Intelligent Systems.** 1 ed. Amsterdam: IOS Press, 2009, v. 186, p. 236-258.

AKAMA, S. & ABE, J. M. Fuzzy annotated logics. In: 8TH INTERNATIONAL CONFERENCE ON INFORMATION PROCESSING AND MANAGEMENT OF UNCERTAINTY IN KNOWLEDGE BASED SYSTEMS, IPMU'2000. **Anais...** Universidad Politécnica de Madrid (Spain), jul. 3-7, 2000, Madri, Espanha, v. 1, 504-508, 2000.

ALMEIDA PRADO, J. P.; ABE J. M.; SCALZITTI, A. Modeling the Behavior of Paraconsistent Robots. In: Advances in Logic Based Intelligent Systems. In: 5TH CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC'2005. In: NAKAMATSU K. & ABE, J. M. Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdan, v. 132, p. 120-126, 289, 2005.

ÁVILA, B. C. **Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança.** 1996. 133 fl. Tese (PhD)Universidade de São Paulo, São Paulo.

ÁVILA, B. C.; ABE, J. M.; PRADO, J. P. A. ParaLog-e: A Paraconsistent Evidential Logic Programming Language. In: XVII INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY. In: IEEE Computer Society Press, p 2-8, Valparaíso, Chile, 1997.

AIRFORCE-TECHNOLOGY.COM. **Predator RQ-1 / MQ-1 / MQ-9 Reaper - Unmanned Aerial Vehicle (UAV), USA.** Disponível em: <http://www.airforce-technology.com/ projects/predator/>. Acesso em 13 de jan. de 2010.

BARRETO, G. A.; ARAÚJO, A. F. R.; ROSA, M. O. Algoritmo de Busca Heurística Usando Redes Neurais Competitivas para Planejamento Ótimo de Trajetória de um Robô Móvel. In: III CONGRESSO BRASILEIRO DE REDES NEURAIS (CBRN'97). **Anais...** Florianópolis, SC, p. 408-413, 1997.

BOREINSTEIN, J. & KOREN, Y. The Vector field Histogram: Fast Obstacle Avoidance for Mobile Robots. **IEEE Journal of Robotics and Automation**. v. 7, p. 278-288, jun. de 1991.

DA COSTA, N. C. A.; ABE, J. M.; DA SILVA FILHO, J. I.; MUROLO, A. C.; LEITE, C. F. S. **Lógica Paraconsistente Aplicada.** São Paulo: Atlas, 1999.

DA COSTA, N. C. A.; ABE, J. M.; SUBRAHMANIAN, V. S. Remarks on Annotated Logic. **Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik**, v. 37, p.561-570, 1991.

DA COSTA, N. C. A.; PRADO, J. P. A.; ABE, J. M.; ÁVIVA, B. C.; RILLO, M. Paralog: Um Prolog Paraconsistente Baseado Em Lógica Anotada. Coleção Documentos Série **Lógica e Teoria da Ciência**, col. Documentos, IEA - USP, São Paulo, n.18, p.1 - 28, 1995.

DA SILVA FILHO, J. I. **Implementação de Circuitos Lógicos Fundamentados em Uma Classe de Lógicas Paraconsistentes Anotada**. 1997. 131 fl. Tese (Mestrado). Escola Politécnica da Universidade de São Paulo, São Paulo.

DA SILVA FILHO, J. I. **Métodos de aplicações da lógica paraconsistente anotada com anotação com dois valores LPA2v com construção de algoritmo e implementação de circuitos eletrônicos.** 1999. 115 fl. Tese (Doutorado). Escola Politécnica da Universidade de São Paulo, São Paulo.

DA SILVA FILHO, J. I. & ABE, J.M. Para-Fuzzy Logic Controller – Part I: A New Method of Hybrid Control Indicated for Treatment of Inconsistencies Designed with the Junction of

the Paraconsistent Logic and Fuzzy Logic. In: INTERNATIONAL ICSC CONGRESS ON COMPUTATIONAL INTELLIGENCE METHODS AND APPLICATIONS - CIMA'99. 1999. Rochester Institute of Technology, RIT, Rochester, N.Y., USA.

DA SILVA FILHO, J. I. & ABE, J. M. Para-Fuzzy Logic Controller – Part II: A Hybrid Logical Controlller Indicated for Treatment of Fuzziness and Inconsistencies. In: INTERNATIONAL ICSC CONGRESS ON COMPUTATIONAL INTELLIGENCE METHODS AND APPLICATIONS - CIMA'99. 1999b. Rochester Institute of Technology, RIT, Rochester, N.Y., USA.

DA SILVA FILHO, J. I. & ABE, J. M. Para-Analyser and Inconsistencies in Control Systems. In: IASTED INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING (ASC'99). **Anais...** Honolulu, Hawaii, USA, ago. 9-12, p. 78-85, 1999c.

DA SILVA FILHO, J. I. & ABE, J. M. Emmy: a paraconsistent autonomous mobile robot, in Logic, Artificial Intelligence, and Robotics. In: 2ND CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC'2001. In: ABE, J.M.; DA SILVA FILHO, J. I. [edts.] **Frontiers in Artificial Intelligence and Its Applications**. Amsterdam/Tokyo: IOS Press/Ohmsha, v. 71, p. 53-61, 287, 2001.

DA SILVA FILHO, J. I. & ABE, J. M. Paraconsistent electronic circuits. **International Journal of Computing Anticipatory Systems,** v. 9, p.337-345, 2001b.

DA SILVA FILHO, J. I. & ABE, J. M. **Fundamentos das Redes Neurais Paraconsistentes**: Destacando Aplicações em Neurocomputação. São Paulo: Arte & Ciência, 2001c.

DA SILVA FILHO, J.I. & ABE, J. M. Para-Control: An Analyser Circuit Based On Algorithm For Treatment of Inconsistencies. In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS, ISAS. SCI 2001, Vol. XVI,

Cybernetics and Informatics: Concepts and Applications (Part I), p. 199-203, Orlando, Florida, USA, 2001d.

DA SILVA FILHO, J. I., TORRES, C. R. & ABE, J. M. Robô Móvel Autônomo Emmy: Uma Aplicação Eficiente da Lógica Paraconsistente Anotada, **Seleção Documental**, ISSN 1809-0648, Número 3, Editora ParaLogike, Santos – São Paulo, pág. 19-26, Julho-Setembro/2006.

DA SILVA FILHO, J. I.; ABE, J. M.; LAMBERT-TORRES, G. **Inteligência Artificial com Redes de Análises Paraconsistentes:** Teoria e Aplicação. Rio de Janeiro: LTC, 2008.

DE MORAES, C. C. & CASTRUCCI, P. L. **Engenharia de Automação Industrial**. Rio de Janeiro: LTC, 2001.

DESIDERATO, J. M. G. & DE OLIVEIRA, E. N. **Primeiro Protótipo do Robô Móvel Autônomo Emmy II**I. 2006. Monografia (Trabalho de Conclusão de Curso). Universidade Metodista de São Paulo, São Bernardo do Campo, SP.

DU Xin; CHEN Hua-hua; GU Wei-kang. Neural network and genetic algorithm based global path planning in a static environment. **Journal of Zhejiang University SCIENCE**, 6A(6) p.549-554, 2005.

ELECTROLUX. **The Triobite 2.0.** Disponível em: <http://trilobite.electrolux.com/node217.asp>. Acesso em 15 de jan. de 2010.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Comp. Mag.**, v. 22, n, 6, p. 46-57, jun. de 1989.

FERRARA, L. F. P. **Redes neurais artificiais aplicada em um reconhecedor de caracteres. 2003**. Tese (Mestrado). Universidade Federal de Uberlândia - UFU, Uberlândia, MG.

HELPMATE ROBOTICS. **Robot Navigation Technology**. Disponível em: <http://statusre

ports.atp.nist.gov/reports/91-01-0034.htm>. Acesso em 15 de jan. de 2010.

KUBOTA, N.; HISAJIMA, D.; KOJIMA, F.; FUKUDA, T. Fuzzy and Neural Computing for Communication of a Partner Robot. **Journal of Multiple-Valued Logic and Soft Computing**, Philadelfia, USA, Old City Publishing, v. 9, Number 2, p. 221-239, 2003.

LEAL, B. R., DA SILVA, D. M. & VIEIRA D. B. **Segundo Protótipo do Robô Móvel Autônomo Emmy II**I. 2009. Monografia (Trabalho de Conclusão de Curso). Universidade Metodista de São Paulo, São Bernardo do Campo, SP.

MAIMONE, M.; MATTHIES, L.; OSBORN, J.; ROLLINS, E.; TEZA, J., & THAYER, S. A photo-realistic 3-D mapping system for extreme nuclear environments: Chernobyl. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTIC SYSTEMS (IROS'98). **Anais…** Victoria, Canada, out. de 1998.

MARAN, L. H. C.; RIBA, P. A.; COLLETT, R. G.; DE SOUZA, R. R. **Mapeamento de um ambiente não estruturado para orientação de um robô móvel autônomo utilizando redes neurais paraconsistente**. 2006. Monografia (Trabalho de Conclusão de Curso). Universidade Metodista de São Paulo, São Bernardo do Campo.

MARIO, M. C. **Proposta de aplicação das redes neurais artificiais paraconsistentes como classificador de sinais utilizando aproximação funcional**. 2003. Tese (Mestrado). Universidade Federal de Uberlândia - UFU, Uberlândia, MG.

MARIO, M. C. **Modelo de Análises de Variáveis Craniométricas através das Redes de Neurais Artificiais Paraconsistentes.** 2006. Tese (Doutorado) Faculdade de Medicina da Universidade de São Paulo – FMUSP, São Paulo.

MARIO, M. C. Sistema Classificador de Sinais Projetado com Redes Neurais Artificiais Paraconsistentes. **Revista Seleção Documental,** v2 n.7 jul./set. de 2007, p. 17-27, Ed. Paralogike Santos, São Paulo.

MARTINS, H. G. **A Lógica Paraconsistente Anotada de Quatro Valores**: LPA4v aplicada em Sistema de Raciocínio Baseado em Casos para o Restabelecimento de Subestações Elétricas. 2003. 145 fl. Tese (Ph D). Universidade Federal de Itajubá, UNIFEI, Itajubá.

MILLS, J. W.; WALKER, T.; HIMEBAUGH, B. Lukasiewicz' Insect: Continuous-Valued Robotic Control After Ten Years. **Journal of Multiple-Valued Logic and Soft Computing,** Philadelfia, USA, Old City Publishing, v. 9, n. 2, p. 131-146, 2003.

NAKAMATSU, K.; ABE, J.M.; SUZUKI, A. Defeasible Reasoning Between Conflicting Agents Based on VALPSN. AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE - AAAI'99, AAAI Press – American Association for Artificial Intelligence, Menlo Park, California, USA, p. 20-27, 1999.

NAKAMATSU, K.; ABE, J.M.; SUZUKI, A. Annotated Semantics for Defeasible Deontic Reasoning, Rough Sets and Current Trends in Computing, THE SECOND INTERNATIONAL CONFERENCE ON ROUGH SETS AND CURRENT TRENDS IN COMPUTING - RSCTC'2000. Banff, Canada, Lecture Notes in Artificial Intelligence series, LNAI 2005, Springer-Verlag, 470-478, 2000.

NEHMZW, Ulrich. **Mobile Robotics**: A Pratical Introduction. Nova Iorque: Springer-Verlag, 2000.

POLAROID, Manual do sensor de ultrassom com sistema POLAROID 500, 1996.

PRADO, J. P. A. **Uma Arquitetura em IA Baseada em Lógica Paraconsistente.** 1996. Tese (Doutorado). Escola Politécnica da Universidade de São Paulo.

PRASSLER, E.; RITTER, A.; SCHAEFFER, C.; FIORINI, P. A Short History of Cleaning Robots. **Journal of Autonomous Robots.** Springer Netherlands, p. 211-226, 2004.

ROSA E SILVA, S. **Aplicação da Lógica Paraconsistente Anotada no método de campos potenciais para navegação de robôs**. 2005. Tese (Mestrado). Engenharia de Produção da Universidade Paulista, São Paulo.

SCALZITTI, A.; DA SILVA FILHO, J. A.; ABE. A Formalization for Signal Analysis of Information in Annotated Paraconsistent Logics. In: 2ND CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC'2001. Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdan, v. 71, p.215-223, p. 286, 2001.

SHOVAL, S.; ULRICH, I.; BORENSTEIN, J. Robotics-Based Obstacle-Avoidance Systems for the Blind and Visually Impaired NavBelt and the GuideCane. IEEE ROBOTICS & AUTOMATION MAGAZINE, 9-20, mar. de 2003.

SYLVAN, R. & ABE, J. M. On general annotated logics, with an introduction to full accounting logics. **Bulletin of Symbolic Logic,** n. 2, p. 118-119, 1996.

SONY-AIBO. **Aibo, Entreteinment Robot AIBO**. Disponível em: <http://www.sonyaibo.net/home.htm>. Acesso em 15 de jan. de 2010.

SPAWAR, Space and Naval Warfare Systems Command. **Robart**. Disponível em: <http://www.spawar.navy.mil/robots/land/robart/robart.html>. Acesso em 13 de jan. de 2010.

SUBRAHMANIAN, V.S., On the Semantícs of Quantítatíve Logíc Programs, In: 4TH IEEE SYMPOSIUM ON LOGIC PROGRAMMING, COMPUTER SOCIETY PRESS. **Anais…** Washington D.C. 1987, p 173-182.

TORRES, C. R. **Sistema Inteligente Paraconsistente para Controle de Robôs Móveis Autônomos.** 2004. 85 fl. Tese (Mestrado). Universidade Federal de Itajubá - UNIFEI, Itajubá.

TORRES, C. R.; ABE, J. M.; LAMBERT-TORRES, G. Sistema Inteligente para Controle de Robôs Móveis Autônomos. I WORKSHOP UNIVERSIDADE–EMPRESA EM AUTOMAÇÃO, ENERGIA E MATERIAIS. **Anais...** Taubaté, São Paulo, 2004.

TORRES, C. R.; ABE, J. M.; LAMBERT-TORRES, G. Robô Móvel Emmy II. **Coleção Documentos**, s. Lógica e Teoria da Ciência, IEA-USP, n. 58, p. 1-14, 2005.

TORRES, C. R., ABE, J. M. & LAMBERT-TORRES, G. Sistema Inteligente para Controle de Robôs Móveis Autônomos, Coleção Documentos, Série Lógica e Teoria da Ciência, IEA-USP, ISSN 16799429, Número 58, pág. 15-28, 2005b.

TORRES, C. R., ABE, J. M.; LAMBERT-TORRES, G. Robô Móvel Autônomo Emmy II, **Seleção Documental**, n. 4, ed. ParaLogike, Santos, São Paulo, p. 5-10, out./dez. de 2006.

TORRES, C. R.; LAMBERT-TORRES, G.; SILVA, L. E. B. da; ABE, J. M. Intelligent System of Paraconsistent Logic to Control Autonomous Moving Robots. In: 32ND ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY. Paris: IEEE Press, 2006.

TORRES, C. R., ABE, J. M., LAMBERT-TORRES, G. Sistema Inteligente Paraconsistente para Controle de Robôs Móveis Autonômos. **Seleção Documental**, ISSN 1809-0648, Número. 7, Editora ParaLogike, Santos – São Paulo, pág. 13-18, Julho-Setembro/2007.

TORRES, C. R. & BOMBACINI M. R. Robô Autônomo com Sistema de Controle microprocessado e Fundamentado em Lógica Paraconsistente. **Seleção Documental,** n. 5, ed. ParaLogike, Santos, São Paulo, p. 14-18, jan./mar. de 2007.

TORRES, C. R.; ABE, J. M.; LAMBERT-TORRES, G.; DA SILVA FILHO, J. I., MARTINS, H. G. Autonomous Mobile Robot Emmy III. In: Nakamatsu, K.; Phillips-Wren,

G.; Jain, L.C.; Howlett, R. J.[Orgs.]. **New Advances in Intelligent Decision Technologies,** 1ª ed. Helderberg: Springer-Verlag, 2009, v. 199, p. 317-327.

ANEXOS

Anexo 1

Mostra-se neste anexo o programa em Linguagem Python do Subsistema de Sensoriamento.

```
# -*- coding: cp1252 -*-

from models import CNAPpa, CNAPa

import psycopg2
import datetime
import math


conn = psycopg2.connect("\
   dbname='teste'\
   user='postgres'\
   host='localhost'\
   password='postgres'\
");


c = conn.cursor()
data_atual = "'" + str(datetime.datetime.now()) + "'"


# Módulo de Sensoriamento - Modelo Unidade do Grau de Evidência da Célula.
class UGEC:

   """
   Uma instância dessa classe representa uma Unidade do Grau de Evidência da Célula.
   """

   m1 = []                 # Distância do obstáculo ao sensor.
   m2 = None               # Grau de evidência sobre o arco AB e AC
   m3 = None               # Grau de evidência atual da célula analisada.
   x = None                # Coordenada x da célula
   y = None                # Coordenada y da célula
   camadas = []            # Camadas da RNAP
   ft = []                 # Fatores de tolerância a certeza e contradição.
   retorno = None          # Saída da RNAP


   # Recebe os graus de evidência.
   def __init__(self, m1, m2, m3, x, y, ftcc1, ftcc2, ftcc3, ftctc4, ftcc4, ftctc5, ftcc5):
      self.m1 = m1
      self.m2 = m2
      self.m3 = m3
      self.x = x
      self.y = y
```

```
        self.ft = []

        self.ft.append(ftcc1)
        self.ft.append(ftcc2)
        self.ft.append(ftcc3)
        self.ft.append(ftctc4)
        self.ft.append(ftcc4)
        self.ft.append(ftctc5)
        self.ft.append(ftcc5)


    # Cria as Células Neurais Artificiais Paraconsistentes.
    def cria_camadas(self):
        self.camadas = []
        c1 = CNAPpa(m1=self.m1, c2=self.ft[0])
        c2 = CNAPpa(m1=self.m2, c2=self.ft[1])
        c3 = CNAPpa(m1=self.m3, c2=self.ft[2])
        c4 = CNAPa(c1.saida(), c2.saida(), self.ft[3], self.ft[4])
        c5 = CNAPa(c4.saida()[0], c3.saida(), self.ft[5], self.ft[6])

        self.camadas.append([c1, c2])
        self.camadas.append([c4, c3])
        self.camadas.append([c5])


    # Retorna a os valores das Células Neurais Artificiais Paraconsistentes da Unidade.
    def saida(self):
        self.cria_camadas()
        s = self.camadas[-1][0].saida()[0]
        self.retorno = round(s,3)
        return s



# Módulo Sensoriamento - Modelo da Unidade de Geração de Dados para a Rede.
class UGDR:

    """
    Uma instância dessa classe representa uma Unidade de Geração de dados para a Rede.
    """

    a = None                        # Distância entre as coordenadas.
    beta = None                      # Ângulo de abertura do sensor.
    d = None                        # Distância entre o obstáculo e o sensor.
    n = None                        # Número de células no arco de abertura do sensor.
    dmax = None                        # Distância máxima medida pelo sensor.
    dmin = None                       # Distância mínima medida pelo sensor.
    alfa = None                      # Ângulo do sensor em relação ao eixo x.
    xa = None                        # Coordenada x atual (onde o robô se encontra).
    ya = None                        # Coordenada y atual (onde o robô se encontra).
    coordenada = []                     # Lista das coordenadas das possíveis células com obstáculo.
    rede = []                       # RNAP de cada coordenada.
    sair = None                       # Determina o final do algoritmo.


    # Recebe as configurações do ambiente e do sensor.
    def __init__(self, a=None, beta=None, n=None, dmax=None, dmin=None):
        print '\n\n***** Configurações Iniciais do Algoritmo ****'
        if not a:
```

```
        self.a = self.valida_entrada("Distância entre coordenadas: ")
    if not beta:
        self.beta = self.valida_entrada("Ângulo de abertura do sensor: ")
    if not n:
        self.n = self.valida_entrada("Número de células no arco de abertura do sensor: ")
    if not dmax:
        self.dmax = self.valida_entrada("Distância máxima medida pelo sensor: ")
    if not dmin:
        self.dmin = self.valida_entrada("Distância mínima medida pelo sensor: ")


# Recebe uma variável de entrada do usuário e verifica se o valor está dentro do intervalo [0,1]
def valida_entrada(self, msg):
    entrada = None
    msg_erro = '\nDigite um número maior ou igual a 0'
    while not entrada or float(entrada) < 0:
        try:
            if entrada:
                print msg_erro
            entrada = raw_input(msg)
            float(entrada)
        except:
            print msg_erro
    return float(entrada)


# Busca informações
def busca_dados(self):
    print '\n\n***** Leitura dos dados do Sensor *****'
    self.sair = raw_input('Ler o Sensor (S/N): ')
    if self.sair.lower() == 'n':
        pass
    else:
        self.coordenada = []
        self.d = self.valida_entrada("Distância entre o obstáculo e o sensor: ")
        self.alfa = self.valida_entrada("Ângulo do sensor em relação ao eixo x: ")
        self.xa = self.valida_entrada("Coordenada x atual (onde o robô se encontra): ")
        self.ya = self.valida_entrada("Coordenada y atual (onde o robô se encontra): ")

        if self.d >= self.dmax: self.d = self.dmax
        if self.d <= self.dmin: self.d = self.dmin


# Faz ajustes na projeção.
def ajuste_projecao(self, v):
    (i,d) = str(v).split('.')

    if float(d) < 1./2:
        return int(i)
    return int(i) + 1


# Transforna ângulo em graus para radianos
def transforma_grau_radiano(self, grau):
    return grau * 2 * math.pi / 360.0


# Calcula a projeção no eixo x da coordenada central.
def calcula_projecao_central_x(self):
    grau = self.alfa
```

```
        r = (self.d * (math.cos(self.transforma_grau_radiano(grau))))
        return r


    # Calcula a projeção no eixo y da coordenada central.
    def calcula_projecao_central_y(self):
        grau = self.alfa
        r = (self.d * (math.sin(self.transforma_grau_radiano(grau))))
        return r


    # Calcula a projeção no eixo x das células a direita da coordenada central.
    def calcula_projecao_direita_x(self, b, i):
        grau = self.alfa-(b * i)
        r = (self.d * (math.cos(self.transforma_grau_radiano(grau))))
        return r


    # Calcula a projeção no eixo y das células a direita da coordenada central.
    def calcula_projecao_direita_y(self, b, i):
        grau = self.alfa-(b * i)
        r = (self.d * (math.sin(self.transforma_grau_radiano(grau))))
        return r


    # Calcula a projeção no eixo x das células a esquerda da coordenada central.
    def calcula_projecao_esquerda_x(self, b, i):
        grau = self.alfa+(b * i)
        r = (self.d * (math.cos(self.transforma_grau_radiano(grau))))
        return r


    # Calcula a projeção no eixo y das células a esquerda da coordenada central.
    def calcula_projecao_esquerda_y(self, b, i):
        grau = self.alfa+(b * i)
        r = (self.d * (math.sin(self.transforma_grau_radiano(grau))))
        return r


    # Cálculo da cordenada x ou y.
    def calcula_coordenada(self, p, k):                 # k = self.xa ou self.ya, p = projeção x ou projeção y
        c = (p/self.a) + k
#       return c                                    # Retorna x ou y
        return self.ajuste_projecao(c)                  # Retorna x ou y


    # Gera os graus de evidência da célula central.
    def grava_celula_central(self):
        self.busca_dados()
        if self.sair.lower() == 'n':
            pass
        else:
            m1 = 1 - (self.d/self.dmax)
            m2 = 1
            px = self.calcula_projecao_central_x()
            py = self.calcula_projecao_central_y()
            x = self.calcula_coordenada(px, self.xa)
            y = self.calcula_coordenada(py, self.ya)

            # Verificar data do BD antes de atribuir o valor de m3.
```

```
            m3 = 0
            c.execute('SELECT evidencia, data FROM matriz WHERE x=%s AND y=%s' % (x, y))
            r = c.fetchall()

            if r:
               ge = r[0][0]
               data_antiga = r[0][1]

               hoje = datetime.datetime.now()

               d = hoje - data_antiga
               if (d.days == 0) and (d.seconds / 60) < 60:
                  m3 = ge

            self.coordenada.append([x,y,m1,m2,m3])


      # Grava na lista de coordenadas
      def grava_coordenadas(self, x, y, a):
         m1 = a[2]
         m2 = a[3] - (1./self.n)


         # Verificar data do BD antes de atribuir o valor de m3.
         m3 = 0
         c.execute('SELECT evidencia, data FROM matriz WHERE x=%s AND y=%s' % (x, y))
         r = c.fetchall()

         if r:
            ge = r[0][0]
            data_antiga = r[0][1]

            hoje = datetime.datetime.now()

            d = hoje - data_antiga
            if (d.days == 0) and (d.seconds / 60) < 60:
               m3 = ge
#           print r
#       print 'm3 = %s' % m3
         self.coordenada.append([x, y, m1, m2, m3])


      # Gera a lista de saída com todas as coordenadas identificadas pelo sensor.
      def gera_coordenadas_direita_esquerda(self):
         self.grava_celula_central()

         if self.sair.lower() == 'n':
            pass
         else:
            b = self.beta/self.n
            i = 1
            while i <= self.n:
               # Calcula coordenadas da direita.
               pxd = self.calcula_projecao_direita_x(b, i)
               pyd = self.calcula_projecao_direita_y(b, i)
               xd = self.calcula_coordenada(pxd, self.xa)
               yd = self.calcula_coordenada(pyd, self.ya)

               if i == 1:
                  a = self.coordenada[-1]                    # última coordenada gravada na variável self.coordenada
```

```
            else:
                a = self.coordenada[-2]

            self.grava_coordenadas(xd, yd, a)

            # Calcula coordenadas da esquerda.
            a = self.coordenada[-2]

            pxe = self.calcula_projecao_esquerda_x(b, i)
            pye = self.calcula_projecao_esquerda_y(b, i)
            xe = self.calcula_coordenada(pxe, self.xa)
            ye = self.calcula_coordenada(pye, self.ya)

            self.grava_coordenadas(xe, ye, a)

            i = i + 1


    # Atualização da base de dados (executa a RNAP).
    def atualiza_bd(self, ftcc1, ftcc2, ftcc3, ftctc4, ftcc4, ftctc5, ftcc5):
        self.rede = []
        for co in self.coordenada:
            rede = UGEC(co[2], co[3], co[4], co[0], co[1], float(ftcc1), float(ftcc2), float(ftcc3), float(ftctc4),
float(ftcc4), float(ftctc5), float(ftcc5))
            s = round(rede.saida(),3)
            self.rede.append(rede)

            # Grava na base de dados a coordenada e seus graus de evidência.
            c.execute('SELECT id FROM matriz WHERE x=%s AND y=%s' % (co[0], co[1]))
            r = c.fetchall()

            if r:
                id = r[0][0]
                c.execute('UPDATE matriz SET evidencia=%s, data=%s WHERE id=%s' % (s, data_atual, id))
                conn.commit()
            else:
                c.execute('INSERT INTO matriz (x, y, evidencia, data) VALUES (%s,%s,%s,%s)' % (co[0], co[1], s,
data_atual))
                conn.commit()


    # Imprime as coordenadas resultantes da leitura do sensor
    def imprime_coordenadas(self):
        d = []
        e = []

        i = 1
        while i < len(self.coordenada):
            co = self.coordenada[i]
            r = self.rede[i].retorno
            s = 'Coordenada (%s,%s): μ1 = %s | μ2 = %s | μ3 = %s | Saída da RNAP = %s' % (co[0], co[1], co[2],
co[3], co[4], r)

            if (i % 2) == 0: e.append(s)                    # Coordenadas da esquerda
            else: d.append(s)                           # Coordenadas da direita
            i = i + 1

        co = self.coordenada[0]
        r = self.rede[0].retorno
```

```
        print '\n\n***** Coordenada Central ***** \nCoordenada (%s,%s): \nμ1 = %s | μ2 = %s | μ3 = %s| Saída
da RNAP = %s' % (co[0], co[1], co[2], co[3], co[4], r)

        if d:
            print '\n\n***** Coordenadas da Direita *****\n'
            for c in d:
                print c

        if e:
            print '\n\n***** Coordenadas da Esquerda *****\n'
            for c in e:
                print c


    # Executa o algoritmo.
    def executa_algoritmo(self):
        print '\n\n***** Configurações da RNAP *****'
        print '>>> Primeira Camada'
        ftcc1 = self.valida_entrada("Fator de Tolerância à Certeza da 1ª CNAPpa: ")
        ftcc2 = self.valida_entrada("Fator de Tolerância à Certeza da 2ª CNAPpa: ")
        ftcc3 = self.valida_entrada("Fator de Tolerância à Certeza da 3ª CNAPpa: ")
        print '>>> Segunda Camada'
        ftctc4 = self.valida_entrada("Fator de Tolerância à Contradição da 1ª CNAPa: ")
        ftcc4 = self.valida_entrada("Fator de Tolerância à Certeza da 1ª CNAPa: ")
        print '>>> Terceira Camada'
        ftctc5 = self.valida_entrada("Fator de Tolerância à Contradição da 2ª CNAPa: ")
        ftcc5 = self.valida_entrada("Fator de Tolerância à Certeza da 2ª CNAPa: ")

        while self.sair == None or self.sair.lower() != 'n':
            self.gera_coordenadas_direita_esquerda()

            if self.sair != None and self.sair != 'n':
                self.atualiza_bd(ftcc1, ftcc2, ftcc3, ftctc4, ftcc4, ftctc5, ftcc5)
                self.imprime_coordenadas()
```

Anexo 2

Apresenta-se aqui o algoritmo completo do Subsistema de Planejamento.

```
Bloco Verifica_Condicao(c, mlim)
Enquanto (c < mlim) E (m ≠ 0) faça:
        Executa Bloco execute_movimento(m)
        m       ← Executa Bloco movimento()
        c       ← Executa Bloco condições(m) – entra apenas o grau de evidência


Bloco Verifica_Movimento_Atras(m, n)
Se (m+n) > 8 então:
        Executa Bloco execute_movimento (m+n-8)
Se não:
        Executa Bloco execute_movimento (m+n)


Bloco Planejamento:
Mlim    ← Entra com valor limite de evidência nas células
Xo      ← Entra com a coordenada x de origem
Yo      ← Entra com a coordenada y de origem
Xd      ← Entra com a coordenada x de destino
Yd      ← Entra com a coordenada y de destino
p       ← Entra com o número de células a frente do robô

m       ← Executa Bloco Movimento()
c       ← Executa Bloco Condicoes(m)[0]
n       ← 0

Enquanto m ≠ 0 faça:
        Executa Bloco Verifica_Condicao(c, mlim)

        Se m ≠ 0 então:
                n       ← n + 1

                Se n = 4 então:
                Se (m + n) > 8 então:
                        c ← Executa Bloco Condicoes(m + n – 8) – entra apenas o grau de evidência
                Se não:
                        c ← Executa Bloco Condicoes(m + n) – entra apenas o grau de evidência

                Se c > mlim então:
                                retorno ← Executa Bloco Beco_Sem_Saida
                        Imprime "Executou o Bloco Beco_Sem_Saida(m+n)"
                                Se retorno = 1então:
                                Finaliza o algoritmo
                                n       ← 0
                Se não:
                        Executa Bloco Verifica_Movimento_Atras(m,n)
                        n       ← 1
                m       ← Executa Bloco Movimento()

                Se (m+n) > 8 então:
                c_hor   ← Executa Bloco Condicoes (m+n-8) – entra apenas o grau de evidência
```

```
Se não:
c_hor   ← Executa Bloco Condicoes (m+n) – entra apenas o grau de evidência


Se (m-n) < 1 então:
c_ant   ← Executa Bloco Condicoes (m-n+8) – entra apenas o grau de evidência

Se não:
c_ant   ← Executa Bloco Condicoes (m-n) – entra apenas o grau de evidência




Se (c_hor < mlim) OU (c_ant < mlim) então:
        Se (c_hor ≠ c_ant) E [(c_hor >= mlim) OU (c_ant >= mlim)] então:
                Se c_hor >= mlim então:
                        Se (m-n) < 1 então:
                                Executa Bloco execute_movimento (m-n+8)
                        Se não:
                                Executa Bloco execute_movimento (m-n)
                Se não:
                        Se (m+n) > 8 então:
                                Executa Bloco execute_movimento (m+n-8)
                        Se não:
                                Executa Bloco execute_movimento (m+n)
        Se não:
                Se (m+n) > 8 então:
                        Mimed_hor       ← Executa Bloco mimed (m+n-8)
                Se não:
                        Mimed_hor       ← Executa Bloco mimed (m+n)

                Se (m-n) < 1 então:
                        Mimed_ant       ← Executa Bloco mimed (m-n+8)
                Se não:
                        Mimed_ant       ← Executa Bloco mimed (m-n)

                Se Mimed_hor ≠ Mimed_ant então:
                        Se Mimed_hor > Mimed_ant então:
                                Se (m-n) < 1 então:
                                        Executa Bloco execute_movimento (m-n+8)
                                Se não:
                                        Executa Bloco execute_movimento (m-n)
                        Se não:
                        Se (m+n) > 8 então:
                                        Executa Bloco execute_movimento (m+n-8)
                                Senão:
                                        Executa Bloco execute_movimento (m+n)




                Se não:
                Se (m+n) > 8 então:
                                (Xm, Ym)← Executa Bloco Destino_Futuro (m+n-8)
                        Se não:
                        (Xm, Ym)← Executa Bloco Destino_Futuro (m+n)

                        am      ← |Xm – Xo|
                        bm      ← |Ym – Yo|
```

dm ← $\sqrt{am^2 + bm^2}$

Se (m-n) < 1 então:
(Xn, Yn)← Executa Bloco Destino_Futuro (m-n+8)
Se não:
(Xn, Yn)← Executa Bloco Destino_Futuro (m-n)

an ← |Xn – Xo|
bn ← |Yn – Yo|
dn ← $\sqrt{an^2 + bn^2}$

Se dm <= dn então:
Se (m+n) > 8 então
Executa Bloco execute_movimento (m+n-8)
Se não:
Executa Bloco execute_movimento (m+n)
Se não:
Se (m-n) < 1 então:
Executa Bloco execute_movimento (m-n+8)
Se não:
Executa Bloco execute_movimento (m-n)

m ← Executa Bloco Movimento( )
c ← Executa Bloco Condicoes(m) – entra apenas o grau de evidência

Apaga todos os registros da tabela ‘rastro’ do BD.

```
Bloco Movimento ( )
        Se Yd >= Yo então:
                Se Xd = Xo então:
                        Se Yd = Yo então:
                                m ← 0
                        Senão:
                                m ← 1
                Senão:
                        Se Yd = Yo então:
                                Se Xd > Xo então:
                                        m ← 3
                                Senão:
                                        m ← 7
                        Senão:
                                Se Xd > Xo então:
                                        m ← 2
                                Senão:
                                        m ← 8
        Senão:
                Se Xd = Xo então:
                        m ← 5
                Senão:
                        Se Xd < Xo então:
                                m ← 6
                        Senão :
                                m ← 4
        Retorna m
```

```
Bloco execute_movimento (m)
        Mostre a mensagem 'Movimento: m'

        f       ← mimed(m)
        Mostre a mensagem 'Grau de Evidência da Direção: f'

        (ge, i)  ← Condicoes(m)
        Mostre a mensagem 'Grau de Evidência Máximo da Direção: ge'
        Mostre a mensagem 'Distância da célula onde se encontra o grau de evidência Máximo: i'

        Adiciona o a coordenada (Xd,Yd) com grau de evidência 1 na tabela 'rastro' do BD.

        (Xd, Yd)        ← Executa Bloco Destino_Futuro (m)

        n ← 0


Bloco Destino_Futuro(m)
        Se m = 1 então:
                x ← Xd
                y ← Yd -1
        Senão:
                Se m = 2 então:
                        x ← Xd – 1
                        y ← Yd – 1
                Senão:
                        Se m = 3 então:
                                x ← Xd – 1
                                y ← Yd
                        Senão:
                                Se m = 4 então:
                                        x ← Xd – 1
                                        y ← Yd + 1
                                Senão:
                                        Se m = 5 então:
                                                x ← Xd
                                                y ← Yd + 1
                                        Senão:
                                                Se m = 6 então:
                                                        x ← Xd + 1
                                                        y ← Yd +1
                                                Senão:
                                                        Se m = 7 então:
                                                                x ← Xd + 1
                                                                y ← Yd
                                                        Senão:
                                                                x ← Xd + 1
                                                                y ← Yd - 1
        Retorna x, y
```

```
Bloco Coordenadas(m)

        i ← 0
        (x, y)    ← Executa Bloco Destino_Futuro(m)
        evid      ← Lista vazia
        coordenada← Lista vazia


        Enquanto i < p faça:
                rastro ← Consulta na tabela 'rastro' do BD o grau de evidência de (x,y).
                evid[i]← Adiciona na lista o valor do grau de evidência de (x,y) consultado na tabela 'matriz'
do BD
                coordenada[i]← Adiciona na lista a coordenada (x,y) consultada na tabela 'matriz' do BD

                Se n ≠ 4 e rastro = 1 então: "Quando n=4 o robô deve ir para tras. Neste caso não se deve
        consultar o rastro"
                        evid[i] = 1

                Se m = 1 então:
                        y ← y – 1
                Senão:
                        Se m = 2 então:
                                x ← x –1
                                y ← y – 1
                        Senão:
                                Se m = 3 então:
                                        x ← x –1
                                Senão:
                                        Se m = 4 então:
                                                x ← x –1
                                                y ← y + 1
                                        Senão:
                                                Se m = 5 então:
                                                y ← y + 1
                                                Senão:
                                                        Se m = 6 então
                                                                x ← x +1
                                                                y ← y + 1
                                                        Senão:
                                                        Se m = 7 então:
                                                                x ← x +1
                                                                Senão:
                                                                        x ← x +1
                                                                        y ← y - 1
                i ← i + 1
```

```
Bloco Condicoes(m)
        evid      ← Executa Bloco Coordenadas(m)

        ge ← Executa Bloco Rede_Maximizacao(evid)

        j ← 0
        Enquanto j < p faça:
        Se evid[j] = ge então:
                i ← j
        j ← j + 1

        Retorna ge, i


Bloco Rede_Maximizacao(evid)
        celulas   ← Lista vazia

        p1 = evid[0]
        p2 = evid[1]
        celulas[0] ← Cria CNAPCls de Maximização com os parâmetros p1 e p2

        i ← 2
        j ← 1
        Enquanto i < p faça:
        p1← celulas[-1]
        p2 ← evid[i]
                celulas[j] ← Cria CNAPCls de Maximização com os parâmetros p1 e p2
        i ← i + 1
        j ← j + 1

        Retorna celulas[-1]
```

```
Bloco mimed(m)
        evid      ← Executa Bloco Coordenadas(m)

        ge ← Executa Bloco Rede_Media(evid)

        Retorna ge


Bloco Rede_Media(m)
        celulas   ← Lista vazia

        p1 = evid[0]
        p2 = evid[1]
        celulas[0] ← Cria CNAPa com os parâmetros p1 e p2

        i ← 2
        j ← 1
        Enquanto i < p faça:
        p1← celulas[-1]
        p2 ← evid[i]
                celulas[j] ← Cria CNAPCls de Maximização com os parâmetros p1 e p2
        i ← i + 1
        j ← j + 1

        Retorna celulas[-1]



Bloco Beco_Sem_Saida(m)
        b ← 1 (Estamos usando b porque o n = 4. E o n = 4 é importante no Bloco Coordenadas)
        c_hor ← 0
        c_ant ← 0
        retorno ← 0
        a ← 1


        Enquanto a = 1 faça:
        Se b > 4 então:
                Imprime "BECO SEM SAÍDA"
                retorno = 1
                        Retorna retorno (significa que volta para onde chamou retornando o valor 1)

                Se (m + b) > 8 então:
                c_hor ← Executa Bloco Condicoes(m + b – 8) – entra apenas o grau de evidência
                Senão:
                c_hor ← Executa Bloco Condicoes(m + b) – entra apenas o grau de evidência
                Se (c_hor < mlim) então:
                Apaga os registros das coordenadas da lista "coordenada" do BD rastro (A lista "coordenada"
está localizada no Bloco Coordenadas)
                a ← 0
                Se (m-b) < 1 então:
                c_ant ← Executa Bloco Condicoes(m-b+8) – entra apenas o grau de evidência
        Senão:
                        c_ant ← Executa Bloco Condicoes(m-b) – entra apenas o grau de evidência


                Se (c_ant < mlim) então:
                Apaga os registros das coordenadas da lista "coordenada" do BD rastro (A lista "coordenada"
        está localizada no Bloco Coordenadas)
                        a ← 0
```

b ← b + 1

Retorna retorno (significa que volta para onde chamou retornando o valor 0)

Anexo 3

Mostra-se neste anexo o programa em Linguagem Python do Subsistema de Planejamento.

```
from models import CNAPCls, CNAPa

import psycopg2
import datetime
import math
from decimal import Decimal


conn = psycopg2.connect("\
   dbname='teste'\
   user='postgres'\
   host='localhost'\
   password='postgres'\
");


c = conn.cursor()


# Módulo de Planejamento.
class ModuloPlanejamento:

   """
   Uma instância dessa classe representa o módulo de planejamento do Robô.
   """

   mlim = None             # Valor limite de evidência nas células.
   x0 = None               # Coordenada x de origem.
   y0 = None               # Coordenada y de origem.
   xd = None               # Coordenada x de destino.
   yd = None               # Coordenada y dde destino.
   p = None                # Número de células a frente do Robô.
   m = None                # Movimento atual.
   c = None                # Condições.
   n = None                # Direção do movimento.
   evid = []               # Graus de evidência das p células a frente do Robô.
```

```
    coordenada = []            # Coordenadas p células a frente do Robô.
    celulas_max = []           # CNAPCls de Maximização (Grau de evidência máximo das p células a frente do Robô)
    celulas_med = []           # CNAPa (Grua de evidência médio das p células a frente do Robô)
    saida = []                 # Saída do algoritmo impressa no arquivo



    # Recebe as configurações iniciais do ambiente.
    def __init__(self, mlim=None, x0=None, y0=None, xd=None, yd=None, p=None):
        print '\n\n***** Configurações Iniciais do Algoritmo ****'

        self.mlim = self.valida_entrada("Valor Limite de Evidência nas Células: ")

        self.x0 = self.valida_entrada("Coordenada x de origem: ")

        self.y0 = self.valida_entrada("Coordenada y de origem: ")

        self.xd = self.valida_entrada("Coordenada x de destino: ")

        self.yd = self.valida_entrada("Coordenada y de destino: ")

        self.p = self.valida_entrada("Número de células a frente do Robô: ")

        self.m = self.movimento()

        self.c = self.condicoes(self.m)[0]

        self.saida.append([self.x0, self.y0])


    # Recebe uma variável de entrada do usuário e verifica se o valor é um número válido.
    def valida_entrada(self, msg, e=None):
        entrada = None
        msg_erro = '\nDigite um número'
        while True:
            try:
                if e:
                    entrada = e
                else:
```

```
                entrada = raw_input(msg)
            float(entrada)
            break
        except:
            print msg_erro
            entrada = raw_input(msg)
    return float(entrada)


# Algoritmo Principal do Módulo de Planejamento.
def planejamento(self):
    self.n = 0
    while self.m != 0:
        self.verifica_condicao()
        if self.m != 0:
            self.n = self.n + 1
#             print 'n: %s (antes da condição de n =4)' % self.n
            if self.n == 4:
                if (self.m + self.n) > 8:
                    self.c = self.condicoes(self.m + self.n - 8)[0]
                else:
                    self.c = self.condicoes(self.m + self.n)[0]

                if self.c > self.mlim:
                    print "\nExecutou o Bloco Beco_Sem_Saída"
                    retorno = self.beco_sem_saida(self.m + self.n)
                    if retorno == 1:
                        break
                    else:
                        print 'RETORNOU'
                    n = 0
                else:
                    self.verifica_movimento_atras()
                    self.n = 1

#                 self.verifica_movimento_atras()
#                 print 'n: %s' % self.n
                self.m = self.movimento()
```

```
            if (self.m + self.n) > 8:
                c_hor = self.condicoes(self.m + self.n - 8)[0]
            else:
                c_hor = self.condicoes(self.m + self.n)[0]

            if (self.m - self.n) < 1:
                c_ant = self.condicoes(self.m - self.n + 8)[0]
            else:
                c_ant = self.condicoes(self.m - self.n)[0]

            if (c_hor < self.mlim) or (c_ant < self.mlim):
#                print 'M: %s' % self.m
                if c_hor != c_ant and (c_hor >= self.mlim or c_ant >= self.mlim):
                    if c_hor >= self.mlim:
                        if (self.m - self.n) < 1:
                            self.execute_movimento(self.m - self.n + 8)
                        else:
                            self.execute_movimento(self.m - self.n)
                    elif (self.m + self.n) > 8:
                        self.execute_movimento(self.m + self.n - 8)
                    else:
                        self.execute_movimento(self.m + self.n)
                else:
                    if (self.m + self.n) > 8:
                        mimed_hor = self.mimed(self.m + self.n - 8)
                    else:
                        mimed_hor = self.mimed(self.m + self.n)

                    if (self.m - self.n) < 1:
                        mimed_ant = self.mimed(self.m - self.n + 8)
                    else:
                        mimed_ant = self.mimed(self.m - self.n)

                    if mimed_hor != mimed_ant:
                        if mimed_hor > mimed_ant:
                            if (self.m - self.n) < 1:
                                self.execute_movimento(self.m - self.n + 8)
                            else:
                                self.execute_movimento(self.m - self.n)
                        elif (self.m + self.n) > 8:
```

```
                    self.execute_movimento(self.m + self.n - 8)
                else:
                    self.execute_movimento(self.m + self.n)
            else:
                if (self.m + self.n) > 8:
                    (xm, ym) = self.destino_futuro(self.m + self.n - 8)
                else:
                    (xm, ym) = self.destino_futuro(self.m + self.n)

                am = abs(xm - self.x0)
                bm = abs(ym - self.y0)
                dm = math.sqrt((am**2 + bm**2))

                if (self.m - self.n) < 1:
                    (xn, yn) = self.destino_futuro(self.m - self.n + 8)
                else:
                    (xn, yn) = self.destino_futuro(self.m - self.n)

                an = abs(xn - self.x0)
                bn = abs(yn - self.y0)
                dn = math.sqrt((an**2 + bn**2))

                if dm <= dn:
                    if (self.m + self.n) > 8:
                        self.execute_movimento(self.m + self.n - 8)
                    else:
                        self.execute_movimento(self.m + self.n)
#                       print 'Condição 2.3.1'
                else:
                    if (self.m - self.n) < 1:
                        self.execute_movimento(self.m - self.n + 8)
#                           print 'Condição 2.3.2'
                    else:
                        self.execute_movimento(self.m - self.n)
#                           print 'Condição 2.3.3'

#                   print 'Condição 2.3'

#                   print 'dm = %s' % dm
#                   print 'dn = %s' % dn
```

```
#                       print 'Condição 2'

 #                print 'Condição no sentido horário: %s' % c_hor
#                 print 'Condição no sentido anti-horário: %s' % c_ant

              self.m = self.movimento()
              self.c = self.condicoes(self.m)[0]
#           print 'Xd = %s' % self.xd
#           print 'Yd = %s' % self.yd
#           print 'm = %s' % self.m
#           print 'c = %s' % self.c
        c.execute('DELETE FROM rastro')
        conn.commit()


        self.gera_arquivo()                         # Gera arquivo XML

        arq = open('saida_simples.xml', 'r')
        conteudo = arq.read()
        arq.close()
        print '\n\n*** Conteúdo do Arquivo XML gerado para o Planejador Simples ***\n'
        print conteudo


    # Gera arquivo XML com as informações da variável saída.
    def gera_arquivo(self):
        x = str(self.saida[0][0])
        y = str(self.saida[0][1])
        movimento = str(self.saida[1:])
        movimento = movimento[1:len(movimento)-1]

        arq = file('saida_simples.xml','w')                         # Abrir arquivo para gravação
        xml = '''\
        <saida>
        <x_origem>%s</x_origem>
        <y_origem>%s</y_origem>
        <movimento>%s</movimento>
        </saida>
        '''
```

```
    saida = (x, y, movimento)
    arq.write(xml % saida)
    arq.close()

    return arq


def verifica_condicao(self):
    while (self.c < self.mlim) and (self.m != 0):
        self.execute_movimento(self.m)
        self.m = self.movimento()
        self.c = self.condicoes(self.m)[0]


def verifica_movimento_atras(self):
    if (self.m + self.n) > 8:
        self.execute_movimento(self.m + self.n - 8)
    else:
        self.execute_movimento(self.m + self.n)


def movimento(self):
    if self.yd >= self.y0:
        if self.xd == self.x0:
            if self.yd == self.y0: m = 0
            else: m = 1
        elif self.yd == self.y0:
            if self.xd > self.x0: m = 3
            else: m = 7
        elif self.xd > self.x0: m = 2
        else: m = 8
    else:
        if self.xd == self.x0: m = 5
        elif self.xd < self.x0: m = 6
        else: m = 4

    return m


def destino_futuro(self, m):
```

```
        if m == 1:
            x = self.xd
            y = self.yd - 1
        elif m == 2:
            x = self.xd - 1
            y = self.yd - 1
        elif m == 3:
            x = self.xd - 1
            y = self.yd
        elif m == 4:
            x = self.xd - 1
            y = self.yd + 1
        elif m == 5:
            x = self.xd
            y = self.yd + 1
        elif m == 6:
            x = self.xd + 1
            y = self.yd + 1
        elif m == 7:
            x = self.xd + 1
            y = self.yd
        else:
            x = self.xd + 1
            y = self.yd - 1

        return x, y


    def execute_movimento(self, m):
#       self.m = m                  # Atualiza m - Veririfcar com o Cláudio
        print '\nMovimento: %s' % m

        self.saida.append(m)

        f = self.mimed(m)
        print 'Grau de Evidência da Direção: %s' % f

        (ge, i) = self.condicoes(m)
        print 'Grau de Evidência Máximo da Direção: %s' % ge
        print 'Distância da célula onde se encontra o Grau de Evidência Máximo: %s' % i
```

```
    c.execute('INSERT INTO rastro (x, y, evidencia) VALUES (%s,%s,1)' % (self.xd, self.yd))
    conn.commit()

    (self.xd, self.yd) = self.destino_futuro(m)

    self.n = 0

    print 'Destino (%s,%s)' % (self.xd, self.yd)



def coordenadas(self, m):
    i = 0
    (x, y) = self.destino_futuro(m)
    self.evid = []
    self.coordenada = []

    while i < self.p:
        # Consulta grau de evidência da coordenada (x,y) na tabela 'rastro' da base de dados.
        c.execute('SELECT evidencia FROM rastro WHERE x=%s AND y=%s' % (x, y))
        rt = c.fetchall()
        if rt:
            rt = rt[0][0]

        # Consulta grau de evidência da coordenada (x,y) na tabela 'matriz' da base de dados.
        c.execute('SELECT evidencia, data, id FROM matriz WHERE x=%s AND y=%s' % (x, y))
        r = c.fetchall()

        if r:
            ge = r[0][0]
            data_antiga = r[0][1]
            id = r[0][2]

            hoje = datetime.datetime.now()

            d = hoje - data_antiga
            if (d.days == 0) and (d.seconds / 60) < 60:
                e = ge
            else:
```

```
                e = 0
                c.execute('UPDATE matriz SET evidencia = 0 WHERE id=%s' % id)
                conn.commit()
            else:
                e = 0                               # Quando a coordenada não está na base de dados insere o valor 0
na lista

            self.evid.append(e)                     # Insere o grau de evidência da coordenada (x,y) na lsta.
            self.coordenada.append([x,y])           # Insere a coordenada (x,y) na lista.

            if self.n != 4 and rt == 1:             # Verifica se o robô já passou pela coordenada (x,y)
                self.evid[-1] = 1

            if m == 1:
                y = y - 1
            elif m == 2:
                x = x - 1
                y = y - 1
            elif m == 3:
                x = x - 1
            elif m == 4:
                x = x - 1
                y = y + 1
            elif m == 5:
                y = y + 1
            elif m == 6:
                x = x + 1
                y = y + 1
            elif m == 7:
                x = x + 1
            else:
                x = x + 1
                y = y - 1

            i = i + 1


        def condicoes(self, m):
#           self.m = m                  # Atualização do m - Verificar com o Cláudio!
            self.coordenadas(m)
```

```
        ge = self.rede_maximizacao()
#       print 'GE Max: %s' % ge

#       print 'Bloco Condições, evid = %s' % self.evid

        i = self.evid.index(Decimal(str(ge))) + 1

#       print 'Bloco Condições, evid = %s' % self.evid
        return ge, i


    def mimed(self, m):
#       self.m = m                  # Atualiza m - Verificar com o Cláudio.
        self.coordenadas(m)
#       print 'Bloco Mimed, evid = %s' % self.evid
        ge = self.rede_media()

#       print 'GE Med: %s' % ge
        return ge


    def beco_sem_saida(self, m):
        b = 1                       #estamos usando b poqrque n = 4. E o n = 4 é importante no Bloco Coordenadas)
        c_hor = 0
        c_ant = 0
        retorno = 0
        a = 1

        while a == 1:
            if b > 4:
                print 'BECO SEM SAÍDA'
                retorno = 1
                return retorno

            if (m + b) > 8:
                c_hor = self.condicoes(m + b - 8)[0]
            else:
                c_hor = self.condicoes(m + b)[0]
```

```
            if c_hor < self.mlim:
                for cd in self.coordenada:
                    c.execute('DELETE FROM rastro where x=%s and y=%s' % (cd[0], cd[1]))
#                     c.execute('SELECT x, y, evidencia FROM rastro')
#                     r = c.fetchall()
#                     print r
                    conn.commit()
                a = 0

            print 'm: %s' % m
            print 'b: %s' % b
            print self.coordenada
            if (m - b) < 1:
                c_ant = self.condicoes(m - b + 8)[0]
            else:
                c_ant = self.condicoes(m - b)[0]

            if c_ant < self.mlim:
                for cd in self.coordenada:
                    c.execute('DELETE FROM rastro where x=%s and y=%s' % (cd[0], cd[1]))
#                     c.execute('SELECT x, y, evidencia FROM rastro')
#                     r = c.fetchall()
#                     print r
                    conn.commit()
                a = 0

            print 'm: %s' % m
            print 'b: %s' % b
            print self.coordenada


            b = b + 1

        return retorno


    def rede_maximizacao(self):
        p1 = self.evid[0]
        p2 = self.evid[1]
        c = CNAPCls(p1, p2)
```

```
        self.celulas_max.append(c)

        i = 2
        j = 1
        while i < self.p:
            p1 = self.celulas_max[-1].sinais_resultantes(1)
            p2 = self.evid[i]
            c = CNAPCls(p1, p2)
            self.celulas_max.append(c)
            i = i + 1
            j = j + 1
#           print p1
#           print p2
        return self.celulas_max[-1].sinais_resultantes(1)


    def rede_media(self):
        ftct = 1
        ftc = 0
        p1 = self.evid[0]
        p2 = self.evid[1]
        c = CNAPa(p1, p2, ftct, ftc)
        self.celulas_med.append(c)

        i = 2
        j = 1
        while i < self.p:
            p1 = self.celulas_med[-1].saida()[0]
            p2 = self.evid[i]
            c = CNAPa(p1, p2, ftct, ftc)
            self.celulas_med.append(c)
            i = i + 1
            j = j + 1
        return self.celulas_med[-1].saida()[0]
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo