ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

EMILIANO GONÇALVES DE CASTRO

INTELIGÊNCIA DE ENXAMES APLICADA NA SÍNTESE DE ESTRATÉGIAS DE CAÇA EM AMBIENTE TRIDIMENSIONAL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia

São Paulo
2005

UNIVERSIDADE DE SÃO PAULO

Relatório de Defesa

Relatório de defesa pública de Dissertação do(a) Senhor(a) Emiliano Gonçalves de Castro no Programa: Engenharia Mecânica, do(a) Escola Politécnica da Universidade de São Paulo.

Aos 26 dias do mês de outubro de 2005, realizou-se a Defesa da Dissertação do(a) Senhor(a) Emiliano Gonçalves de Castro, apresentada para a obtenção do título de Mestre em Engenharia - Área: Engenharia Mecânica - Opção: Mecatrônica, intitulada:

"Inteligência de enxames aplicada na síntese de estratégias de caça em ambiente tridimensional"

Após declarada aberta a sessão, o(a) Sr(a) Presidente passa a palavra aos examinadores para as devidas argüições que se desenvolvem nos termos regimentais. Em seguida, a Comissão Julgadora proclama o resultado:

Codpes/Ou.
63453
PMR

Nome dos Participantes da Banca	Vínculo do Docente	Sigla da Unidade	Resultado
Marcos de Sales Guerra Tsuzuki	Presidente	EP - USP	Aprovado.
Jonas de Carvalho	Titular	EESC - USP	Aprovado.
Fabio Kawaoka Takase	Titular	EP - USP	Aprovado

Resultado Final: APROVADO.

Parecer da Comissão Julgadora *

APROVADO.

Comentários da Defesa (opcional)

Eu, Elisabete Aparecida F da Silva Ramos, Técnico Acadêmico, lavrei a presente ata, que assino juntamente com os(as) Senhores(as). São Paulo, aos 26 dias do mês de outubro de 2005.

Jonas de Carvalho

Fabio Kawaoka Takase

Marcos de Sales Guerra Tsuzuki
Orientador(a)

* Obs: Se o candidato for reprovado por algum dos membros, o preenchimento do parecer é obrigatório.

Nos termos do artigo 110, do RG-USP, encaminhe-se o presente relatório à CPG, para homologação.

Impresso em: 25/10/2005

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

EMILIANO GONÇALVES DE CASTRO

INTELIGÊNCIA DE ENXAMES APLICADA NA SÍNTESE DE ESTRATÉGIAS DE CAÇA EM AMBIENTE TRIDIMENSIONAL

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do Título de Mestre em Engenharia

Área de Concentração:
Engenharia Mecânica

Orientador: Prof. Dr. Marcos de Sales Guerra Tsuzuki

São Paulo
2005

Dedico este trabalho aos meus pais, que através de seus exemplos de vida sempre me mostraram o caminho certo a ser seguido.

Agradecimentos

A todos os colegas de curso, professores e funcionários da Escola Politécnica que contribuíram, direta ou indiretamente, para a realização deste trabalho.

Aos vários amigos e colegas que demonstraram curiosidade sobre esta pesquisa. Nossas conversas produziram contribuições que eles até desconhecem.

Aos meus colegas do Laboratório de Geometria Computacional. Em particular ao Murilo, Thiago e Vinicius, pela colaboração durante a pesquisa, e à Christiane, pela atenciosa revisão do texto.

Ao Prof. Marcio Lobo Netto, que durante a disciplina de "Vida Artificial e Ambientes Virtuais em Computação Gráfica" promoveu uma série de discussões que tiveram forte influência na definição do tema da pesquisa.

E especialmente ao Prof. Marcos Tsuzuki. Sua experiência como pesquisador proporcionou valiosas sugestões, e sua orientação foi inestimável ao longo de todo este trabalho.

Resumo

CASTRO, E. G. **Inteligência de enxames aplicada na síntese de estratégias de caça em ambiente tridimensional**. 2005. 111 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.

Sistemas de inteligência artificial distribuída podem ser ferramentas poderosas numa ampla variedade de aplicações práticas. Sua característica mais surpreendente, o comportamento emergente, é também a maior responsável pela dificuldade em projetar estes sistemas. Este trabalho propõe uma ferramenta capaz de gerar as estratégias individuais para os elementos de um sistema multiagente de forma a proporcionar ao grupo uma maneira de obter os resultados desejados, trabalhando de forma coordenada e cooperativa. Foi adotado como exemplo de aplicação um problema onde um conjunto de predadores deve capturar uma presa no espaço contínuo tridimensional. Foi implementado um sistema de síntese de estratégias cujo mecanismo interno envolve a integração entre um simulador e o algoritmo de otimização por Enxame de Partículas, uma técnica da Inteligência de Enxames. O sistema foi testado em diversos cenários de simulação e foi capaz de sintetizar automaticamente estratégias de caça bem sucedidas, atestando que a ferramenta desenvolvida pode fornecer, contanto que trabalhe com modelos bem elaborados, soluções satisfatórias para problemas de natureza complexa, de difícil resolução a partir de abordagens analíticas.

Palavras-chave: Otimização, Enxame de Partículas, Simulação, Inteligência de Enxames, Sistemas Multiagente, Inteligência Artificial Distribuída, Estratégias de Caça, Jogo da Perseguição.

Abstract

CASTRO, E. G. **Swarm intelligence applied in the synthesis of hunting strategies in three-dimensional environment**. 2005. 111 f. Dissertation (Master's Degree) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.

Distributed artificial intelligence systems can be powerful tools in a wide range of practical applications. Its most interesting feature, the emergent behavior, is also the major responsible for the hardness of projecting these systems. This work proposes a tool capable of generating individual strategies for the multiagent system elements in order to provide to the group a way to get the desired results, working in a coordinate and cooperative manner. A problem was adopted as application example where a team of predators must capture a prey in the three-dimensional continuous space. A strategies synthesis system was implemented, whose inner mechanism involves the integration between a simulator and the Particle Swarm optimization algorithm, a Swarm Intelligence technique. The system was tested in several simulation scenarios and was capable of automatically synthesize successful hunting strategies, certifying that the developed tool can supply, as long as it works with well elaborated models, satisfactory solutions for problems of complex nature, of difficult resolution from analytical approaches.

Keywords: Optimization, Particle Swarm, Simulation, Swarm Intelligence, Multiagent Systems, Distributed Artificial Intelligence, Hunting Strategies, Pursuit Game.

Sumário

1. INTRODUÇÃO, 12

2. TÉCNICAS DE OTIMIZAÇÃO, 18

2.1. Otimização de simulação, 21

2.2. Meta-heurísticas de otimização, 23

3. INTELIGÊNCIA DE ENXAMES, 26

3.1. Otimização por Enxame de Partículas (PSO), 29

4. SISTEMAS MULTIAGENTE, 35

4.1. Estratégias cooperativas em Inteligência Artificial Distribuída, 37

4.2. Exemplos de aplicações em engenharia, 42

5. O JOGO DE CAÇA TRIDIMENSIONAL, 45

5.1. Do plano discreto ao espaço contínuo tridimensional, 46

5.2. Comportamento da presa e dos predadores, 48

5.3. Parâmetros do jogo de caça, 50

5.4. As estratégias dos predadores, 52

6. SÍNTESE DE ESTRATÉGIAS PELO ENXAME DE PARTÍCULAS, 57

6.1. O ambiente de simulação, 59

6.2. Interface e visualização, 61

6.3. Otimização integrada no programa *PREDADOR*, 67

7. RESULTADOS, 70

8. CONCLUSÕES, 85

Referências Bibliográficas, 88

Anexo, 92

Lista de Figuras e Gráficos

Figura 1.1 – Diagrama de fluxo do otimizador e do Sistema Multiagente, 15

Figura 5.1 – Máquina de Estados Finitos (*Finite State Machine*) da presa, 49

Figura 5.2 – Máquina de Estados Finitos (*Finite State Machine*) dos predadores, 50

Figura 5.3 – Bolhas de percepção, proximidade e vital da presa, 52

Figura 6.1 – Diagrama de fluxo de informações, 58

Figura 6.2 – Tela do programa *PREDADOR*, 61

Figura 6.3 – Detalhe do painel de controle do programa *PREDADOR*, 62

Figura 6.4 – Janelas das câmeras de acompanhamento, 64

Figura 6.5 – O "radar 3D" , 66

Gráfico 7.1 – Análise de convergência, 74

Gráfico 7.2 – Influência de X_1 no Cenário 1, 75

Gráfico 7.3 – Influência de X_2 no Cenário 1, 76

Gráfico 7.4 – Influência de X_3 no Cenário 1, 76

Gráfico 7.5 – Influências combinadas de X_2 e X_3 no Cenário 1, 77

Gráfico 7.6 – Influência de X_1 no Cenário 2, 78

Gráfico 7.7 – Influências de X_2 e X_3 no Cenário 2, 79

Gráfico 7.8 – Influência de X_1 no Cenário 3, 79

Gráfico 7.9 – Influências de X_2 e X_3 no Cenário 3, 80

Gráfico 7.10 – Comparação entre as influências de X_2 e X_3 nos 3 cenários, 81

Figura 7.1 – Animação da movimentação das partículas no PSO, 82

Lista de Tabelas

Tabela 2.1 – Mecanismos de intensificação e diversificação, 25

Tabela 7.1 – Configuração do cenário básico (Cenário 1), 72

Tabela 7.2 – Estratégias geradas para o cenário básico (Cenário 1), 73

Tabela 7.3 – Parâmetros das estratégias geradas para os 3 cenários, 81

Lista de Siglas

AI	Inteligência Artificial (*Artificial Intelligence*)
DAI	Inteligência Artificial Distribuída (*Distributed Artificial Intelligence*)
FSM	Máquina de Estados Finitos (*Finite State Machine*)
GA	Algoritmos Genéticos (*Genetic Algorithms*)
PSO	Otimização por Enxame de Partículas (*Particle Swarm Optimization*)
SA	Recozimento Simulado (*Simulated Annealing*)

"No deserto de Nevada, uma experiência terminou tragicamente. Uma nuvem de nanopartículas – microrrobôs – escapou do laboratório. Esta nuvem é auto-sustentável e capaz de se reproduzir. É inteligente e aprende com a experiência. Para todos os efeitos práticos, ela está viva.

Ela foi programada para ser um predador. Está evoluindo rapidamente, tornando-se mais mortal a cada hora que passa.

Todas as tentativas de destruí-la fracassaram.

E nós somos a presa."

(Michael Crichton, em Prey*)*

1. INTRODUÇÃO

O mundo ao nosso redor está em contínua evolução, e raramente nos damos conta de todas as implicações deste fato. Não é comum ficarmos pensando, por exemplo, numa doença epidêmica mudando as suas características na medida em que a epidemia se alastra. Nem imaginamos a evolução de plantas e animais como um fenômeno que ocorre em termos de dias ou semanas, embora isso de fato aconteça. E nem é comum pensarmos nos nossos jardins como palcos de uma constante e sofisticada guerra química, com plantas produzindo pesticidas em resposta a ataques de insetos, e insetos desenvolvendo suas resistências aos pesticidas naturais das plantas [CRICHTON, 2002]. Embora isso ocorra também.

É isso que enxergaríamos se nos déssemos conta, no dia a dia, do verdadeiro significado da evolução. Veríamos um mundo onde não apenas plantas e insetos, mas todos os seres vivos estão mudando a cada instante, em resposta a cada outro vegetal ou animal. Populações inteiras de organismos estão em constante mudança, em ascenção ou declínio, traçando os destinos de suas espécies.

Este cenário de transformação perpétua e incansável implica num mundo no qual todas as ações humanas têm, necessariamente, efeitos incertos. O sistema total que chamamos de biosfera é tão complicado que não podemos saber de antemão as conseqüências de nada que fazemos. E esta incerteza é característica de todos os sistemas complexos, incluindo aqueles feitos pelo próprio homem [CASTI, 1997]. Ou porque não compreendemos o suficiente, ou porque o ambiente, sempre em transformação, respondeu às nossas ações de maneiras inesperadas.

Em algumas áreas, esta imprevisibilidade é enormemente amplificada pelo poder da tecnologia. Uma das áreas mais delicadas é a que diz respeito ao ponto de encontro da

nanotecnologia, biotecnologia e computação. O que estas três áreas têm em comum é a capacidade de lançar entidades auto-replicantes no nosso ambiente.

> *"Dentro de cinquenta a cem anos, é muito provável que uma nova classe de organismos vá emergir. Estes organismos serão artificiais no sentido de que serão originalmente projetados por humanos. Entretanto, eles irão se reproduzir, e irão "evoluir" para algo diferente de suas formas originais; eles serão "vivos" de acordo com qualquer definição razoável desta palavra. Estes organismos evoluirão de maneira fundamentalmente diferente... O ritmo... será extremamente rápido... O impacto sobre a humanidade e a biosfera poderá ser enorme, maior que o da revolução industrial, armas nucleares ou poluição ambiental. Nós devemos agir já para controlar a emergência de organismos artificiais..."*
>
> *Doyne Farmer e Alleta Belin,* apud [CRICHTON, 2002]

Mesmo sem chegar ao extremo de temer uma dominação do nosso planeta, é perfeitamente compreensível e justificável a apreensão que o desenvolvimento destas tecnologias traz para uma parte da comunidade científica. O aparecimento de seres[1] minúsculos (ou até virtuais), com reduzida capacidade de "estrago" quando isolados, mas com enorme potencial de destruição quando atuam em conjunto (dado o imprevisível resultado das complexas inter-relações destes seres) pode, de fato, trazer enormes problemas. Ou um novo universo de soluções.

Existem diversas abordagens para tentar conduzir esta brutal "força criativa" a serviço do homem. No campo da engenharia, merece destaque a área de Inteligência Artificial Distribuída e otimização por Inteligência de Enxames, onde problemas complexos têm sido resolvidos por indivíduos autônomos simples que apresentam um comportamento

[1] Vivos ou artificiais, muito embora esta distinção esteja cada dia mais difícil com os avanços na área de vida artificial [ADAMI, 1998].

global que não é evidente a partir da análise individual de cada um[2]. Comportamentos como, por exemplo, a capacidade de auto-organização.

Uma vez que o poder destas abordagens reside em boa parte na característica de comportamento emergente, gostaríamos, idealmente, de poder forçar essa emergência a ir sempre diretamente de encontro às expectativas de solucionar os nossos problemas. Em outras palavras, gostaríamos de *projetar* a emergência. Que, pela própria definição, não é passível de projeto no seu sentido mais tradicional.

Aliás, é até possível que esta incapacidade possa enquadrar a inteligência humana dentro de uma escala de complexidade. Afinal, a partir de ponto de vista um pouco menos antropocêntrico, "*a emergência pode ser considerada um conceito que representa a simplicidade das nossas mentes; não é uma propriedade do sistema em questão, mas da nossa incapacidade de entendê-lo*" [KENNEDY, EBERHART, 2001].

Em casos onde é necessária a avaliação de sistemas complexos com grande número de variáveis e interações para os quais não há solução analítica, em geral ligados ao auxílio na tomada de decisões, a simulação vem tendo o seu papel reconhecido. Isso se deve a uma série de fatores, dentre eles a capacidade que a simulação apresenta de modelar aspectos dinâmicos e estocásticos de um sistema, gerando resultados mais precisos do que os obtidos a partir de planilhas de cálculo estáticas e determinísticas, como notado em [MAGOULAS, *et al*, 2002].

Uma simulação é considerada uma ferramenta para responder questões do tipo "e se..." porque é, isoladamente, uma técnica de *avaliação* de soluções, e não de *síntese* de soluções. Mas esta situação pode mudar com a ajuda de um procedimento de otimização. Nesse caso, uma ferramenta que integre simulação e otimização poderia responder não apenas questões do tipo "e se..." mas também questões do tipo "como...".

O objetivo deste trabalho vai exatamente nesta direção: propor uma ferramenta de síntese de soluções para um problema envolvendo um sistema dinâmico e complexo (gerar estratégias para um jogo de caça entre predadores e presa num ambiente

[2] Esta é uma das definições de "emergência".

tridimensional) cuja solução envolve o comportamento emergente de seus agentes (no caso, dos predadores). E esta ferramenta tem, nas suas "entranhas", um algoritmo de otimização (a Otimização por Enxame de Partículas, técnica da Inteligência de Enxames) que é, por sua vez, também baseado em comportamento emergente (Fig. 1.1). Ou seja, uma ferramenta que usa emergência para *projetar* emergência.

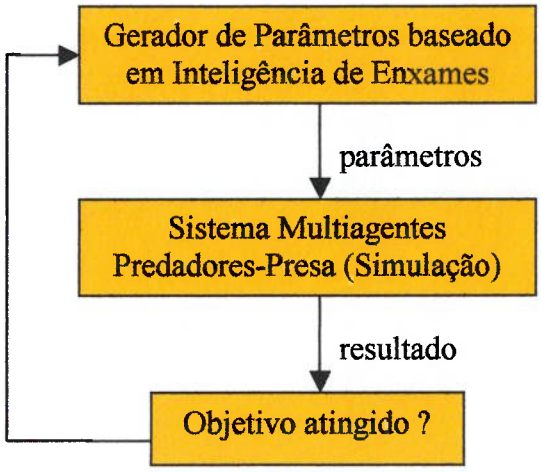

Figura 1.1 – Diagrama de fluxo do otimizador (gerador de parâmetros) baseado em Inteligência de Enxames e do Sistema Multiagente.

De certa forma, é assim que a natureza vem resolvendo os seus problemas[3]. As soluções encontradas nos seres vivos[4] são resultado, quase sempre, de um processo evolutivo, baseado na seleção natural dos organismos mais adaptados; no caso, aqueles que, por um processo estocástico, possuem as melhores características em relação a determinado problema gerado pelo meio ambiente. A solução também *emerge*, por assim dizer, deste processo evolutivo. Em outros casos, os organismos reagem de forma social ao problema, através de interações entre os indivíduos: deste processo também emergem soluções, originadas a partir de uma espécie de inteligência coletiva.

[3] Se não for muito ousado imaginar que a natureza tem algum compromisso com a resolução de problemas.
[4] Em termos anatômicos, fisiológicos, comportamentais, etc.

A ferramenta proposta neste trabalho tira proveito do crescente poder computacional para simular processos que, em suas mais próximas contrapartidas naturais, seriam extremamente mais demorados. É como se dispuséssemos do poder da evolução natural de Darwin, só que ao ritmo dos microprocessadores.

Embora isso possa parecer confuso à primeira vista, é esperado que ao longo deste trabalho fique clara a distinção de que no caso dos predadores (sistema multiagente), a emergência diz respeito ao comportamento resultante da simulação de um fenômeno real (a caça em grupo), enquanto que no segundo caso (o do enxame de partículas) a emergência está relacionada a um fenômeno natural que serviu de inspiração para um programa de computador (insetos sociais).

Esta sutil distinção, discutida em [KENNEDY, EBERHART, 2001], diz respeito à utilidade das metáforas na compreensão de sistemas matemáticos abstratos. Não há nada no algoritmo do Recozimento Simulado (*Simulated Annealing*) que o relacione intrinsecamente ao resfriamento de moléculas. O algoritmo pode ser descrito em símbolos algébricos abstratos, como qualquer outro algoritmo. Mas foi a metáfora do recozimento que permitiu a sua conceitualização (e a criação do método, antes de mais nada), assim como é esta metáfora que permite que as pessoas entendam como ele funciona: seria bem mais difícil ter uma idéia básica do processo através de símbolos algébricos puros.

Esta aparente ambigüidade (nas duas versões de emergências citadas) decorre do fato de que na ferramenta projetada neste trabalho existe um programa de computador que contém uma rotina que procura *simular* um processo real (o simulador) e outra que é *explicada* em termos de um processo real (o otimizador). A metáfora é, neste contexto, o outro lado da simulação; e analisar estes dois conceitos lado a lado talvez ajude a facilitar a compreensão. Existe uma simetria especial entre um programa de computador e determinado fenômeno do mundo real. Por um lado, a simulação é um programa que ajuda a explicar algo sobre o mundo; e por outro, o mundo às vezes simplesmente oferece um modelo que nos ajuda a entender um programa.

Assim, seria mais completo (e talvez mais elucidativo) afirmar que o objetivo deste trabalho é fazer com que emergência (metafórica) projete emergência (simulada). E que

embora a aplicação escolhida aqui tenha caráter lúdico, serão apresentados no Capítulo 4 alguns exemplos de uso de Inteligência Artificial Distribuída em engenharia, que fazem parte de uma mesma classe de problemas e que podem, analogamente, sofrer a mesma abordagem proposta neste trabalho.

Ao longo deste trabalho vários dos conceitos já citados serão apresentados com maior detalhamento. No Capítulo 2, serão analisadas algumas técnicas de otimização; uma delas, a Otimização por Enxame de Partículas, uma técnica de Inteligência de Enxames, será abordada com mais detalhes no Capítulo 3; no Capítulo 4 serão discutidos alguns conceitos de Inteligência Artificial Distribuída e Sistemas Multiagente; o problema da estratégia de caça em grupo no ambiente tridimensional, que será caracterizado como o cenário de estudo desta pesquisa, será definido em detalhes no Capítulo 5; o Capítulo 6 abordará o projeto de uma ferramenta para síntese de estratégias, que é o objetivo deste trabalho e alguns resultados serão apresentados no Capítulo 7; algumas considerações finais serão relacionadas no Capítulo 8.

A escolha de uma otimização inspirada em comportamento emergente para resolver um problema que envolve o mesmo fenômeno de inteligência "social" não foi completamente acidental. Estudos como o realizado por [MAGOULAS, *et al*, 2002] indicam que as técnicas de inteligência de enxames levam uma significativa vantagem sobre outros métodos (como Recozimento Simulado e Algoritmos Genéticos) em otimização de simulações. O que não deixa de ser mais uma evidência de que a inteligência distribuída veio para ficar. Ou, numa visão mais apocalíptica, que a dominação já começou.

2. TÉCNICAS DE OTIMIZAÇÃO

O processo evolutivo, baseado na seleção natural, é responsável pelo fato de encontrarmos nos seres vivos soluções engenhosas para uma ampla gama de desafios impostos pelo meio ambiente, desde o nível comportamental e fisiológico até os mais elementares processos bioquímicos. O problema de capturar uma presa para garantir a própria subsistência é, por exemplo, a metáfora para o problema da caça apresentado neste trabalho.

Ainda a respeito desta metáfora, deve-se destacar que apenas obter sucesso na caçada não é suficiente. Também é importante encontrar uma estratégia para garantir a melhor probabilidade de sucesso nas mais diversas circunstâncias e, na medida do possível, da forma mais rápida. Este é um fator decisivo para que os predadores tenham uma vantagem competitiva capaz de favorecer a perpetuação da espécie. Do ponto de vista evolutivo, o que importa para os predadores não é a adoção de uma estratégia qualquer, mas de uma estratégia otimizada.

Em geral, o termo otimização refere-se ao processo de ajuste de um sistema de modo a se obter o melhor resultado possível. Algumas vezes, um bom resultado já é o suficiente, sobretudo quando existem fontes de incerteza e erros de medição superiores às diferenças entre boas soluções. Nestes casos, a busca pelo melhor possível seria impraticável ou mesmo desnecessária [GRANDI, 2003].

Situações nas quais um problema complexo deve ser resolvido de forma satisfatória e com restrições de tempo são as que representam os problemas práticos mais comuns. A resolução destes problemas por métodos exatos, quando possível, oferece soluções ótimas. Mas, em muitos casos, a um custo computacional bastante elevado.

Logo, o termo otimização corresponderia à tentativa de encontrar uma solução aceitável ao invés de uma solução ótima, comprovável matematicamente. Afinal, seguindo na metáfora motivadora, a da caça, a natureza não busca necessariamente uma solução ótima global para um problema; o que importa é apenas que a solução seja boa o bastante. O que nos traz a *Lei da Suficiência* [KENNEDY, EBERHART, 2001]: Se uma solução é *boa* o bastante[5], e é *rápida* o bastante e é *barata* o bastante, então ela é *suficiente*. Na maior parte das aplicações do mundo real, ficamos satisfeitos com soluções suficientes.

A otimização pode ser visualizada como um processo que ocorre em três espaços numéricos inter-relacionados [KENNEDY, EBERHART, 2001]:

- **Espaço de Parâmetros**: Contém todos os valores de todos os elementos – os parâmetros – que podem servir de entrada para a função que se deseja otimizar. Os problemas mais interessantes de otimização são multidimensionais, e às vezes híbridos em relação à natureza de seus parâmetros (alguns contínuos e outros discretos). Às vezes existem regiões impraticáveis no espaço de parâmetros, que correspondem a padrões de entrada que são paradoxais, inconsistentes ou sem significado.

- **Espaço de Funções**: Uma função é um conjunto de operações definidas sobre os parâmetros, e o espaço de funções contém os resultados destas operações. Geralmente o espaço de funções é unidimensional, mas saídas multidimensionais podem ser encontradas, por exemplo, em otimizações multi-objetivas, muito comuns em tomada de decisão.

- **Espaço de Aptidão**: Contém os níveis de sucesso com os quais os padrões de parâmetros otimizam os valores no espaço de funções. Só pode ser unidimensional. Quando o objetivo é maximizar o resultado de uma única função (o que é o caso na otimização para o jogo de caça proposto), o Espaço de Funções e o Espaço de Aptidão são os mesmos. Mas vale a pena manter a

[5] "*Boa* o bastante" significa que a solução satisfaz as especificações.

distinção em mente: o objetivo da otimização é encontrar os parâmetros que maximizem a aptidão.

Para otimizações de funções com espaços de parâmetros contínuos (o que é o caso na otimização dos parâmetros das estratégias dos predadores), em muitos casos práticos existem diversos obstáculos para a aplicação de métodos matemáticos clássicos, como a presença de descontinuidades, ruídos e não-linearidades, o que torna estes problemas indicados para a resolução por meio de métodos heurísticos, dada a sua tolerância e flexibilidade.

Apesar de relativamente recente, a otimização heurística já se tornou um método tradicional de resolução de problemas. Na área de ciências exatas, *heurística* (que significa "a arte de encontrar") é definida com uma metodologia (ou algoritmo) utilizada para resolver problemas por processos que, embora não rigorosos, geralmente refletem o conhecimento humano e permitem obter uma solução satisfatória. E, a despeito de existir algum suporte teórico, a principal justificativa para a utilização da abordagem heurística é o seu desempenho empírico.

As técnicas heurísticas podem ser divididas em duas classes distintas [GRANDI, 2003]: algoritmos construtivos (ou de busca local) e meta-heurísticas (ou heurísticas de coordenação).

Os algoritmos construtivos são baseados nas características estruturais do problema, utilizam essas informações para a própria implementação, e possuem características e desempenhos próprios, que se manifestam de maneira diferente a cada tipo de problema. As meta-heurísticas coordenam e direcionam uma busca local ou construtiva de modo a evitar que o algoritmo fique estagnado em um ponto de ótimo local.

2.1. Otimização de simulação

É muito comum o caso, nas mais inúmeras aplicações, onde o desempenho de um sistema depende do ajuste de um conjunto de parâmetros, mas onde a complexidade deste sistema impede que se possa elaborar um modelo analítico. Nestes casos, é necessário fazer uso de uma simulação para estimar o desempenho do sistema em função de cada conjunto sugerido de parâmetros [ANDRADÓTTIT, 1998].

Entretanto, muitas vezes uma simples avaliação de desempenho é insuficiente, e uma abordagem que envolva algum tipo de processo de exploração pode ser necessário. Nestes casos, podem ser utilizadas técnicas de otimização de simulação (*simulation optimization*), que podem ser definidas como processos de busca dos melhores valores de determinado conjunto de variáveis para um sistema onde o desempenho é avaliado a partir dos resultados gerados por um modelo de simulação deste sistema [ÓLAFSSON, KIM, 2002].

Tipicamente, otimizações de simulação têm como componentes um conjunto de variáveis de "decisão", uma função objetiva e algumas restrições específicas do problema. As variáveis podem ser contínuas ou discretas, as restrições geralmente são expressas como limitações no domínio destas variáveis e a função objetiva é uma função real definida a partir destas variáveis.

Porém, dada a complexidade e a natureza estocástica destes problemas, não existem expressões analíticas para esta função. A função objetiva é, então, estimada usando uma função do resultado estocástico de uma simulação, que tende a ser uma aproximação imparcial da verdadeira função objetiva.

As técnicas de otimização de simulação[6] mais freqüentemente adotadas na literatura são aproximação estocástica [ROBBINS, MONRO, 1951 *apud* ÓLAFSSON, KIM, 2002], metodologia de superfície de resposta, seleção estatística [MATEJCIK, NELSON, 1995 *apud* ÓLAFSSON, KIM, 2002] e busca aleatória. Mais recentemente, buscas baseadas

[6] Em geral classificadas tendo como base a natureza dos domínios das variáveis de "decisão", contínuas ou discretas.

em meta-heurísticas (como recozimento simulado, busca tabu e algoritmos genéticos) também têm sido alvo de pesquisas na área.

De acordo com [ÓLAFSSON, KIM, 2002], muito embora tenham sido projetadas, de forma geral, para operar em contextos determinísticos[7] e não garantam convergência, as técnicas baseadas em meta-heurísticas têm sido particularmente bem sucedidas em aplicações de otimizações baseadas em simulação. São métodos que têm sido aplicados com sucesso em problemas industriais, e foram incorporados em pacotes de *software* de simulação, como o AutoStat da AutoMod (algoritmos genéticos), o OPTIMIZ da SIMUL8 (redes neurais) e o pacote da OptQuest, que funciona com programas de simulação como Arena e Crystal Ball (busca tabu e redes neurais).

Como se pode perceber, os pacotes comerciais de otimização baseada em simulação são atualmente dominados por abordagens meta-heurísticas. Ou seja: na prática, estes métodos têm precedência sobre os algoritmos que tradicionalmente receberam mais atenção da comunidade acadêmica, historicamente focado em abordagens baseadas em gradiente e com provas de convergência [ÓLAFSSON, KIM, 2002].

Das diversas razões que podem ser apontadas para justificar este fato, uma seguramente é a constatação de que propriedades como convergência assintótica não são assim tão relevantes na prática. Além de que as meta-heurísticas são geralmente rápidas e robustas.

O aparente desalinhamento entre a pesquisa acadêmica e a implementação prática tem sido amplamente reconhecido e, pelo teor das pesquisas mais recentes, uma reaproximação está em andamento.

A síntese de estratégias de caça para os predadores é um caso típico de problema onde a complexidade do sistema não permite a elaboração de um modelo analítico. Assim, um modelo de simulação terá o papel de avaliar a aptidão das soluções geradas pelo algoritmo de solução. O resultado da simulação determinará a função objetiva. E vale

[7] Mas tolerantes a ruídos, em alguns casos.

lembrar que o modelo de simulação, ao refletir as "regras do jogo", terá um componente estocástico.

No entanto, a natureza estocástica da simulação não deve ser encarada como um problema. Afinal, todos os processos naturais que serviram de inspiração para os algoritmos evolutivos, incluindo aí aqueles baseados em inteligência de enxames, estão sujeitos a expressivas perturbações aleatórias que não comprometem, a longo prazo, as suas eficácias em encontrar soluções satisfatórias.

2.2. Meta-heurísticas de otimização

A partir de meados da década de 70, diversos métodos implícitos ou explícitos de gerenciamento e coordenação de *diversificação* e *intensificação* de busca foram propostos com o objetivo de melhorar as soluções que existiam até então. Assim nasceu a classe de algoritmos meta-heurísticos.

Segundo [GRANDI, 2003], meta-heurística pode ser definida como uma estratégia de alto nível destinada a coordenar e modificar operações realizadas por uma determinada heurística (normalmente específica ao problema de aplicação) de forma a melhorar seu desempenho. O objetivo principal desta estratégia é evitar que a heurística fique estagnada em pontos de ótimo local do problema. Este objetivo tanto pode ser atingido pela adição da capacidade do algoritmo de atravessar pontos de mínimo local adiante, como pela geração de soluções iniciais (ou pontos de partida) de forma conveniente ao invés de meramente aleatórias. Ainda de acordo com [GRANDI, 2003]:

> *"Muitas destas estratégias podem ser interpretadas como desvios introduzidos nos movimentos realizados pela heurística, de forma a facilitar que a mesma alcance soluções melhores. Estes desvios podem ser introduzidos de várias formas: com a alteração da função objetiva, pela utilização de memória (proveniente de tomadas de decisão anteriores ou conhecimentos prévios sobre o problema), etc."*

A estrutura de cada meta-heurística é baseada na filosofia adotada em sua concepção. Como ilustração das diversas variações, podemos ter [GRANDI, 2003]:

- **Origem**: Uma meta-heurística pode ser desenvolvida a partir de um modelo natural, como os Algoritmos Genéticos ou a Otimização por Colônia de Formigas, de funcionamento análogo à teoria da evolução das espécies ou ao comportamento de formigas, respectivamente. Outras concepções podem ser estabelecidas pelo estudo de uma heurística e o conseqüente desenvolvimento de uma estratégia de supervisão deste processo, como na Busca Tabu.

- **Diversidade**: A escolha de uma única solução de trabalho ou de uma população de soluções simultâneas se constitui em uma outra variação. Métodos de trajetória caracterizam algoritmos que trabalham com uma única solução, como a Busca Tabu e o Recozimento Simulado. Métodos baseados em populações, como os Algoritmos Genéticos, Otimização por Colônia de Formigas e a Otimização por Enxame de Partículas, são constituídos por conjuntos de pontos que evoluem dentro do espaço de busca.

- **Função Objetiva**: Na medida em que alguns algoritmos mantêm intacta a função objetiva, outros a alteram ao longo da busca com o intuito de escapar de pontos de ótimo local.

- **Estrutura da Vizinhança**: Ao passo que a maioria das meta-heurísticas utilizam uma estrutura para a descrição de soluções vizinhas, outras abordagens utilizam um conjunto de estruturas de vizinhança e oferecem a possibilidade de diversificação da busca.

- **Memória**: Uma heurística sem memória utiliza apenas o estado atual do processo de busca como informação. Porém, entre os métodos mais bem sucedidos, a utilização de memória é um elemento comum. São exemplos de memória os movimentos recentes, as soluções visitadas e as decisões tomadas, entre outros.

Tabela 2.1 – Mecanismos de intensificação e diversificação [GRANDI, 2003].

Meta-heurística	Método de Intensificação	Método de Diversificação
Recozimento Simulado[8]	busca local	temperatura
Busca Tabu [8]	busca local	lista tabu
Computação Evolutiva[9]	seleção	mutação
Colônia de Formigas [9]	depósito de feromônio	evaporação
Enxame de Partículas [9]	imitação	comparação probabilística

De modo geral, meta-heurísticas podem ser vistas como estratégias que dosam e gerenciam processos de *intensificação* e *diversificação* em buscas. A busca realizada por uma meta-heurística deve comportar a realização de buscas intensivas em locais do espaço que contenham soluções de qualidade (intensificação) sem a perda da capacidade de direcionar a busca para locais não explorados quando necessário (diversificação). Em outras palavras, uma meta-heurística é um ajuste dinâmico entre a exploração da experiência acumulada durante a busca e a exploração do espaço de busca do problema. Na Tabela 2.1 estes mecanismos são apresentados de forma sintética.

[8] Meta-heurística baseada em trajetória.
[9] Meta-heurística baseada em populações.

3. INTELIGÊNCIA DE ENXAMES

O termo "inteligência de enxames" surgiu pela primeira vez no trabalho seminal de Bonabeau, Dorigo e Theraulaz, pesquisadores do Santa Fe Institute. Originalmente criado para descrever um paradigma particular do campo de pesquisa da Robótica. De acordo com os pesquisadores [BONABEAU, *et al*, 1999]:

> *"O uso da expressão 'inteligência de enxames' para descrever apenas este trabalho parece desnecessariamente restritivo: por isso estendemos esta definição para incluir qualquer tentativa de projetar algoritmos ou dispositivos distribuídos de solução de problemas inspirados pelo comportamento coletivo de colônias de insetos ou outras sociedades animais."*

A rigor, o termo "enxames" já havia sido utilizado neste mesmo contexto num documento ainda anterior do Santa Fe Institute sobre o sistema de simulação SWARM [KENNEDY, EBERHART, 2001]:

> *"Usamos o termo 'enxame' no sentido geral para nos referir a qualquer coleção vagamente estruturada de agentes interativos. O exemplo clássico de enxame é o enxame de abelhas, mas a metáfora do enxame pode ser estendida para outros sistemas com arquitetura similar. Uma colônia de formigas pode ser imaginada como um enxame cujos agentes individuais são formigas, um bando de pássaros é um enxame cujos agentes são pássaros, o trânsito é um enxame de carros, uma multidão é um enxame de pessoas, um sistema imunológico é um enxame de células e moléculas, e a economia é um enxame de agentes econômicos. Apesar da noção de enxame sugerir uma idéia de movimento coletivo no espaço, como no enxame do bando de pássaros, nós*

estamos interessados em todos os tipos de comportamento coletivo, e não apenas movimento espacial."

A Inteligência de Enxames compartilha com a noção amplamente utilizada em Inteligência Artificial de que a cognição humana é o padrão ideal de inteligência. Porém, ao contrário do resto deste campo de conhecimento, não concorda com a idéia de "mente" como equivalente a "cérebro", como parte de um processo privativo interno e isolado. A Inteligência de Enxames parte do pressuposto de que a mente, e o processo cognitivo como um todo, depende fortemente de uma componente social.

Essa componente social está relacionada à capacidade de interação entre os elementos do enxame, possível através de algum tipo de comunicação, implícita ou explícita. A comunicação promove uma mudança cultural, que representa o mecanismo de variação de "atitude" dos componentes do enxame.

A importância desta componente social pode ser percebida pelo provocativo argumento de que os humanos não socializados não têm o que normalmente pensamos como mentes [KENNEDY, EBERHART, 2001]. Não podem pensar nem se comunicar num nível mais elevado, e seu aprendizado é restrito ao tipo de experiência individual que omite a acumulação de conhecimento através da cultura. A partir deste ponto de vista, podemos concluir que a cultura não "fortalece" ou "enriquece" a mente; a cultura "cria" a mente.

Também considerada como uma integrante da classe de otimização da computação evolutiva, a Inteligência de Enxames apresenta razoável semelhança com os Algoritmos Genéticos. Enquanto nos últimos o mecanismo de variação é baseado na evolução biológica, nos enxames a inspiração é a mudança cultural; fenômenos similares, na medida em que são ambos dinâmicos, estocásticos, adaptativos e ocorrem em populações. Mas enquanto a seleção natural é vital na evolução Darwiniana, ela é um aspecto pouco relevante na mudança cultural. Afinal, mentes se adaptam pela mudança, e não pela sobrevivência dos mais aptos, não pela constante substituição dos mais fracos.

Outra importante diferença entre evolução cultural e biológica é que características culturais são transmitidas horizontalmente, ou seja, entre indivíduos de uma mesma geração, enquanto características herdadas geneticamente só podem ser propagadas verticalmente, de uma geração para a próxima.

Os modelos sócio-cognitivos utilizados pela Inteligência de Enxames derivam de duas tradições distintas. Uma delas é a de programas de computador projetados por cientistas sociais para tentar entender melhor o comportamento humano. Um psicólogo, economista ou antropólogo pode ter uma teoria a respeito de como as pessoas pensam ou se comportam, e gostaria de estudar as conseqüências de suas teorias, ou então descobrir como o ajuste de certos parâmetros afeta os resultados da simulação.

A outra tradição vem da ciência da computação, matemática e engenharia. Essa abordagem de modelar a cultura busca formas mais interessantes de resolver problemas matemáticos complexos. Os parâmetros são ajustados não para aprender algo em particular sobre a sociedade humana, mas para atingir os melhores resultados – achar em menor tempo a melhor solução para um problema difícil.

A idéia de que as bases teóricas da Inteligência de Enxames nasceram desta confluência de áreas encontra eco na história dos algoritmos genéticos, onde biólogos e cientistas da computação trabalham lado a lado para explicar e emular o processo dinâmico que é a evolução, para aprender biologia e para resolver problemas de otimização matemática. Também com a pesquisa em redes neurais artificiais, onde psicólogos e cientistas de cognição esperam obter algum tipo de compreensão a respeito do pensamento humano, enquanto cientistas da computação e engenheiros buscam meios poderosos para a análise de dados [KENNEDY, EBERHART, 2001].

Neste trabalho foi utilizada a técnica de otimização por enxame de partículas (*Particle Swarm Optimization* - PSO), uma representante, ao lado da otimização por colônia de formigas, da classe de algoritmos da Inteligência de Enxames.

3.1. Otimização por Enxame de Partículas (PSO)

O algoritmo do PSO possui como bases teóricas modelos sócio-cognitivos bem simples. Seus princípios fundamentais são [KENNEDY, EBERHART, 2001]:

- **Avaliação** dos estímulos, caracterizados como positivos, negativos, atrativos ou repulsivos. A avaliação representa talvez a mais evidente característica comportamental dos seres vivos. O aprendizado torna-se possível quando o organismo é capaz de avaliar e distinguir características do ambiente. Portanto, o aprendizado poderia ser definido como a capacidade do organismo em melhorar a sua avaliação média do ambiente.

- **Comparação** entre indivíduos. Inspirado em modelos de adaptação cultural e teorias sócio-psicológicas, o modelo adaptativo cultural adota como referência o comportamento de outros indivíduos. Isto é, ele estabelece os padrões de referência da sociedade mediante a comparação entre os indivíduos. Em suma, cada indivíduo realiza a comparação de si com os seus vizinhos e imita apenas os que julga superiores.

- **Imitação**. Por se constituir em uma atividade relativamente restrita na natureza, a imitação é encontrada, de fato, apenas entre algumas espécies de animais, incluindo o ser humano. A imitação é o terceiro princípio de funcionamento do enxame de partículas pelo fato de se constituir em uma forma muito eficiente de aprendizado.

A combinação destes três conceitos em programas de computador permitiu que seres muito simplificados fossem capazes de se adaptar e, conseqüentemente, resolver problemas de extrema complexidade [GRANDI, 2003].

O PSO pertence à classe dos algoritmos evolucionários baseados em população, assim como os algoritmos genéticos. Mas ao contrário destes, que têm como metáfora a sobrevivência dos mais bem adaptados, no PSO a motivação é a simulação de um

comportamento social. Assim como os outros algoritmos evolucionários, o PSO é inicializado com uma população de soluções aleatórias. A principal diferença é que no PSO, cada solução potencial (indivíduo) é também associada a uma velocidade randômica na inicialização; as soluções potenciais, chamadas de *partículas*, podem então "voar" pelo espaço de parâmetros do problema [COELHO, KROHLING, 2005].

Cada partícula mantém registro da coordenada no espaço de parâmetros do problema onde obteve, individualmente, a melhor solução até então. Também tem conhecimento da melhor solução obtida por todas as partículas[10] até o momento (global), e das coordenadas correspondentes à esta solução. O algoritmo consiste em, a cada passo de iteração, recalcular a velocidade de cada partícula considerando os melhores registros individuais e globais, ponderados por termos aleatórios gerados de forma independente. A mudança de velocidade (aceleração) é definida então em função de uma avaliação "subjetiva" (probabilística) dessas informações para cada uma dessas partículas.

A idéia de mover cada partícula numa direção relacionada a um ponto do espaço de parâmetros onde ela (ou mesmo outra partícula) obteve um bom resultado parece trivial, óbvia até, mas está associada a uma importante propriedade das "paisagens" em funções multidimensionais, e que pode ser explorada por um otimizador. Esta propriedade estabelece que a distância entre dois pontos no espaço multidimensional está correlacionada com a diferença nos valores de determinada função nestes pontos, desde que esta função não seja aleatória. Ou seja, se o otimizador encontrou um ponto bom, é provável que exista outro ainda melhor ali por perto.

Uma variação do algoritmo do PSO diz respeito à vizinhança considerada. Uma partícula pode usar como referência o melhor resultado obtido por qualquer partícula do conjunto, ou apenas pelas suas partículas "vizinhas". Nesse caso, a vizinhança é no sentido topológico, e não pela eventual proximidade momentânea no espaço de parâmetros. Desta forma, o algoritmo tem então um parâmetro que diz respeito ao tamanho desta vizinhança a ser considerada.

[10] Ou, numa variação do algoritmo, da melhor solução obtida pelas partículas de sua vizinhança pré-definida.

É importante deixar claro que tanto posições quanto velocidades se referem aqui ao espaço *n*-dimensional, são vetores *n*-dimensionais, onde *n* é a dimensão do espaço de parâmetros do problema. Neste sistema, cada partícula *i* na posição $\vec{x}_i$ e com uma velocidade $\vec{v}_i$ é atualizada a cada iteração *t*, de acordo com a seguinte regra:

$$\vec{v}_i(t) = \varpi \cdot \vec{v}_i(t-1) + \varphi_1 \cdot \left[\vec{p}_i - \vec{x}_i(t-1)\right] + \varphi_2 \cdot \left[\vec{p}_g - \vec{x}_i(t-1)\right] \tag{3.1}$$

Onde $\vec{p}_i$ é a melhor posição até então encontrada pela partícula *i*, $\vec{p}_g$ é a melhor posição entre todas as partículas, e φ_1 e φ_2 são variáveis aleatórias positivas, distribuídas uniformemente nos intervalos [0; $\varphi_{1_{max}}$] e [0; $\varphi_{2_{max}}$], calculadas a cada iteração para cada partícula. ϖ é o momento de inércia. Em algumas implementações é utilizado um decréscimo linear deste parâmetro ao longo da busca [GRANDI, 2003].

A seguinte associação, apresentada em [COELHO, KROHLING, 2005], pode ser utilizada para interpretar os 3 termos à direita da expressão (2.2): o primeiro termo diz respeito à componente "inercial" da partícula (onde ϖ representa o "peso" deste momento de inércia); o segundo, à parte "cognitiva" da partícula, pois representa o pensamento independente da própria partícula; por fim, o terceiro termo é a parte "social", que representa a colaboração proveniente da comunicação entre as partículas. φ_1 e φ_2 são os "pesos" das partes "cognitiva" e "social", "empurrando" as partículas, respectivamente, para as melhores soluções encontradas individualmente e globalmente (ou, para ser mais rigoroso, da vizinhança considerada).

$$\vec{v}_i(t) = \boxed{\varpi \cdot \vec{v}_i(t-1)} + \boxed{\varphi_1 \cdot \left[\vec{p}_i - \vec{x}_i(t-1)\right]} + \boxed{\varphi_2 \cdot \left[\vec{p}_g - \vec{x}_i(t-1)\right]} \tag{3.2}$$

componente "inercial" | componente "cognitiva" | componente "social"

A posição de cada partícula é atualizada a cada iteração a partir do vetor de velocidades (adotando aqui um intervalo de tempo unitário entre cada iteração):

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{3.3}$$

Pode ser observado que o efeito da aplicação simultânea destas duas últimas equações é o desvio do "enxame" de partículas na direção do ponto:

$$\frac{\varphi_1 \cdot \vec{p}_i + \varphi_2 \cdot \vec{p}_g}{\varphi_1 + \varphi_2} \tag{3.4}$$

A média desta variável aleatória define um ponto que se encontra em uma região próxima à média ponderada entre as duas melhores posições conhecidas pela partícula, $\vec{p}_i$ e $\vec{p}_g$ [KENNEDY, EBERHART, 2001].

Um outro parâmetro presente neste sistema é o V_{max}, vinculado a um sistema de limitação de velocidade para as partículas. Este sistema impede que o algoritmo desenvolva valores de velocidades não pertinentes ao problema de aplicação, em qualquer uma das dimensões *d* do vetor $\vec{v}_i$:

$$-V_{max} \leq v_{id} \leq V_{max} \tag{3.5}$$

Caso qualquer uma das partículas do sistema desenvolva, em qualquer uma de suas dimensões, uma velocidade superior a V_{max}, este valor é automaticamente saturado em V_{max}. A importância de V_{max} se deve ao fato de que este é o parâmetro que determina, na prática, a resolução com a qual a busca é realizada nas regiões que circundam as soluções correntes. Valores muito elevados para V_{max} privilegiam buscas mais amplas, globais, mas as partículas podem "saltar" boas soluções. Valores muito baixos, por outro lado, privilegiam buscas locais, fazendo com que as partículas talvez não explorem suficientemente além das regiões em que se encontram [COELHO, KROHLING, 2005]. Assim, V_{max} é, por si só, um importante elemento para equilibrar a intensificação e a diversificação.

Caso se considere a aplicação em conjunto dessas regras, é possível descrever o funcionamento do algoritmo do PSO da seguinte maneira [GRANDI, 2003]:

1) **Inicialização**: Esse passo prepara o sistema para ser executado. As velocidades e as posições das partículas são inicializadas aleatoriamente dentro dos limites pré-estabelecidos (posições dentro dos limites do espaço de parâmetros do problema e velocidades, em cada dimensão, dentro de $[-V_{max}, +V_{max}]$).
2) **Atualização**: A cada repetição deste passo são realizados procedimentos de atualização da solução para o problema:
    a) Cada partícula avalia o problema em função de sua posição.
    b) O índice de melhor posição encontrada até então é atualizado para cada partícula (memória individual da melhor posição).
    c) O índice da melhor partícula global (ou da vizinhança) é atualizado (partícula referência para as demais).
    d) As velocidades são atualizadas de acordo com a equação (2.1). Nesse passo, a partícula imita (ou segue) probabilisticamente os resultados obtidos pela melhor partícula, mas também leva em conta os resultados que "experimentou" e "julgou" bons.
    e) As velocidades são saturadas em V_{max} caso necessário, em respeito à equação (2.5).
    f) As posições são atualizadas pela equação (2.3).
3) **Condição de término**: São verificados o número máximo de iterações e o valor da melhor solução corrente. Caso algum deles seja satisfatório, o sistema é finalizado. Outra possibilidade é a detecção da estagnação do algoritmo. Neste caso, o algoritmo também é finalizado se a melhor solução não for alterada por um número determinado de iterações.

De certa forma, pode-se considerar que o PSO tem certa semelhança com os algoritmos genéticos (*Genetic Algorithms* – GA). No PSO, no entanto, ao invés de utilizar operadores genéticos, cada indivíduo (partícula) atualiza a sua posição baseado na sua própria experiência de busca e nas experiências e descobertas de outros indivíduos [MAGOULAS, *et al*, 2002]. A ação de adicionar o termo de velocidade à posição corrente para gerar a próxima posição é semelhante à operação de mutação dos algoritmos evolucionários. Porém, no PSO esta "mutação" é guiada pela própria experiência de "vôo" da partícula, além da experiência de "vôo" de todo o enxame. Em

outras palavras, pode-se considerar que o PSO possui uma mutação com uma "consciência", como apontado por [EBERHART, SHI, 1998].

Outra diferença entre PSO e GA é que o primeiro adota uma abordagem cooperativa na população de partículas envolvidas da resolução do problema, enquanto os algoritmos genéticos partem de uma filosofia fundamentalmente competitiva, ilustrada pela expressão "sobrevivência dos mais aptos".

4. SISTEMAS MULTIAGENTE

A Inteligência Artificial Distribuída (*Distributed Artificial Intelligence* – DAI) é um campo que vem evoluindo e se diversificando rapidamente desde o seu início na segunda metade da década de 1970. Ainda hoje representa uma promissora área de pesquisa e aplicação, característica pela sua base multidisciplinar, reunindo conceitos de campos como ciência da computação, sociologia, economia, administração, organização do trabalho e filosofia. Seu amplo escopo e até a sua multidisciplinaridade natural dificultam uma definição formal que contemple o campo em todos os seus aspectos, mas um ponto de partida pode ser obtido a partir do seguinte conceito [WEISS, 2000]:

> *"DAI é o estudo, construção e aplicação de sistemas multiagente, ou seja, sistemas nos quais diversos agentes inteligentes, interativos, perseguem algum conjunto de objetivos ou desempenham algum conjunto de tarefas."*

Nesta definição, "agentes" são entidades computacionais autônomas que percebem o seu ambiente através de sensores e agem sobre o mesmo por meio de efetuadores. Esta autonomia diz respeito à sua capacidade de, até certo ponto, ter controle sobre o seu comportamento e agir sem a intervenção humana ou de outros sistemas. O termo "inteligentes" está relacionado à capacidade que estes agentes possuem de perseguir seus objetivos e executar suas tarefas de forma a otimizar alguma métrica de desempenho adotada.

Ainda na definição acima, o termo "interativos" indica que os agentes podem ser afetados por outros agentes na perseguição de seus objetivos ou na execução de suas tarefas. Essa interação pode ser direta (através de comunicação explícita, uma linguagem compartilhada entre os agentes) ou indireta, quando os agentes apenas

observam o comportamento uns dos outros ou quando a ação de um agente modifica o ambiente, e esta mudança produz alteração no comportamento de outros agentes.

De acordo com [JENNINGS, *et al*, 1998], as principais características dos sistemas multiagente são as seguintes:

- cada agente possui informações incompletas e é restrito em suas capacidades;
- o controle do sistema é distribuído;
- os dados são descentralizados;
- o processamento de informações (computação) é assíncrono.

O conceito de sistemas multiagente está para a DAI assim como o conceito de agente individual está para a Inteligência Artificial (AI – *Artificial Intelligence*). Ambos (DAI e AI) lidam com aspectos computacionais de inteligência, mas a partir de diferentes pontos de vista e sob diferentes premissas [WEISS, 2000]. Enquanto a AI tem foco sobre "processos cognitivos" dentro de indivíduos, a DAI se concentra em "processos sociais" num grupo de indivíduos. A AI tradicional considera sistemas que têm um único centro interno de decisão e controle, enquanto a DAI considera sistemas onde o processamento e o controle são distribuídos.

Ainda, a AI tradicional tem como fontes de idéia, inspiração e metáfora campos como a psicologia e o behaviorismo, enquanto a DAI parte da sociologia e da economia. Assim, a DAI não é exatamente uma especialização da AI tradicional, e sim uma generalização desta [WEISS, 2000].

Os agentes podem ainda operar, dependendo da implementação computacional, de maneira assíncrona e paralela, o que pode resultar num aumento geral na velocidade de resolução dos problemas. Os sistemas multiagente também são, comparativamente, mais robustos que as implementações de controle centralizado, já que falhas em um ou vários agentes não torna o sistema geral necessariamente inútil, uma vez que os outros agentes ainda disponíveis podem assumir as tarefas dos inativos ou defeituosos.

4.1. Estratégias cooperativas em Inteligência Artificial Distribuída

Uma das grandes vantagens dos Sistemas Multiagente diz respeito à capacidade destas arquiteturas de se auto-organizarem e se adaptarem dinamicamente às modificações do ambiente sem a necessidade da interferência de um controle central ou de um operador externo. Porém, há um grande número de pesquisas no sentido de projetar agentes com inteligência elaborada, tentando emular processos cognitivos e decisórios quase humanos, capazes de negociar explicitamente a coordenação entre seus pares.

Essa abordagem, no entanto, reintroduz uma série de problemas do projeto de sistemas complexos. A ocorrência natural de sistemas de agentes extremamente simples (como populações de insetos e outros animais) oferece evidência de que este tipo de "recuo" não é necessário. São casos onde o comportamento do conjunto é significativamente mais complexo do que o de qualquer de seus indivíduos. Fato que já é conhecido há séculos, apesar de apenas recentemente estar sendo utilizado como inspiração para o projeto de sistemas artificiais. Como foi bem ilustrado em [PARUNAK, 1997]:

> *"O Rei Salomão, pensando nos complexos sistemas necessários para manter seu palácio e para a preparação para campanhas de guerra, escreveu 'Vá até a formiga...; considere seus métodos, e seja sábio. Mesmo sem um guia, supervisor, ou administrador, elas preparam sua comida no verão, e armazenam o alimento durante o tempo de colheita.'. A respeito da complexidade das organizações militares, ficava impressionado que 'os gafanhotos não têm rei, e no entanto todos avançam em batalhões'. Cerca de 3 mil anos depois, em 1994, um participante de um* workshop *da NCMS Virtual Enterprise comentou: 'Nós estamos acostumados a pensar que os* bugs[11] *são o problema da indústria de* software. *Agora suspeitamos que talvez sejam a solução!'".*

[11] Trocadinho com o termo "*bug*", inseto em inglês, nome utilizado comumente para descrever as falhas em programas de computador.

O foco primário da DAI é a coordenação como forma de interação, particularmente importante para atingir o objetivo ou para completar o conjunto de tarefas. E os dois padrões básicos e contrastantes de coordenação são competição e cooperação. Na competição, diversos agentes trabalham uns contra os outros porque seus objetivos são conflitantes. Na cooperação, os agentes trabalham em conjunto e congregam seus conhecimentos e capacidades para atingir um objetivo comum.

Enquanto os agentes competitivos tentam maximizar seus benefícios próprios às custas dos outros, e assim o sucesso de um implica no fracasso dos outros, os agentes cooperativos tentam alcançar como um time um objetivo que, isoladamente, agente algum seria capaz de alcançar, e portanto compartilham todos o mesmo resultado, seja ele o sucesso ou o fracasso [WEISS, 2000]. Neste trabalho, pela própria natureza da aplicação escolhida (a caça em "bando"), serão consideradas apenas as estratégias cooperativas.

Vários sistemas naturais possuem esta característica de auto-organização. A partir do reconhecimento, estudo e compreensão dos princípios envolvidos nestes exemplos, podemos construir sistemas artificiais que emulam as mesmas características [PARUNAK, 1997]. Este estudo pode ser realizado a partir da emulação[12] de hipotéticos comportamentos individuais num ambiente computacional. Se o comportamento geral do sistema emulado corresponder ao comportamento do sistema natural, os comportamentos individuais podem ser considerados modelos dos comportamentos naturais. Essa abordagem é mais sugestiva do que conclusiva, mas como o objetivo aqui é muito mais projetar sistemas artificiais do que explicar os naturais, este tipo de correspondência superficial já é suficiente.

As formigas, por exemplo, constroem trilhas que conectam seu formigueiro com fontes de alimento. Matematicamente, estas trilhas minimizam a energia necessária para que as formigas transportem alimentos para o formigueiro. A Teoria de Grafos possui uma classe de algoritmos para produzir este tipo de otimização, mas não temos notícia de nenhuma formiga que tenha utilizado algum destes algoritmos. O que ocorre é que a

[12] Na emulação, entidade de *software* imitam o comportamento (observado ou postulado) de entidades no domínio do problema. Simulação é uma classe mais ampla de técnicas, que incluem emulação e também modelos que capturam o comportamento agregado do sistema mas não das atividades dos membros individuais [PARUNAK, 1997].

trilha ótima emerge de ações simples por parte destes pequenos insetos. Cada formiga em busca de alimento segue basicamente o mesmo "programa", que consiste num conjunto de cinco regras:

1) Desviar de obstáculos.
2) Movimentar-se aleatoriamente, de preferência na direção de feromônios[13] nas proximidades. Se não captar nenhum feromônio, executar movimento Browniano, escolhendo cada passo a partir de uma distribuição uniforme sobre todas as direções. Sentido algum feromônio, continuar em movimento aleatório, mas agora a partir de uma distribuição de probabilidade que favoreça a direção do odor.
3) Se a formiga estiver carregando comida, produzir e exalar feromônio numa taxa constante à medida que se movimenta.
4) Se a formiga encontrar comida e não estiver carregando nada, pegar a comida.
5) Se a formiga estiver no formigueiro e estiver carregando comida, descarregar a comida.

O movimento aleatório fará com que a formiga acabe encontrando comida, desde que esteja a uma distância do formigueiro compatível com o alcance de sua movimentação. E acabará retornando também ao formigueiro. Aquelas que partiram em direções erradas morrerão de fome ou do ataque de inimigos naturais, mas contanto que exista comida na vizinhança do formigueiro, e enquanto houver formigas o suficiente para explorar o terreno, o alimento será encontrado [PARUNAK, 1997].

Como apenas as formigas que carregam comida depositam feromônios, e como só podem carregar comida após encontrar uma fonte de alimento, todas as trilhas de feromônios levam à fonte de comida. A partir do momento em que uma formiga "carregada" encontrar o caminho de volta para "casa", surgirá a primeira trilha que levará também ao formigueiro, e que estimulará o surgimento de outras. Como os feromônios evaporam, os caminhos que não são interessantes vão sumindo com o tempo, e aqueles mais interessantes vão se reforçando na medida em que outras

[13] Marcadores de odor que vários insetos e outros animais produzem, e que consiste numa espécie de comunicação.

formigas, ao seguir estas trilhas, pegam comida e retornam diretamente para o formigueiro, depositando feromônio em todo o caminho.

A trilha inicial certamente não será reta, mas a tendência das formigas de andar seguindo a direção dos feromônios produz atalhos. A trilha resultará então numa fusão de vários caminhos muito próximos; terá então uma certa "largura", que ficará cada vez mais estreita na medida em que for percorrida. O processo de otimização de caminho resultante não é intuitivamente óbvio a partir dos comportamentos individuais, mas sim emergente a partir da emulação.

Outro interessante exemplo de auto-organização vem de algumas espécies de cupins tropicais, durante a construção dos cupinzeiros[14]. Eles são capazes de construir elaboradas estruturas em forma de domo[15] que começam como pilares; durante a construção, estes pilares vão se aproximando uns dos outros até que seus topos se conectem, formando arcos. E arcos que se conectam resultam num típico domo. Como é freqüentemente observado que a invenção do arco foi um grande marco no desenvolvimento da arquitetura do homem civilizado, é de se estranhar como é possível que simples cupins possam ser capazes de produzir tal feito [KENNEDY, EBERHART, 2001]:

> *"Se nós tivermos a tarefa de construir um arco, nós provavelmente começaríamos a partir de um plano, ou seja, uma representação central de um objetivo e os passos necessários para executá-lo. Então, como o trabalho provavelmente dependerá do trabalho de mais de uma pessoa (a não ser que seja um arco muito pequeno), um time de operários será organizado, com o arquiteto ou alguém que compreenda o plano supervisionando os trabalhadores, instruindo-os sobre onde colocar o material, controlando o cronograma de construção dos pilares e do seu encontro. Nós somos tão dependentes do controle centralizado de funções complexas que às vezes é impossível compreender como a mesma*

[14] Verdadeiras obras arquitetônicas, atingindo às vezes mais de 5 metros de altura e 10 toneladas de massa.
[15] Parte superior de uma construção de forma côncava, também chamada de abóbada.

tarefa pode ser executada por um sistema distribuído, não centralizado."

Aparentemente, os cupins constroem um domo carregando um pouco de sujeira umedecida na boca e obedecendo as seguintes regras:

1) Se mover na direção da maior concentração de feromônio.
2) Depositar o que estiver carregando onde o odor for mais intenso.

Depois de algum movimento aleatório procurando por um campo de feromônio relativamente forte, os cupins começam a construir alguns pilares pequenos, localizados em lugares onde os cupins mais passaram recentemente, e portanto a concentração do odor é mais alta. O feromônio se dissipa com o tempo, portanto para que haja acumulação, o número de cupins deve exceder algum limiar: eles devem depositar feromônio mais rápido que a evaporação da substância.

A ascenção dos pilares é resultado de um processo *autocatalítico*[16]. À medida que os pilares crescem, a concentração de feromônios perto destes pilares aumenta. Um cupim ao se aproximar da área detecta o feromônio, e como existem múltiplos pilares e o cupim se dirige à área de maior concentração, é bem provável que acabe numa área entre dois pilares. Ele é atraído na direção de ambos, e escolhe arbitrariamente um ou outro. Como se aproxima do pilar pela região "entre pilares", é mais provável que escale o pilar pelo lado voltado ao pilar vizinho. Como conseqüência, o depósito de material tende a ser feito na face interna do pilar, e como cada um cresce com mais substância neste lado, quanto mais alto alcança mais se aproxima do outro pilar. O resultado é um arco.

Tanto no exemplo de construção de trilhas de formigas quanto de cupinzeiros, esteve envolvida uma forma indireta de comunicação através dos feromônios destes insetos, método batizado pelo entomologista P. P. Grassé [GRASSÉ, 1959] de *estigmergia*[17], que pode ser de dois tipos, de acordo com a intenção do "emissor" do sinal:

---

[16] Ciclo de retroalimentação positiva. Importante aspecto de muitos sistemas complexos, que permite a amplificação de efeitos aparentemente triviais em significativos.
[17] Comunicação através da alteração do estado do ambiente, de forma que afete o comportamento de outros para os quais o ambiente é um estímulo.

a) estigmergia baseada em "dicas": a mudança no ambiente fornece simplesmente uma dica para o comportamento de outros agentes; não há intenção de comunicação;
b) estimergia baseada em sinais: a mudança no ambiente se constitui numa sinalização, na prática, para os outros agentes; há intenção de comunicação.

Dentre os inúmeros outros exemplos encontrados na natureza de auto-organização cooperativa [PARUNAK, 1997], podemos citar o trabalho de classificação de larvas, ovos, casulos e comida feito pelas formigas dentro dos formigueiros, a divisão de tarefas entre as vespas, a coordenação de movimento de bandos de aves e peixes (*flocking behavior*) e a caçada de alces feita por lobos, exemplo que serviu de inspiração para este trabalho.

Um lobo sozinho não consegue matar um alce, que é maior, mais forte e conta com chifres que afugentam o agressor isolado. Reunidos em alcatéia, os lobos podem então cercar sua presa, de forma que um deles possa atacá-la pelas costas enquanto esta está ocupada com os outros. Essa tarefa, conhecida na literatura como problema da caça ou problema "predador-presa", foi durante alguns anos um grande desafio no campo da Inteligência Artificial Distribuída [PARUNAK, 1997].

4.2. Exemplos de aplicações em engenharia

Muitas aplicações industriais e comerciais são descritas na literatura [WEISS, 2000], muitas delas envolvendo tarefas complexas onde soluções otimizadas são capazes de produzir grande impacto econômico. Alguns exemplos são destacados a seguir:

- Otimização de manufatura industrial e processos de produção como planejamento de produção ou gerenciamento de cadeia de suprimentos, onde agentes representam diferentes células de manufatura ou mesmo companhias inteiras.

- Modelagem e otimização de sistemas de transportes internos, locais, regionais ou internacionais, onde agentes representam veículos de transporte ou mercadorias ou consumidores a serem transportados.
- Aperfeiçoamento do fluxo de tráfego urbano ou aéreo, onde agentes são responsáveis pela interpretação adequada de informações que chegam em diferentes estações de sensoreamento.
- Comércio eletrônico e planejamento de mercados eletrônicos, onde agentes "compradores" e "vendedores" compram e vendem mercadorias em nome de seus usuários.
- Monitoramento em tempo real e gerenciamento de redes de telecomunicações, onde os agentes são responsáveis pelo redirecionamento de chamadas, comutação de sinais e transmissão.
- Gerenciamento de informações em ambientes de informações como a Internet, onde múltiplos agentes são responsáveis pela coleta e filtragem de informações.
- Agendamento automático de reuniões, onde agentes operam em nome de seus usuários para acertar detalhes como local, horário e agenda.
- Análise de processos de negócios dentro de uma empresa ou entre empresas, onde agentes representam as pessoas ou departamentos distintos envolvidos nestes processos em estágios distintos e em diferentes níveis.
- Projeto e reengenharia de padrões de fluxo de controle e informações em organizações de larga escala, sejam elas naturais, técnicas ou híbridas, onde agentes representam as entidades responsáveis por estes padrões.
- Entretenimento eletrônico interativo, jogos de computador onde agentes animados equipados com diferentes personagens jogam uns contra os outros ou contra oponente humanos.
- Investigação e análise de aspectos sociais de inteligência e simulação de fenômenos sociais complexos como a evolução de funções, regras e estruturas organizacionais, onde agentes assumem o papel dos membros de uma sociedade natural sob consideração.

O que estas aplicações têm em comum é que elas apresentam pelo menos uma das seguintes características [BOND, GASSER, 1988]:

1) Distribuição inerente – Elas são inerentemente distribuídas no sentido de que os dados e as informações a serem processadas:
   - chegam em lugares geográficos distintos (distribuição espacial);
   - chegam em momentos diferentes (distribuição temporal);
   - são estruturados em aglomerados cujo acesso e uso requer familiaridade com diferentes linguagens (distribuição semântica);
   - são estruturados em aglomerados cujo acesso e uso requer diferentes capacidades perceptivas, funcionais e cognitivas (distribuição funcional).
2) Complexidade inerente – Elas são inerentemente complexas no sentido de que são muito "grandes" para serem resolvidas por um único sistema centralizado devido às limitações disponíveis em determinado nível de tecnologia de *hardware* ou *software*. Para que os sistemas centralizados possas dar conta dos requisitos das aplicações inerentemente complexas, eles devem ser ampliados na medida necessária, o que é geralmente bastante difícil, demorado e caro.

A aplicação abordada neste trabalho se enquadra dentro desta classe de problemas. Um sistema é projetado para atingir um objetivo que somente é possível através de uma ação coordenada de agentes, atuando em caráter cooperativo. A definição dos parâmetros da estratégia adotada pelo "time" de agentes é definida por uma técnica de otimização, que por sua vez utiliza um simulador para medir o desempenho das estratégias sugeridas. Por analogia, as demais aplicações pertencentes a esta mesma classe de problemas, e cujas soluções também dependam de simulação para a avaliação de alternativas podem, a princípio, adotar implementações que partam da mesma abordagem.

5. O JOGO DE CAÇA TRIDIMENSIONAL

O problema da caça (ou jogo de perseguição) é um tema clássico em Inteligência Artificial. Tal como proposto por [BENDA, *et al*, 1985], consiste em duas classes de agentes (predadores e presas, classicamente quatro predadores e uma única presa) dispostos numa grade retilínea (domínio plano e discreto), todos com a mesma velocidade.

O objetivo dos predadores é capturar a presa. Para cercá-la, cada predador deve ocupar uma "casa" adjacente à da presa na grade retilínea. A presa, por sua vez, "vence" o jogo se conseguir escapar do domínio do "tabuleiro" pelas suas bordas antes que possa ser capturada.

Nesta versão clássica, a movimentação dos agentes é bastante simples: a cada "passo" ou ciclo de simulação, cada agente pode se deslocar uma "casa" na direção vertical ou horizontal, desde que esta casa já não esteja ocupada. Em geral a presa é dotada de movimento aleatório, enquanto a estratégia dos predadores é o foco das abordagens de inteligência artificial.

De acordo com [MANELA, CAMPBELL, 1995], este tipo de problema serve como um excelente *benchmark* para avaliação comparativa de diferentes abordagens de inteligência artificial, tanto utilizando controle central, local ou distribuído. Pela natureza do problema, cada indivíduo influencia e é influenciado por todo o sistema e, dado que o objetivo não pode ser alcançado por um agente isoladamente (na definição clássica, são necessários os quatro predadores para cercar a presa), é natural a emergência de estratégias cooperativas.

Este trabalho propõe uma evolução no jogo em sua forma clássica, de forma que a caça e a perseguição extrapolem o plano discreto e passem a se desenvolver num domínio

contínuo e tridimensional. Um outro ponto de evolução consiste numa formulação mais elaborada para os comportamentos e as estratégias tanto dos predadores quanto da presa.

5.1. Do plano discreto ao espaço contínuo tridimensional

As simplificações contidas na forma clássica do jogo de perseguição permitem que toda a ênfase fique concentrada nas estratégias, uma vez que a metáfora do domínio plano e discreto, aliada à característica de movimentação simplista não dá muita margem às diferenças de implementações nem às peculiaridades de simulação.

No entanto, as estratégias geradas a partir deste cenário não encontram muita aplicabilidade em situações mais próximas da realidade. A discretização do domínio, por exemplo, é completamente desprovida de sentido se comparada às situações observadas na natureza. É claro, sempre é possível dizer que se trata simplesmente de uma questão de resolução; basta que se aumente suficientemente a resolução para que mesmo o modelo discreto possa se tornar adequado; aliás, qualquer simulação executada num computador digital sofre, em maior ou menor grau, com o efeito de discretização.

O problema é que na abordagem clássica a própria movimentação está intimamente relacionada com a discretização. Aumentada a resolução, a idéia de ocupar "a próxima casa adjacente" fica sem sentido. A velocidade dos agentes na abordagem tradicional está também intrinsecamente associada à resolução, e portanto deve observar certos limites práticos. Tratar o jogo de caça no domínio contínuo significa, portanto, dotá-lo de maior verossimilhança com as situações reais.

A questão da dimensão do espaço de movimentação do problema está mais relacionada à abrangência e versatilidade do cenário de simulação. O modelo clássico está preso ao

plano, e possui associado a ele uma série de metáforas de perseguição entre animais terrestres.

Ao transpor o domínio para o espaço tridimensional, esta restrição é eliminada e torna-se possível modelar virtualmente todos os cenários de perseguição encontrados na natureza (peixes, pássaros, etc.) e gerados pela tecnologia (aviões de caça, submarinos, etc.), incluindo aí aplicações de simulação (industrial, militar, etc.) e entretenimento (animação, jogos de computador, etc.).

É claro que, uma vez transposta a barreira entre o universo bidimensional e o tridimensional, é natural que a complexidade das estratégias de caça também aumente. Assim como é razoável imaginar que uma caça ou uma perseguição num ambiente tridimensional não é trivialmente percebida e interpretada por humanos, que ao longo de milhares de anos de evolução participaram e adaptaram os seus circuitos cognitivos às caçadas e perseguições que podem ser modeladas no domínio plano (afinal, somos animais terrestres, com limitada capacidade de locomoção e autonomia em ambientes aquáticos). Nosso próprio sistema de visão não ajuda muito a perceber o espaço ao nosso redor em todas as direções, a não ser que abusemos da flexibilidade dos nossos pescoços.

É intuitivo que ao migrar do plano para o espaço a presa passe a levar uma certa vantagem. Ela ganha uma nova dimensão para escapar dos seus captores, e assim é de se esperar que seja comprometido o equilíbrio entre ela e os seus predadores. Aliás, o estabelecimento de um número fixo de predadores representa, na prática, uma desnecessária limitação do problema, por desperdiçar a oportunidade de análise da sensibilidade do tamanho do "time" de caçadores na definição das estratégias. Para viabilizar este tipo de análise, neste jogo o número de predadores é um parâmetro livre, configurável.

5.2. Comportamento da presa e dos predadores

Com as alterações efetuadas no jogo, foi necessário que alguns conceitos fossem redefinidos. A própria tarefa dos predadores, que era cercar a presa, ocupando as quatro "casas" adjacentes, deixa de ter sentido num domínio contínuo. A definição de captura foi reformulada como sendo a situação em que pelo menos um dos predadores alcance a presa. Tecnicamente, que invada uma espécie de "bolha vital", uma pequena esfera (outro parâmetro livre do jogo) relacionada ao volume corporal da presa e com centro nela.

A presa, por sua vez, tem como objetivo impedir que os predadores a capturem durante determinado tempo. O critério clássico para a fuga, escapar pelas bordas do "tabuleiro", deixou de ter sentido por estar sendo utilizado como domínio todo o espaço R^3, que é ilimitado. A limitação de tempo se deve à uma série de circunstâncias práticas, dentre elas o fato de que o desempenho no jogo servirá, em última análise, como função objetiva de um algoritmo de otimização; e deve haver, naturalmente, um critério de parada previsto para os casos em que os predadores são incapazes de ter sucesso na captura da presa.

O comportamento da presa neste jogo de perseguição 3D no espaço contínuo também evoluiu em relação à sua versão clássica. Ela passa a ser controlada por uma máquina de estados finitos (*Finite State Machine* – FSM). Máquinas de estados finitos são estruturas de representação de comportamento e controle de uso comum em trabalho de simulação em computação evolutiva [Miranda, et al, 2004; Neves, Netto, 2003]. A máquina projetada para descrever o comportamento da presa possui três estados, conforme apresentado na Fig. 5.1:

- Passeio Livre: Estado inicial do jogo. Neste estado a presa se comporta como na versão clássica, possuindo movimento aleatório em baixa velocidade.

- Fuga: A presa, ao perceber que está sendo caçada, pois pelo menos um predador está dentro da sua "bolha de percepção" com velocidade acima de determinado

limiar ou entrou, em qualquer velocidade, em sua "bolha de proximidade", passa a se movimentar na sua velocidade máxima, numa direção calculada a partir da posição dos dois predadores mais próximos. Se as condições que determinaram a entrada neste estado deixarem de existir, a presa volta para o estado de Passeio Livre.

- Capturada: Neste estado, pelo menos um predador já invadiu a "bolha vital" da presa. Ela é considerada capturada e é cessado o seu movimento. O jogo é encerrado.

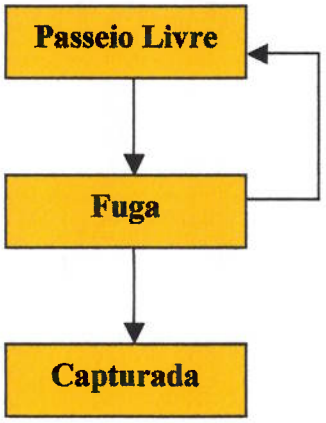

Figura 5.1 – Máquina de Estados Finitos (*Finite State Machine*) da presa.

Durante o estado de fuga, a estratégia da presa é pré-definida, ou seja, a sua direção de movimento é calculada pela soma dos vetores associados às direções da presa aos dois predadores mais próximos (orientados no sentido de fuga, evidentemente), ponderados pelo inverso de suas distâncias à presa.

Os predadores têm o seu comportamento controlado por uma FSM (Fig. 5.2) composta pelos três estados descritos a seguir:

- Caça: Estado inicial do jogo. O predador se movimenta numa velocidade que têm direção e módulo definidos pela sua estratégia. E é exatamente a síntese desta estratégia que será o alvo deste trabalho, o produto da ferramenta de síntese que envolve um simulador e um algoritmo de otimização.

- Perseguição: O predador entra neste estado quando a presa entra em modo de fuga, percebendo a aproximação deste ou mesmo de outro predador. Quando em perseguição, o predador tem sua direção de movimentação apontada exclusivamente pela posição da presa, e se movimenta em sua velocidade máxima, não havendo a partir de então mais nenhum tipo de estratégia.

- Captura: O predador invadiu a "bolha vital" da presa e procedeu a sua captura. O jogo é encerrado.

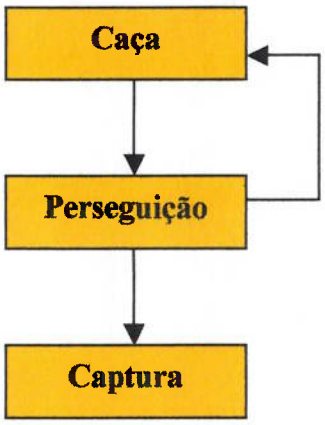

Figura 5.2 – Máquina de Estados Finitos (*Finite State Machine*) dos predadores.

5.3. Parâmetros do jogo de caça

Dadas as suas regras gerais, o jogo de caça tridimensional ainda tem certa margem de versatilidade. Várias combinações podem ser testadas e comparadas, dependendo de como são definidos certos parâmetros livres, que junto com alguns parâmetros de simulação compõem o cenário de configuração do jogo:

- **número de predadores** (n_{pred}): tamanho do conjunto de predadores; mínimo de dois "caçadores".

- **duração máxima do jogo** ($t_{máx}$): tempo máximo que os predadores têm para capturar a presa; passado este tempo, a presa é considerada vencedora.

- **velocidade de passeio da presa** (v_{presa1}): velocidade adotada pela presa enquanto no estado de passeio.

- **velocidade de fuga da presa** (v_{presa2}): velocidade adotada pela presa no estado de fuga.

- **distância inicial** (D_0): distância entre a presa e cada um dos predadores no início do jogo.

- **raio da "bolha de percepção" da presa** (R_{perc}): raio da esfera, com centro na presa, dentro da qual um predador com velocidade acima de determinado limiar (v_{crit}) chama a atenção e dispara na presa a mudança do estado de passeio para o de fuga (Fig. 5.3).

- **raio da "bolha de proximidade" da presa** (R_{prox}): raio da esfera, com centro na presa, dentro da qual um predador, entrando com qualquer velocidade, dispara na presa a mudança do estado de passeio para o de fuga; é menor que R_{perc} (Fig. 5.3).

- **raio da "bolha vital" da presa** (R_{vital}): raio da esfera, com centro na presa, dentro da qual um predador é considerado em contato físico com a presa, procedendo a captura e encerrando o jogo; é menor que R_{prox} (Fig. 5.3).

- **velocidade de perseguição dos predadores** (v_{pred2}): velocidade adotada pelos predadores em estado de perseguição.

- **velocidade crítica de aproximação dos predadores** (v_{crit}): velocidade (limiar) a partir da qual um predador pode ser percebido pela presa caso esteja dentro da sua "bolha de percepção".

- **coeficiente de inércia do simulador** ($C_{inércia}$): coeficiente (entre 0 e 1) para simular efeito de inércia, tanto para os predadores quanto para a presa, a cada passo da simulação. É aplicado de acordo com a seguinte expressão:

$$\vec{v}(t) \leftarrow C_{inércia} \cdot \vec{v}(t-1) + (1 - C_{inércia}) \cdot \vec{v}(t) \qquad (5.1)$$

É de se esperar que, para cada conjunto (coerente) de parâmetros livres, a ferramenta de síntese de estratégias apresente um resultado diferente, desde que as circunstâncias provocadas por estes parâmetros nas condições de caça sejam suficientemente distintas.

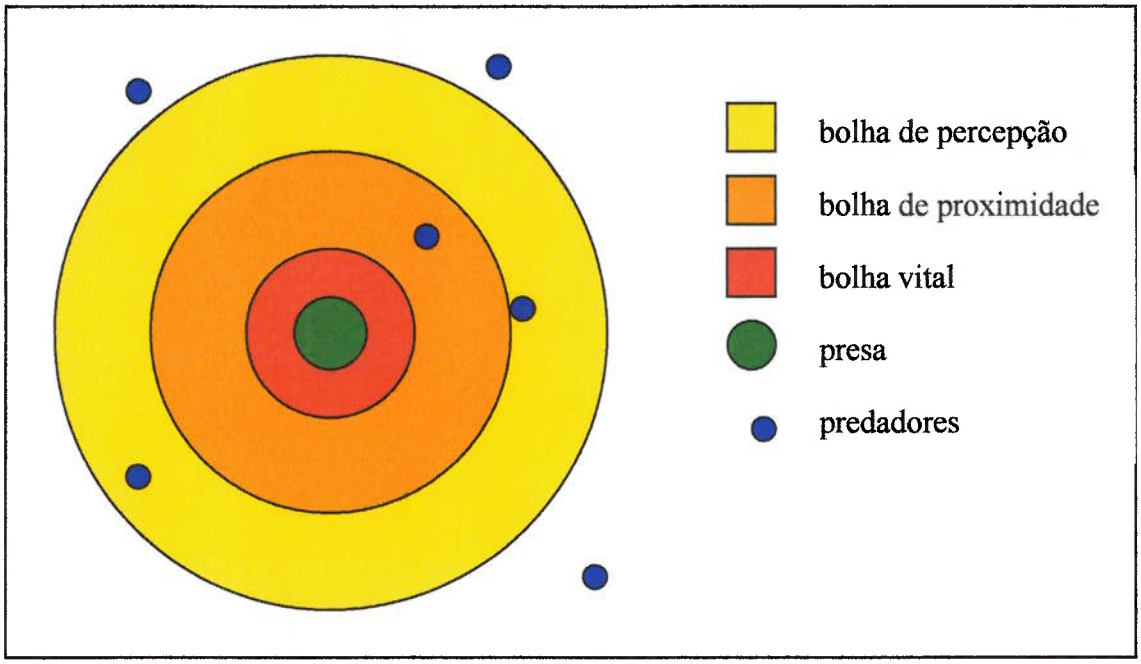

Figura 5.3 – Bolhas de percepção, proximidade e vital da presa.

5.4. As estratégias dos predadores

Conforme já apresentado em capítulos anteriores, vários comportamentos aparentemente sujeitos a um rigoroso controle central são, de fato, gerados pelo efeito combinado de regras simples, que requerem pouca capacidade sensorial e tomada de decisão. Em [REYNOLDS, 1999], um trabalho muito influente sobre simulação de comportamento de bando em pássaros, Reynolds assumiu que os pássaros eram guiados

por três forças locais: evitar as colisões, equiparar a velocidade e se aproximar do centro do bando. Implementando apenas estas três regras, as simulações de Reynolds apresentavam comportamento bastante realista.

A idéia amplamente difundida de que os bandos de pássaros possuem um líder coordenando as manobras foi descartada pela pesquisa do biólogo Frank Heppner, da Universidade de Rhode Island. Após filmar a movimentação de pássaros, Heppner concluiu que não há, de fato, nenhum líder nestes bandos. Qualquer pássaro poderia parecer como liderando uma manobra, dependendo do momento. O bando mantinha um tipo de equilíbrio dinâmico, mas sem nenhum controle central [HEPPNER, GRENANDER, 1990]. Um pássaro no meio do bando é capaz de partir numa direção para buscar alimento que nem viu, assim como também começar a fugir de predadores que nem tinha percebido, apenas seguindo o movimento do bando. É como se a sua capacidade sensorial fosse ampliada pelo bando, aumentando assim as suas chances de sobrevivência.

No problema elaborado para este trabalho, os predadores não precisam se movimentar em bando, mas para aumentar as suas chances de sobrevivência devem ser capazes de se auto-organizar para cercar e capturar uma presa e assim garantir o seu alimento, sem contar, evidentemente, com nenhum tipo de comunicação direta durante a tarefa. Em [PARUNAK, 1997] é observado que equipes de policiais usam o rádio para coordenar suas manobras de cerco, mas lobos não usam *walkie-talkies* na hora em que saem em bando pra caçar um alce. Através de algumas regras baseadas em pistas visuais sobre as posições do alce e dos outros lobos, eles devem ser capazes de cercar o alce sem nenhuma forma explícita de comunicação ou negociação (nada de "eu vou pelo lado norte; por que você não vai pelo sul ?").

Todos os predadores devem compartilhar (como parte da definição do problema) a mesma estratégia de caça, que não envolve memória nem comunicação direta entre os agentes. Tomando como base o universo sensorial dos predadores no momento da caça, a estratégia deve, no máximo, levar em consideração os seguintes parâmetros:

- direção da presa ($\vec{q}_{presa}$)

- direção do predador (vizinho) mais próximo ($\vec{q}_{viz}$)
- distância da presa (D_{presa})

Como a referência cinemática para os predadores são as suas próprias posições, os dois primeiros parâmetros devem ser calculados da seguinte forma:

$$\vec{q}_{presa} = \frac{\vec{P}_{presa} - \vec{P}_{predador}}{\left|\vec{P}_{presa} - \vec{P}_{predador}\right|} \tag{5.2}$$

$$\vec{q}_{viz} = \frac{\vec{P}_{vizinho} - \vec{P}_{predador}}{\left|\vec{P}_{vizinho} - \vec{P}_{predador}\right|} \tag{5.3}$$

Onde $\vec{P}_{presa}$, $\vec{P}_{predador}$ e $\vec{P}_{vizinho}$ são os vetores de posição (em relação ao centro de coordenadas do sistema) da presa, do predador em questão e do seu vizinho mais próximo, respectivamente.

A única capacidade de decisão permitida aos predadores no jogo é controlar a sua própria movimentação, ou seja, mudar dinamicamente a sua velocidade vetorial em função dos parâmetros fornecidos pelo seu sistema sensorial ($\vec{q}_{presa}$, $\vec{q}_{viz}$ e D_{presa}).

A formulação de estratégia desenvolvida para este trabalho obedece à seguinte expressão:

$$d = \min\left(\frac{D_{presa}}{D_0}, 1\right) \tag{5.4}$$

$$\vec{v}_{pred}(d) = v_{pred2} \cdot d^{X_1} \cdot \frac{\left(d^{X_2} \cdot \vec{q}_{presa} + X_3 \cdot d \cdot \vec{q}_{viz}\right)}{\left|\left(d^{X_2} \cdot \vec{q}_{presa} + X_3 \cdot d \cdot \vec{q}_{viz}\right)\right|} \tag{5.5}$$

Onde:

$\vec{v}_{pred}(d)$: velocidade (vetorial) do predador em função da distância (normalizada) à presa;

v_{pred2} : velocidade (módulo) adotada pelos predadores no estado 2 (estado de perseguição);

D_0 : distância entre a presa e o predador no início do jogo;

X_1, X_2 e X_3 : parâmetros de decisão (fornecidos pela ferramenta de síntese de estratégias).

A variável X_1 está relacionada à influência da distância ao predador no módulo de sua velocidade; X_2 reflete a importância da direção da presa na composição vetorial da direção da velocidade, enquanto X_3 participa, na mesma ponderação, representando a influência da direção do vizinho mais próximo.

Para respeitar a formulação do problema, estas variáveis devem ser procuradas dentro dos seguintes domínios contínuos:

$$X_1 \in [0;10]$$
$$X_2 \in [0;10]$$
$$X_3 \in [-10;10]$$

As estratégias ficam então definidas por triplas ordenadas (X_1, X_2, X_3) que definem o comportamento da movimentação dos predadores. A formulação apresentada permite não-linearidade e um amplo leque de possibilidades tanto para o módulo quanto para a composição vetorial da velocidade em função da distância dos predadores à presa.

Evidentemente, o mesmo problema comportaria outras inúmeras formulações de estratégia. Ampliando a capacidade sensorial, por exemplo, a velocidade dos predadores poderia ser função também da distância ao vizinho mais próximo. Ou mesmo mantendo o conjunto definido neste trabalho, poderiam ser formuladas expressões matemáticas bastante diversas.

Embora fosse tentador analisar mais detidamente o problema e equacionar da maneira mais conveniente, esta não seria uma abordagem muito justa. Afinal, na maior parte dos

problemas reais de inteligência artificial distribuída, os agentes também possuem uma série de limitações, tanto sensoriais quanto de processamento cognitivo. E em todos os casos, o projeto dos comportamentos e estratégias é obrigado a levar em consideração as restrições práticas dos agentes. A formulação meio "engessada" proposta neste trabalho tem o papel de representar as restrições práticas que fatalmente surgiriam em aplicações do mundo real.

6. SÍNTESE DE ESTRATÉGIAS PELO ENXAME DE PARTÍCULAS

Já foi apresentada a importância de um ambiente de simulação para avaliar o desempenho de estratégias que, de outra forma, não poderiam ser testadas em nenhum tipo de expressão analítica. O efeito combinado[18] e a questão temporal[19] são alguns dos fatores que, neste problema, apontam para o uso de um simulador.

O diagrama de fluxo de informações do sistema (Fig. 6.1) apresenta, de forma esquemática, a estrutura proposta para uma ferramenta de síntese de estratégias. Inicialmente o problema, que no caso deste trabalho é a definição de uma estratégia para que os predadores tenham sucesso dentro das regras definidas para o jogo de caça, foi modelado de forma conveniente. A partir deste modelo, foi implementado um ambiente de simulação que, ao lado de um algoritmo de otimização, compõe uma ferramenta de síntese que fornecerá, ao final do processo, uma solução satisfatória para o problema inicial.

Durante o processo de síntese de estratégias, o simulador troca continuamente informações com o otimizador. O otimizador, seguindo o seu algoritmo (enxame de partículas), define alguns "pontos" no espaço de soluções e submete, uma a uma, estas soluções-candidatas ao simulador, que as recebe e interpreta como um conjunto de parâmetros de entrada para a simulação. Ao final da simulação, o resultado desta é enviado de volta ao otimizador, que interpreta esta informação como a função objetiva da solução-candidata que havia enviado. Processa esta informação dentro da rotina do algoritmo e envia a próxima solução-candidata, reiniciando um ciclo que só se encerra quando o otimizador encontra algum dos seus critérios de parada.

---

[18] A mesma estratégia sendo usada por um conjunto de predadores, cada um influenciando na movimentação do outro, numa reação em cadeia.

[19] A caça só pode ser verificada na observação dos efeitos da estratégia ao longo do tempo.

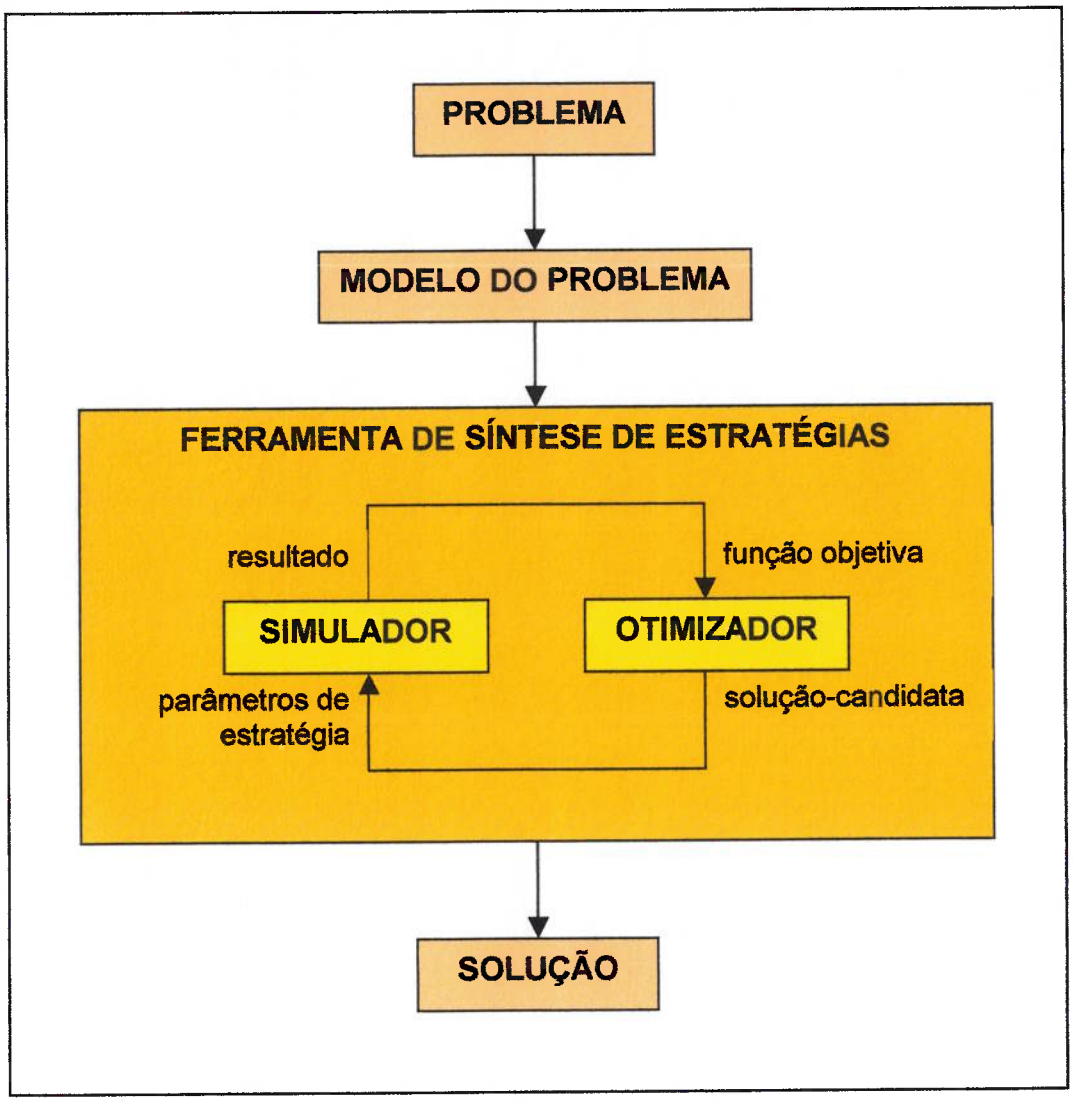

Figura 6.1 – Diagrama de fluxo de informações.

Dentro do simulador, existe um conjunto de predadores que se movem num espaço físico tridimensional (embora virtual) e o resultado da caçada depende do comportamento emergente resultante da interação entre os agentes. O movimento de cada agente descreve uma trajetória num espaço físico. Já no otimizador, o algoritmo se desenvolve como se, metaforicamente, um conjunto de partículas se movesse num espaço de soluções[20] (neste caso também tridimensional, definido pelo trio de parâmetros) e o resultado da otimização depende do comportamento emergente resultante da interação entre as partículas. Metaforicamente, o movimento das partículas descreve trajetórias no espaço cognitivo multidimensional da inteligência de enxames.

[20] O espaço de soluções pode assumir qualquer dimensionalidade, dependendo da modelagem do problema.

Para efeito de síntese de estratégias, um simulador que calculasse "no escuro" toda a dinâmica de movimentação ao longo do jogo e fornecesse como saída o resultado da caçada seria mais que suficiente. No entanto, até para acompanhar o processo e verificar se o simulador está de fato operando dentro das regras propostas, é importante que a ferramenta de simulação tenha como característica a capacidade de visualização do jogo.

Mas ao migrar do plano para o espaço tridimensional, o jogo de caça trouxe um problema de visualização. Isto se deve a uma certa dificuldade de acompanhar, ao longo da simulação, a movimentação espacial dos predadores e da presa em tempo real.

Esta dificuldade em perceber rapidamente o posicionamento e a movimentação dos agentes do jogo no espaço tridimensional foi levada em consideração no projeto do simulador, que é bastante dedicado à questão da visualização. E foi uma das principais razões para que se decidisse pela implementação de um programa de computador especialmente desenvolvido para este fim. Os pacotes comerciais, apesar de já contarem com um arsenal matemático que facilitaria a modelagem do problema, não permitiriam a construção de mecanismos de visualização específicos para a análise deste problema. Daí surgiu a necessidade da implementação do programa *PREDADOR*.

6.1. O ambiente de simulação

Para ilustrar a simulação do jogo de caça dentro do programa *PREDADOR*, todos os agentes foram representados por peixes. Estes foram escolhidos por serem animais que, ao contrário dos humanos, estão sujeitos às perseguições que só podem ser modeladas a partir de domínios tridimensionais.

Neste programa, foi construída então uma classe "Peixe" que contém todas as propriedades e métodos comuns aos agentes. Assim como duas sub-classes, "Predador" e "Presa", herdeiras dos métodos e propriedades da classe "Peixe", e diferenciadas com

suas características individuais (velocidades, "gatilhos" de transição para os estados das máquinas de estados finitos, etc.).

Tendo cada agente um objeto[21] correspondente, estes são inicializados no ambiente de simulação da seguinte maneira:

a) a presa é posicionada na origem do sistema de coordenadas globais (0,0,0);
b) os predadores têm suas posições sorteadas (uma primeira manifestação do caráter estocástico da simulação) dentro de uma superfície esférica de raio igual à distância inicial $\mathbf{D_0}$ (distância entre a presa e os predadores no início do jogo, um dos parâmetros livres do jogo);
c) todos os agentes têm as suas orientações iniciais aleatórias (outro fator não-determinístico).

Por uma questão de coerência visual, mas que não interfere absolutamente nos resultados das simulações, todos os peixes têm os seus modelos geométricos orientados de forma que sua "frente" esteja apontada para a orientação do seu modelo correspondente, e o vetor que aponta para o seu "lado de cima" esteja, sempre que possível, contido num plano vertical (mesmo plano que os vetores orientação do peixe e aceleração da gravidade).

Na simulação, as velocidades dos predadores e da presa possuem um termo de "momento", que emulam uma espécie de "inércia" de movimento mesmo sem o processamento das equações dinâmicas, que necessariamente envolveriam forças e massas dos agentes. Este mecanismo impede que os agentes (no caso os peixes) possam se movimentar de forma implausível no cenário de simulação, executando curvas e manobras visivelmente fora da realidade dinâmica que deveria imperar numa situação real.

O programa *PREDADOR* foi inicialmente projetado para atualização gráfica a uma taxa de 20 Hz (exibindo a movimentação dos peixes na tela a 20 quadros por segundo). Posteriormente, para reduzir o tempo de processamento da ferramenta de síntese, foi

[21] Usando o sentido empregado pelo paradigma computacional de Programação Orientada a Objetos.

implementada uma taxa alternativa de 80 Hz. Neste modo especial, um segundo real equivale a quatro segundos de simulação, como se fosse um vídeo exibido em *fast forward*.

6.2. Interface e visualização

O programa *PREDADOR* foi concebido de forma a apresentar uma interface bastante simples e intuitiva (Fig 5.2).

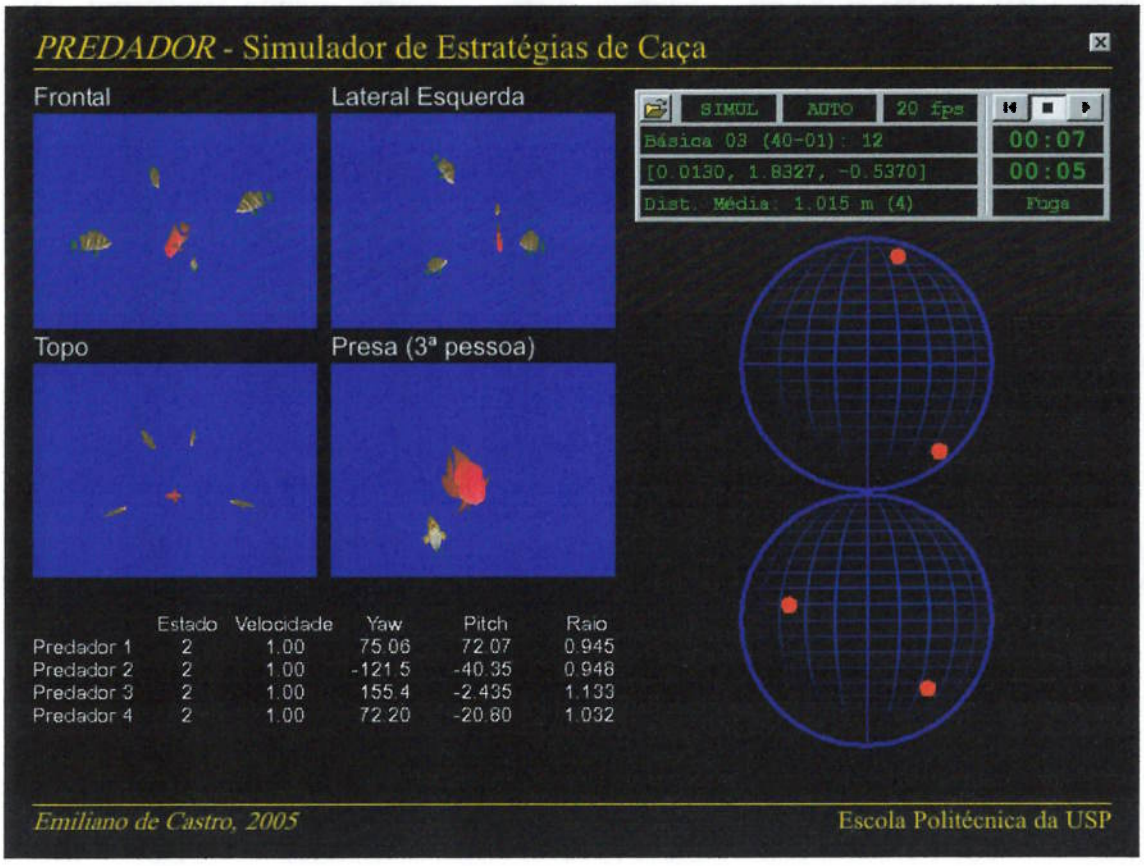

Figura 6.2 – Tela do programa *PREDADOR*.

Todos os botões e *displays* foram agrupados num painel de controle, do qual um de seus botões permite que o usuário carregue um arquivo contendo um dos três tipos de configurações possíveis:

1) Simulação: contém o cenário de configuração do simulador e uma estratégia individual (para testes e avaliação); a qualquer instante, é possível interromper e reiniciar a simulação.
2) Otimização: contém o cenário de configuração do simulador e o cenário de configuração do otimizador (PSO); a qualquer instante, é possível interromper e reiniciar a otimização.
3) Animação: a partir de uma adaptação do arquivo de configuração de otimização, reproduz uma animação com a movimentação das partículas resultante do processo de otimização; a qualquer instante, é possível interromper e reiniciar a animação.

Figura 6.3 – Detalhe do painel de controle do programa *PREDADOR*.

No exemplo ilustrado na Fig. 6.3, a estratégia utilizada na simulação é a correspondente à partícula de número 1 da 40ª iteração (40-01), a linha logo abaixo exibe os seus parâmetros de estratégia (X_1, X_2 e X_3) e a linha inferior apresenta a distância média dos predadores em perseguição (o número de predadores em perseguição é apresentado entre parênteses).

Há dois cronômetros à direita no painel. O primeiro indica o tempo de caça[22] e o segundo, disparado só quando o primeiro pára, marca o tempo de perseguição[23]. Há também um campo indicando o estado atual da presa (no exemplo, em fuga).

A parte superior do painel apresenta o estado de alguns modos do simulador:

- **Modo de operação**: O programa pode funcionar nos modos de simulação (apenas testando uma única estratégia), otimização (gerando estratégias e

[22] Todos os agentes se encontram no primeiro estado de suas FSMs.
[23] Todos os agentes se encontram no segundo estado de suas FSMs.

avaliando) e animação (apresentando o resultado visual da movimentação das partículas ao longo do processo de otimização).

- **Piloto da presa**: pode ser automático ("AUTO"), onde a presa é regida pela regra de fuga descrita anteriormente, ou em *game mode* ("MANUAL"), modo em que o usuário pode "pilotar" a presa utilizando o teclado ou *joystick*. Comutável a partir da tecla "M". Apenas uma possibilidade para que o usuário possa "sentir na pele" o problema da presa ao ser caçada por um conjunto de predadores.

- **Freqüência do simulador**: pode ser 20 ou 80 Hz, comutáveis a partir da tecla "F".

Os estados dos predadores (codificados de 1 a 3, na mesma seqüência em que foram apresentados neste texto), assim como suas velocidades (em m/s) e suas coordenadas cartesianas (em metros) podem ser monitorados a partir da tabela exibida quando é pressionada a tecla "T".

Não é tarefa fácil interpretar a movimentação dos peixes através da mera observação das suas coordenadas cartesianas. Uma forma mais simples pode ser obtida pressionando a tecla "Y", que também exibe a tabela mas numa versão com coordenadas esféricas[24], um pouco mais amigável. Nesta tabela os ângulos estão em graus, e o raio (em metros) pode ser interpretado como sendo a distância de cada predador à presa, que é uma informação relevante para acompanhar a caçada.

Apesar de oferecer informações detalhadas sobre posições e velocidades dos peixes, acompanhar a simulação através da observação das tabelas é tarefa sobre-humana. Para facilitar a compreensão por parte do usuário o programa simulador foi dotado de quatro janelas de visualização que exibem as projeções perspectivas de câmeras de acompanhamento (Fig. 6.4).

[24] Calculadas em relação a um centro de coordenadas fixo à presa.

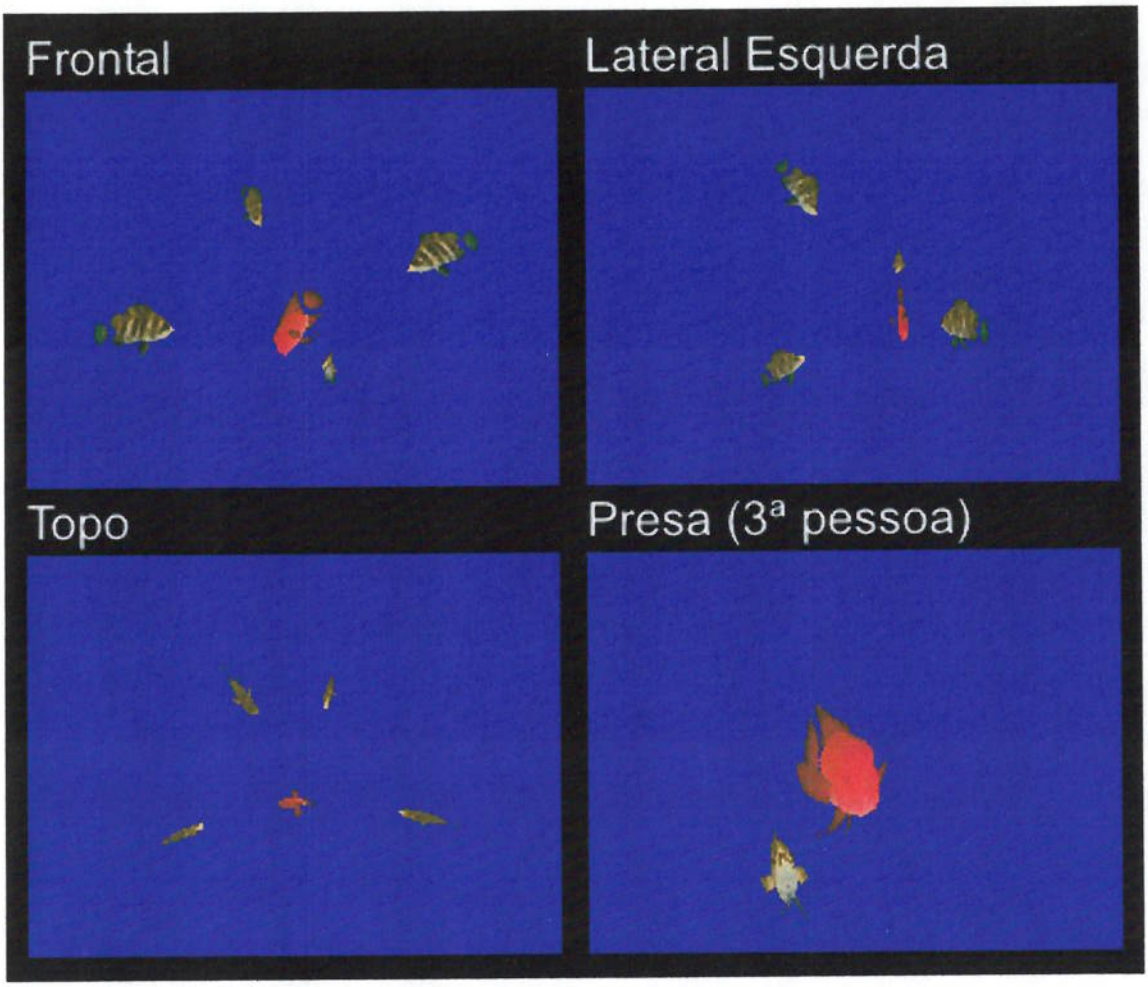

Figura 6.4 – Janelas das câmeras de acompanhamento.

Das quatro janelas, três têm posicionamento automático (frontal, lateral esquerda e topo). São sempre alinhadas com os eixos x, y e z, respectivamente, e são livres para se movimentar (sem perder o alinhamento com os eixos) de forma a manterem um enquadramento ótimo para visualizar a presa e seus caçadores.

A quarta câmera de acompanhamento (a inferior direita) pode ser escolhida dentre as seguintes alternativas:

- câmera focalizando a presa em 3ª pessoa, localizada um pouco à frente, acima e à direita dela (tecla "0");
- câmeras "grudadas" aos predadores (no máximo seis câmeras), em 1ª pessoa (tecla "1" a "6");
- câmera "grudada" à presa, em 1ª pessoa (tecla "7").

O programa *PREDADOR* tem ainda alguns recursos disponíveis para facilitar a visualização e verificar as condições do simulador, acionados via teclado:

- Grande-angulares (tecla "G"): apresenta duas câmeras adicionais, grande-angulares, representando o que seria uma visualização da presa através de dois "olhos" virtuais, um de cada lado do corpo, capazes de exibir projeções perspectivas com ângulos de abertura de 150°.
- "Congelamento" dos predadores (tecla "C"): permite que os predadores fiquem imóveis para que a presa possa "navegar" tranqüilamente entre eles e testar novas possibilidades dentro da estratégia vigente.
- "Bolhas" da presa (teclas "B" e "N"): exibem respectivamente a "bolha de percepção" e a "bolha de proximidade" da presa como duas esferas semitransparentes, facilitando a visualização, por parte do usuário, de quais predadores se encontram em cada região de sensibilidade da presa.

Mesmo com todos os recursos de visualização implementados, ainda não é trivial para o usuário acompanhar a movimentação dos predadores. Ele se encontraria em apuros se estivesse no lugar da presa, o que é facilmente percebido ao ativar o controle manual da presa (tecla "M"). Ao pilotar a presa a partir do teclado ou do *joystick*, o usuário se depara com a dificuldade de se orientar no espaço e fugir dos predadores a partir dos elementos gráficos disponíveis até então. As tabelas não são práticas para a orientação, e as câmeras de acompanhamento só ajudam se consideradas em conjunto, e não é nada fácil processar as informações visuais das quatro (ou mesmo três) imagens em tempo real.

Por trás desta dificuldade provavelmente se esconde uma razão evolutiva. O usuário (que é humano) não teve, ao contrário do que provavelmente ocorreu com os peixes, o seu sistema cognitivo adaptado para processar informações visuais vindo de todas as direções. Ao dirigir um carro, por exemplo, contamos com a ajuda de espelhos retrovisores, para suprir ao menos em parte a imensa região invisível atrás das nossas cabeças. Mesmo as posições acima e abaixo não são bem visualizadas sem alguma rotação proporcionada pelo pescoço.

Para tentar fazer uma "ponte" entre estas diferenças de processamento visual foi implementado o que se decidiu chamar de "radar 3D", acionado a partir da tecla "R". Este radar consiste de dois discos (Fig. 6.5), sendo o superior utilizado para representar a parte frontal da presa e o inferior a parte traseira, funcionando como se fosse um retrovisor.

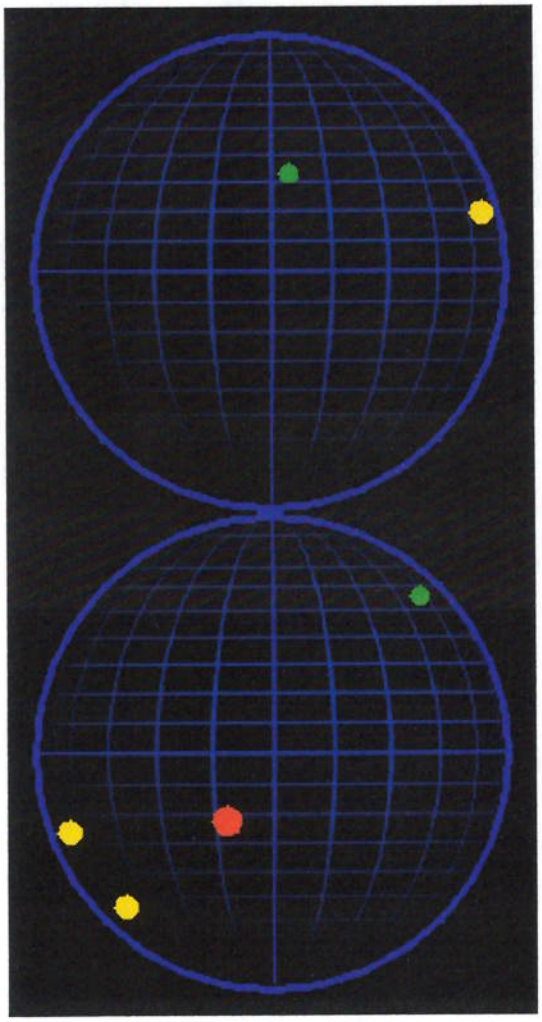

Figura 6.5 – O "radar 3D".

A partir das coordenadas esféricas disponíveis na tabela (que na tela do simulador é exibida logo abaixo das janelas das câmeras de acompanhamento), as posições dos predadores são classificadas e mapeadas de acordo com as seguintes regras:

a) Os predadores são representados no radar por pequenos círculos (bolinhas), cujos tamanhos e cores indicam a sua classificação em relação à distância da presa: pequenos e verdes para predadores distantes, que ainda não entraram na

"bolha de percepção" da presa; médios e amarelos, para predadores que já entraram em modo de perseguição; e grandes e vermelhos para predadores que estão dentro da "bolha vital" da presa (ou seja, que capturaram a presa).

b) Numa analogia com o sistema de coordenadas geográficas, a coordenada correspondente ao ângulo de "*Yaw*" determina qual o "meridiano" em que cada predador será representado. De 90° a -90° são representados no disco superior, da esquerda pra direita, enquanto que os ângulos de 90° a 180° e depois de -180° a -90° ficam no disco inferior, também da esquerda pra direita.

c) Seguindo ainda a analogia geográfica, a coordenada correspondente ao ângulo de "*Pitch*" determina qual o "paralelo" em que será representado o predador: 90° no "Pólo Norte", -90° no "Pólo Sul" e 0° na "Linha do Equador".

A utilização deste radar facilita enormemente a interpretação da caçada e da perseguição por parte do usuário. Após um curto período de adaptação, o usuário facilmente percebe que, uma vez que pretende fugir dos predadores, o mais interessante é manter os predadores na parte traseira (pois o peixe só nada para frente). E isso pode ser facilmente traduzido na tarefa de manter as "bolinhas" do radar no disco inferior (que representa a vista traseira).

Desta forma, o usuário em pouco tempo passa a "pilotar" a presa com o foco da sua atenção no centro do disco superior do radar (que representa a direção em que está se locomovendo), procurando se distanciar ao máximo de eventuais "bolinhas" que surjam neste disco, e dando atenção especial às "bolinhas" amarelas, que representam os predadores mais próximos antes de uma eventual captura.

6.3. Otimização integrada no programa *PREDADOR*

O programa *PREDADOR*, ao importar um arquivo contendo a descrição do cenário de configuração de otimização (arquivo texto contendo valores para todas as variáveis

livres da simulação e da otimização), está pronto para iniciar (a partir do comando do usuário pelo botão de *play*) o algoritmo do PSO. Ele processa, na seqüência, todas as simulações para todo o conjunto de partículas, atualiza os dados sobre a otimização e parte para uma nova iteração. À medida que vai processando as iterações, o programa vai registrando no próprio arquivo os resultados. Assim, é possível interromper a otimização e retomá-la posteriormente a partir do ponto onde parou.

O algoritmo de otimização, "embutido" dentro do programa PREDADOR, utiliza para avaliar o resultado da simulação a seguinte regra, que serve como função objetiva:

a) Para estratégias que conseguem capturar a presa dentro do tempo disponível (a variável livre $t_{máx}$), o resultado da simulação é:

$$f(X_1, X_2, X_3) = t_{caça} + t_{pers} \tag{6.1}$$

Onde $t_{caça}$ é o tempo utilizado na caça e t_{pers} o tempo gasto na perseguição (ambos em segundos), que são exibidos nos dois cronômetros do painel de controle.

b) Para estratégias que *não* conseguem capturar a presa dentro do tempo disponível, o resultado da simulação é:

$$f(X_1, X_2, X_3) = t_{máx} \cdot \left(1 + \frac{D_{média}}{D_0}\right) \tag{6.2}$$

Onde $t_{máx}$ é o tempo máximo permitido para a simulação (em segundos), $D_{média}$ é a distância média dos predadores à presa no final da simulação e D_0 a distância inicial entre os predadores e a presa (ambas em metros).

Além das informações de configuração de simulação, necessárias para as simulações que avaliarão as estratégias geradas pelo PSO, o algoritmo de otimização também lê no arquivo de configuração os seguintes parâmetros:

- **Máximo de Iterações**: número máximo de iterações do PSO; é o critério de parada utilizado.

- **Número de Partículas**: quantidade de partículas do PSO.

- **Número de Simulações por Partícula**: quantidade de simulações efetuada para cada estratégia (partícula); a função objetiva final é a média do desempenho em todas as simulações.

- **Tamanho da Vizinhança**: número de partículas do subconjunto considerado como "vizinhança" de cada partícula (incluindo ela mesma) para a comparação entre os melhores resultados individuais já obtidos.

- ϖ : momento de inércia do PSO.

- $\varphi_{1_{max}}$: valor máximo do coeficiente ("peso") da componente "cognitiva" da velocidade de cada partícula (relacionada ao seu melhor resultado individual).

- $\varphi_{2_{max}}$: valor máximo do coeficiente ("peso") da componente "social" da velocidade de cada partícula (relacionada ao melhor resultado da sua vizinhança).

- V_{max} : valor máximo de saturação da velocidade das partículas em cada dimensão considerada.

7. RESULTADOS

Foram realizados dezenas de ensaios de otimização, variando tanto o cenário de configuração de simulação, para analisar diferentes "regras" para o jogo de caça tridimensional, quanto a configuração de otimização, para estudar a sensibilidade dos parâmetros da otimização por enxame de partículas.

O parâmetro "número de simulações por partícula" se revelou, logo nos ensaios preliminares, de fundamental importância dentro do processo de otimização. Em alguns testes considerando apenas uma única simulação por partícula, era comum que algumas partículas apresentassem desempenhos de difícil reprodução, resultantes de inicializações do simulador extremamente favoráveis (todos os predadores já praticamente cercando a presa, por exemplo). Como a inicialização da posição dos predadores é estocástica, esse é um fenômeno difícil de ser evitado. E que tinha como conseqüência o fato de que estes resultados ficavam registrados no algoritmo e "puxavam", em certa medida, o enxame para uma solução muitas vezes longe de ser a mais adequada, mas que teve "sorte" quando foi avaliada.

Muito embora o algoritmo do PSO seja robusto o bastante para escapar destas armadilhas, o seu desempenho fica bastante prejudicado, uma vez que várias iterações são desperdiçadas enquanto a influência destes resultados excepcionais não enfraquece. Ao aumentar esse número de 1 para 5, essas situações foram praticamente eliminadas, e como a média dos desempenhos das simulações é que é considerada como função objetiva da partícula, essa repetição acaba funcionando como uma espécie de filtro contra os resultados decorrentes de condições aleatórias extraordinárias.

Não foram adotados critérios de parada baseados na análise da evolução do melhor resultado, ou mesmo aqueles onde um determinado desempenho já é bom o suficiente. Pela curiosidade em analisar, nos ensaios realizados, a convergência do algoritmo do

PSO, as otimizações foram conduzidas até determinado número máximo de iterações (50 ou 100, dependendo do ensaio), após alguns testes preliminares que apontaram convergência com menos de 50 iterações.

O tempo total de otimização, considerando este critério de parada que é o número máximo de iterações, é naturalmente função de uma série de parâmetros, que podem ser divididos em dois grupos: de simulação e de otimização. No primeiro grupo, o mais evidente é a duração máxima de simulação (45 segundos na maioria dos ensaios). No segundo, são preponderantes o número máximo de iterações (50 ou 100), o número de partículas (foram feitos ensaios com 10 e 20) e o número de simulações por partícula (5 na maior parte dos ensaios).

Foram feitas otimizações com diversos ajustes de parâmetros, e pode-se considerar que boa parte das estratégias foi sintetizada num período entre 7 e 12 horas.

Uma análise do comportamento de convergência do PSO, aliada a uma compreensão aprofundada da natureza do problema, pode levar a um ajuste de parâmetros de otimização mais eficiente. O que infelizmente não é o caso na maioria dos exemplos desta classe de problemas, devido à sua complexidade inerente. Assim, parece indicado buscar um equilíbrio, um compromisso entre a qualidade de solução aceitável e o tempo disponível para o processo de otimização.

O parâmetro relativo à duração máxima de simulação deve ser encarado também sob outra perspectiva, além daquela em que produz impacto no tempo total de síntese de estratégias. Ele é intrinsecamente ligado ao processo de caça, e sua adequação depende muito das relações entre as velocidades dos peixes e os raios das "bolhas" de sensibilidade da presa. Tem também papel fundamental na definição da função objetiva. Um tempo de duração máxima muito reduzido pode impedir que algumas estratégias inicialmente não muito interessantes se desenvolvam, mas que abririam o caminho para soluções mais otimizadas. Já uma duração muito longa reduziria este problema, mas elevaria talvez exageradamente o tempo total de otimização para um patamar muito elevado. Como conseqüência, sugere-se que uma série de simulações preliminares, ensaios de adaptação, sejam realizados num primeiro momento, para que o usuário possa "sentir" o impacto dos parâmetros de simulação na dinâmica de caça.

O programa PREDADOR possui um modo de simulação em alta velocidade (80 Hz), capaz de reduzir a um quarto o tempo total de simulação. Mas isso só é possível se a capacidade de processamento do computador encarregado da execução do programa tiver condições de suportar essa velocidade. Um truque adotado ao longo de algumas otimizações foi desabilitar a interface visual proporcionada pelas câmeras de acompanhamento, tabelas e o radar 3D, liberando assim o processador de uma série de funções não essenciais para o funcionamento da ferramenta de síntese.

Foi criado um cenário de simulação básico (Cenário 1), de referência, testado com uma série de ensaios preliminares. A partir deste, foram ensaiadas variações com o objetivo de detectar a sensibilidade de alguns dos parâmetros. Este cenário possui a seguinte configuração:

Tabela 7.1 – Configuração do cenário básico (Cenário 1).

n_{pred}	= 6	(número de predadores)
$t_{máx}$	= 45 s	(duração máxima do jogo)
v_{presa1}	= 0,2 m/s	(velocidade de passeio da presa)
v_{presa2}	= 1,0 m/s	(velocidade de fuga da presa)
D_0	= 10 m	(distância inicial)
R_{perc}	= 5 m	(raio da "bolha de percepção" da presa)
R_{prox}	= 3 m	(raio da "bolha de proximidade" da presa)
R_{vital}	= 0,3 m	(raio da "bolha vital" da presa)
v_{pred2}	= 1,0 m/s	(velocidade de perseguição dos predadores)
$v_{crít}$	= 0,6 m/s	(veloc. crítica de aproximação dos predadores)
$C_{inércia}$	= 0,95	(coeficiente de inércia)

Foram testadas algumas configurações do otimizador para gerar a melhor estratégia para este cenário de simulação básico, tomando como base valores sugeridos pela literatura da área [KENNEDY, EBERHART, 2001]. Os resultados obtidos estão apresentados na Tabela 7.2.

Tabela 7.2 – Estratégias geradas para o cenário básico (Cenário 1).

	CfgOtim A	**CfgOtim B**	**CfgOtim C**	**CfgOtim D**
Número de Partículas	10	10	10	20
Núm. de Simul. por Partícula	1	5	5	5
Tamanho da Vizinhança	5	5	5	5
Momento de Inércia (ϖ)	1,0	1,0	1,0	1,0
$\varphi_{1_{max}}$	0,5	0,5	0,4	0,7
$\varphi_{2_{max}}$	0,2	0,2	0,2	0,4
V_{max}	1,0	2,0	2,0	2,0
Tempo de caçada da melhor estratégia gerada (s)	11,6[25]	10,4	11,4	10,2

Uma rápida análise da Tabela 7.2 mostra que os diferentes ajustes do PSO levaram a parâmetros de estratégia com desempenhos muito próximos, todos proporcionando caçadas muito rápidas. A configuração A foi a que mais demorou para atingir um resultado satisfatório, o que deve ser creditado ao já citado efeito prejudicial do reduzido número de simulações de partícula deste ajuste (apenas uma simulação).

O Gráfico 7.1 permite uma análise da convergência dos diferentes conjuntos de parâmetros do PSO utilizados. A configuração A teve a convergência mais lenta, enquanto B e C, que possuíam ajustes muito parecidos, acabaram apresentando curvas de convergência muito similares. Já a configuração D mostrou a mais rápida convergência, o que pode ser atribuído ao maior número de partículas, capaz de "varrer" o universo de soluções com maior eficiência.

Os parâmetros de estratégia gerado nestes ensaios são triplas ordenadas muito parecidas, indicando "pontos" no universo de soluções muito próximos uns dos outros. A configuração D, por exemplo, gerou como melhor estratégia (pela partícula de número 14, na 50ª iteração) aquela capaz de capturar a presa após uma caçada de, em média, 10,2 segundos, e definida pelos seguintes parâmetros:

[25] Valor obtido na iteração de número 163.

$X_1 = 0{,}0753$

$X_2 = 1{,}6763$

$X_3 = -0{,}2653$

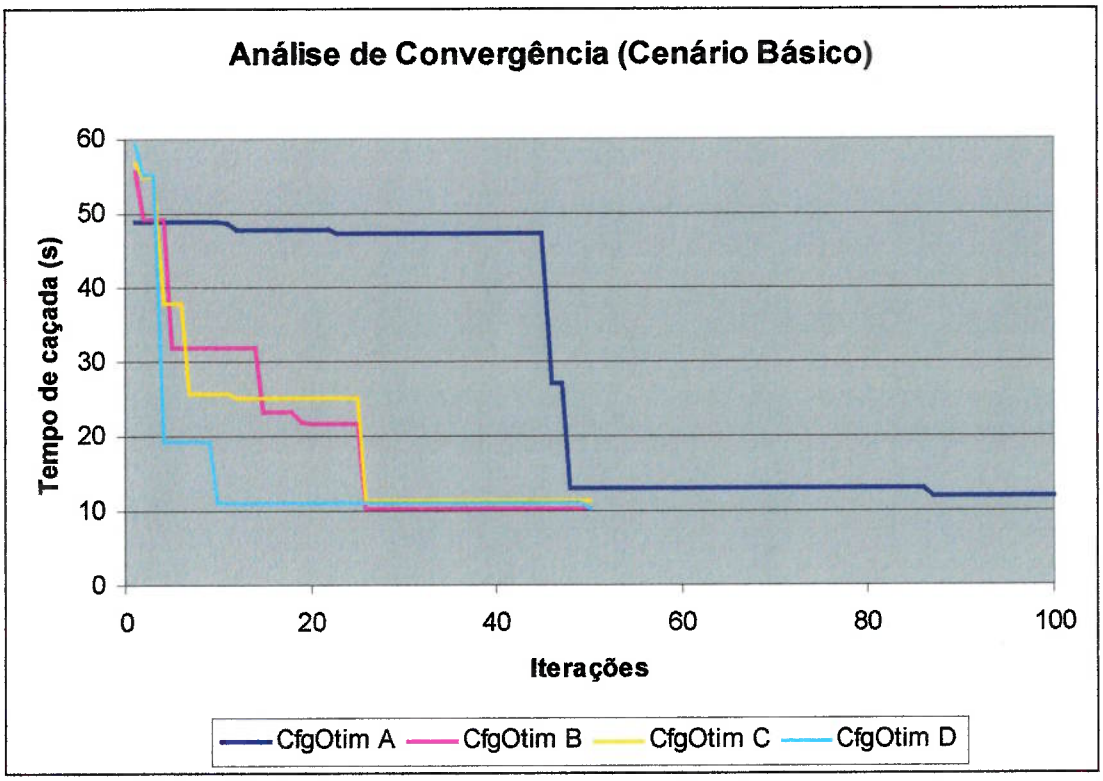

Gráfico 7.1 – Análise de convergência.

Pela natureza da fórmula da estratégia, não é tarefa simples visualizar o papel de cada parâmetro na definição do movimento de um predador[26]. Ainda assim, é possível traçar algum tipo de análise a partir de certas correlações.

Se considerarmos a fórmula da velocidade dos predadores:

$$\vec{v}_{pred}(d) = v_{pred_2} \cdot d^{X_1} \cdot \frac{\left(d^{X_2} \cdot \vec{q}_{presa} + X_3 \cdot d \cdot \vec{q}_{viz}\right)}{\left|\left(d^{X_2} \cdot \vec{q}_{presa} + X_3 \cdot d \cdot \vec{q}_{viz}\right)\right|} \qquad (5.1)$$

podemos concluir que o módulo da velocidade obedece à seguinte expressão:

[26] E ainda menos do grupo de predadores, dado o fenômeno pouco previsível que é a emergência.

$$\frac{|\vec{v}_{pred}(d)|}{v_{pred2}} = d^{X_1} \tag{5.2}$$

Desta expressão pode-se extrair uma relação para associar o comportamento da velocidade dos predadores em função da distância (normalizada) até a presa, que é o resultado da influência de X_1 (Gráfico 7.2). Vale lembrar que v_{pred2} é a velocidade máxima dos predadores e que:

$$d = \min\left(\frac{D_{presa}}{D_0}, 1\right) \tag{5.3}$$

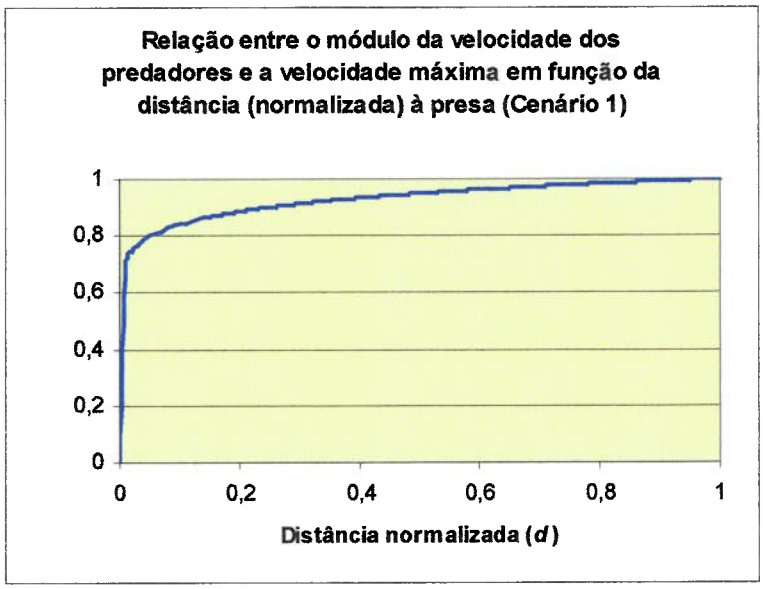

Gráfico 7.2 – Influência de X_1 no Cenário 1.

Analogamente, pode-se analisar as influências de X_2 e X_3 nos "pesos" das direções da presa e do predador mais próximo, respectivamente, na composição vetorial da velocidade de um predador (Gráficos 7.3 e 7.4).

A direção da velocidade dos predadores obedece a uma soma vetorial ponderada das direções da presa e do predador (vizinho) mais próximo, e portanto não é simples imaginar as influências de forma isolada. Uma melhor maneira de analisar o "peso" relativo entre estas duas direções (e que resulta da influência combinada de X_2 e X_3) pode ser representada pela divisão:

$$\frac{\left(d^{X_2}\right)}{\left|X_3 \cdot d\right|} \tag{5.4}$$

que é, naturalmente, função da distância normalizada (Gráfico 7.5).

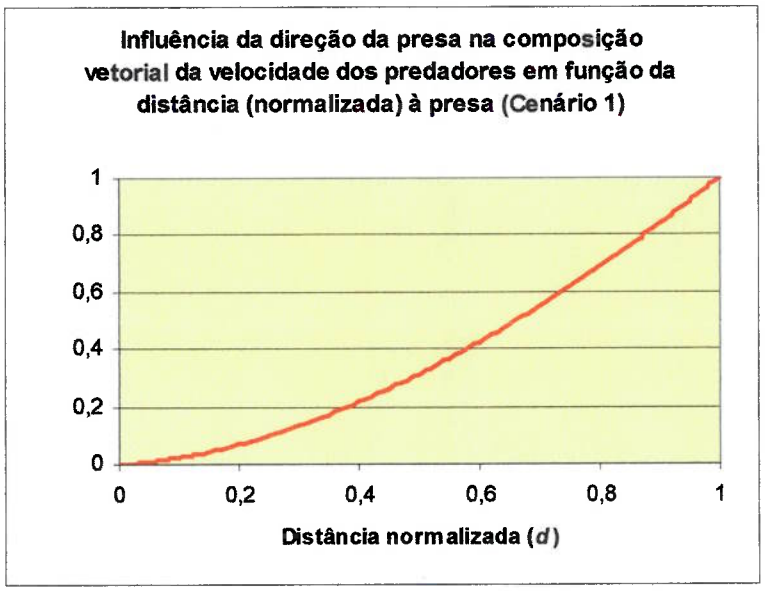

Gráfico 7.3 – Influência de X_2 no Cenário 1.

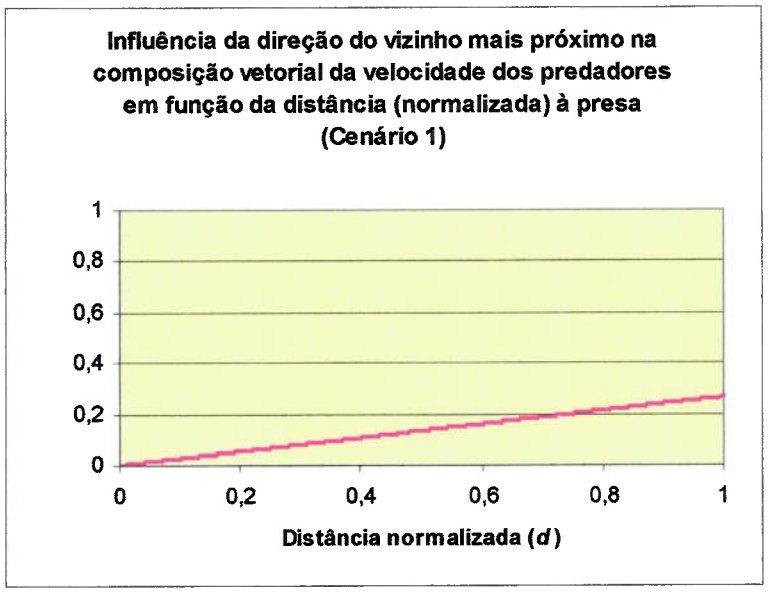

Gráfico 7.4 – Influência de X_3 no Cenário 1.

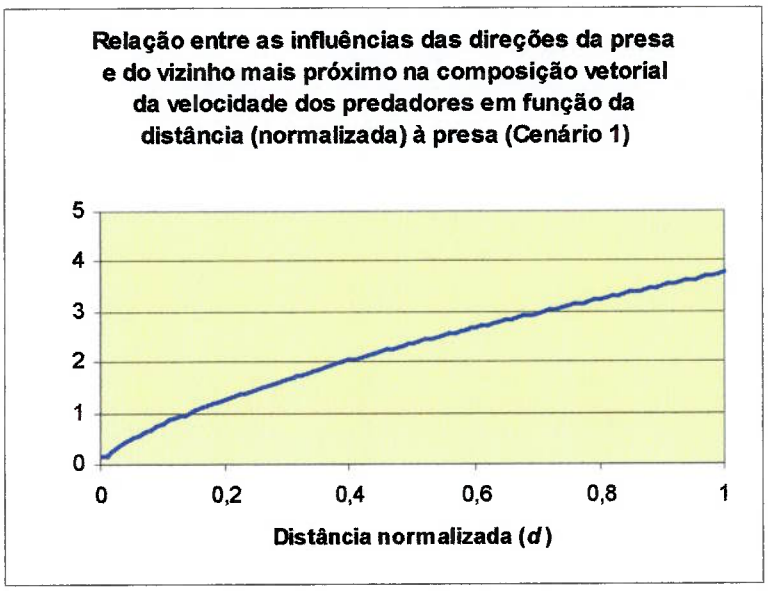

Gráfico 7.5 – Influências combinadas de X_2 e X_3. no Cenário 1.

Foram realizados ensaios semelhantes com outras configurações para o cenário de simulação, com o objetivo obter informações sobre a sensibilidade das estratégias às diversas variáveis livres do jogo de caça.

O Cenário 2 consiste numa variação com todos os parâmetros iguais ao cenário básico, com a diferença de que ao invés de 6 predadores, este novo cenário conta apenas com 3. Diferentes ajustes de otimização (nos moldes dos ensaiados para o cenário básico) resultaram em estratégias também com resultados muito semelhantes.

Novamente para ficar no exemplo, uma das configurações gerou como melhor estratégia (pela partícula de número 1, na 34ª iteração) aquela capaz de capturar a presa após uma caçada de, em média, 16,2 segundos, e definida pelos seguintes parâmetros:

$X_1 = 0{,}1069$
$X_2 = 1{,}1535$
$X_3 = -0{,}8577$

Os Gráficos 7.6 e 7.7 apresentam uma idéia da influência dos parâmetros na estratégia gerada. É evidente que não têm, no entanto, a mesma capacidade de demonstração da visualização da caçada no Programa *PREDADOR*. As diferenças entre as estratégias

geradas para o Cenário 1 e o Cenário 2 podem ser evidenciadas nas curvas das relações entre as influências das direções da presa e do vizinho.

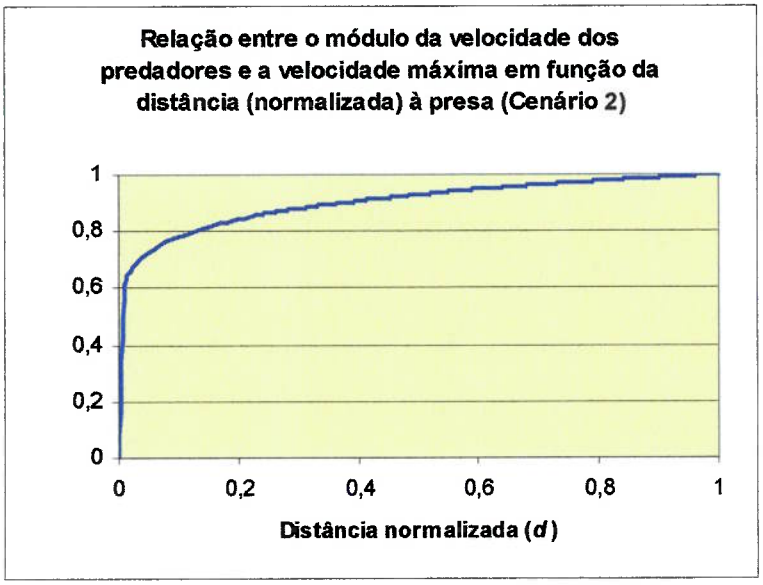

Gráfico 7.6 – Influência de X_1 no Cenário 2.

Para o Cenário 1, a estratégia privilegia fortemente a direção da presa, enquanto que no Cenário 2, com 3 predadores, a estratégia estabelece um maior equilíbrio entre esta direção e a do vizinho mais próximo (como pode ser observado na curva quase constante da relação entre as influências do Gráfico 7.7). O que pode ser explicado da seguinte forma: com um time mais "enxuto", os predadores são forçados a dar mais atenção ao distanciamento mútuo como forma de organizar um cerco à presa.

Numa segunda variação do cenário básico, foi construído o Cenário 3, com 4 predadores e raios das bolhas de percepção e proximidade iguais a 2 metros (no cenário básico, valiam 5 e 3 metros, respectivamente). Também foram utilizadas diferentes configurações de otimização, que resultaram em estratégias muito semelhantes.

Uma das configurações gerou como melhor estratégia (pela partícula de número 1, na 51ª iteração) aquela capaz de capturar a presa após uma caçada de, em média, 11,4 segundos, e definida pelos seguintes parâmetros:

$X_1 = 0{,}1068$

$X_2 = 2{,}1879$

$X_3 = -0{,}1549$

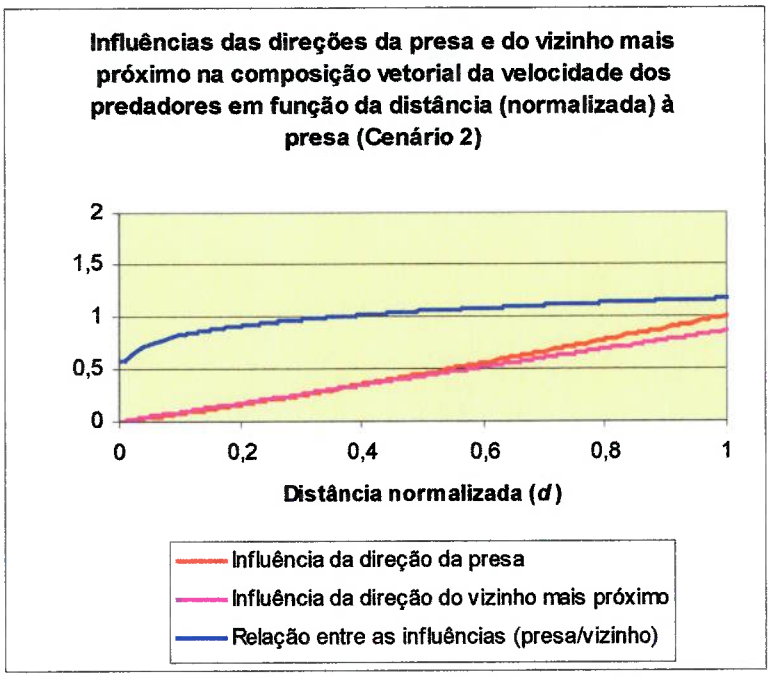

Gráfico 7.7 – Influências de X_2 e X_3 no Cenário 2.

Os Gráficos 7.8 e 7.9 representam a influência dos parâmetros na estratégia gerada. Novamente, uma comparação entre a estratégia do Cenário 3 e aquela gerada para o cenário básico (Cenário 1) pode ser feita a partir das curvas das relações entre as influências das direções da presa e do vizinho.

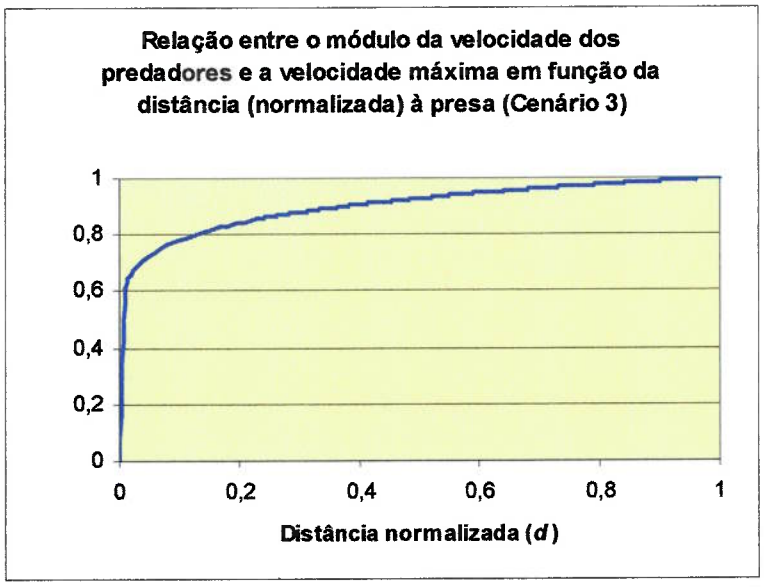

Gráfico 7.8 – Influência de X_1 no Cenário 3.

Enquanto a estratégia do Cenário 2 praticamente equilibra as influências entre as direções da presa e do vizinho mais próximo, a estratégia gerada para o Cenário 3 propõe uma ponderação muito mais elevada para a direção da presa, ainda mais desigual que no caso do Cenário 1. E que pode ser explicada da seguinte maneira: embora também tenha um time menor (4 predadores) que o time do Cenário 1 (6 predadores), este cenário não depende muito de um cerco bem organizado à presa; os reduzidos raios das bolhas de percepção e proximidade (2 metros) fazem com que a presa só perceba que está sendo caçada no final do processo, quando uma ação evasiva já tem poucas chances mesmo contra um cerco não muito bem armado.

Este cenário tem um importante papel de validação da formulação matemática elaborada para representar as estratégias. Embora o Cenário 3 não dependa do cerco, tem curva de módulo de velocidade (dependente de X_1) quase igual à do Cenário 2, e o forte desbalanceamento das influências entre as direções da presa e do vizinho mais próximo mostram que existe uma sensibilidade de X_2 e X_3 em relação aos raios das bolhas de percepção e proximidade.

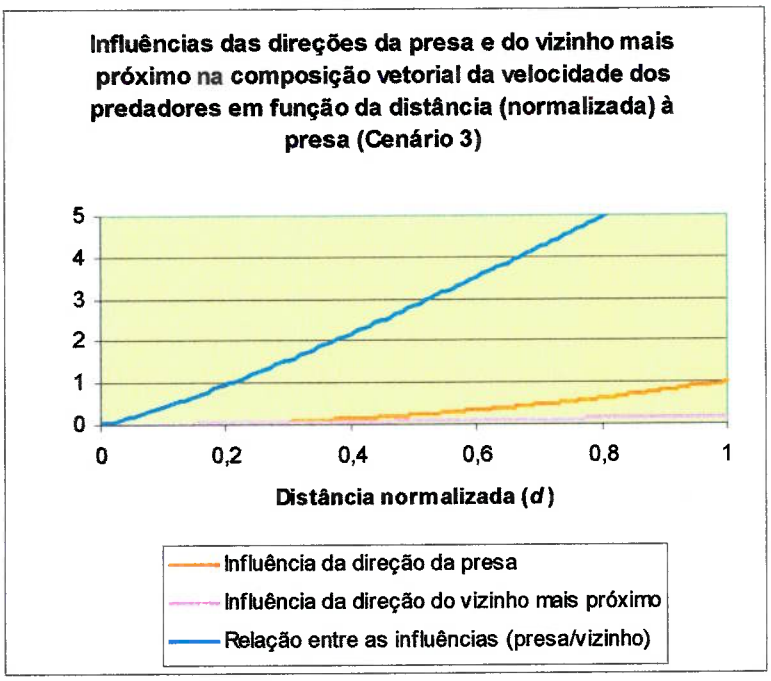

Gráfico 7.9 – Influências de X_2 e X_3 no Cenário 3.

Os parâmetros gerados para os 3 cenários podem ser comparados na Tabela 7.3. E no Gráfico 7.10 é apresentada uma comparação das relações de influências entre as direções da presa e do vizinho mais próximo.

Tabela 7.3 – Parâmetros das estratégias geradas para os 3 cenários.

	X_1	X_2	X_3	Tempo médio de caçada
Cenário 1	0,0753	1,6763	-0,2653	10,2 s
Cenário 2	0,1069	1,1535	-0,8577	16,2 s
Cenário 3	0,1068	2,1879	-0,1549	11,4 s

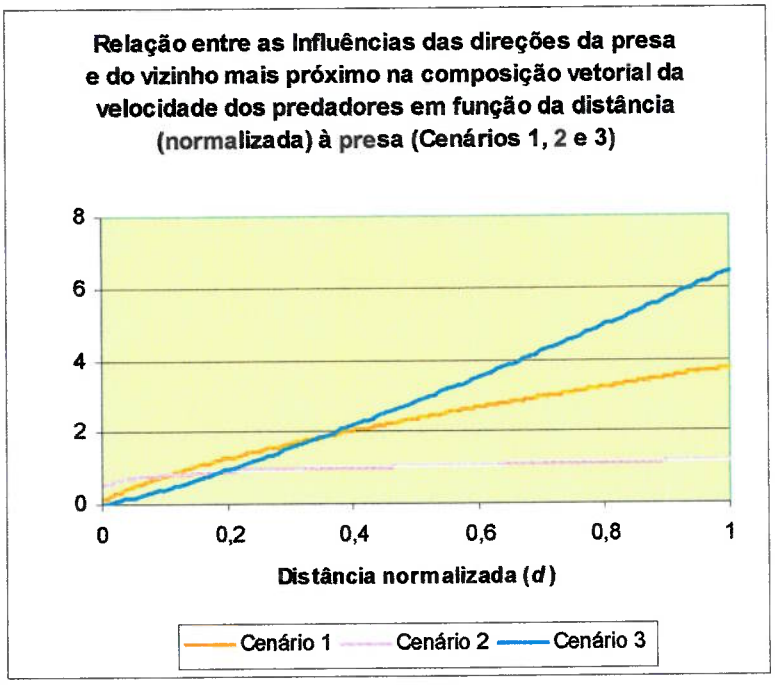

Gráfico 7.10 – Comparação entre as influências de X_2 e X_3 nos 3 cenários.

Além do gráfico de análise de convergência, uma forma de acompanhar visualmente o processo de otimização do PSO é proporcionado pelo modo de animação do Programa PREDADOR. A figura 7.1 apresenta 4 "momentos" congelados de uma animação do cenário básico, com PSO configurado para 20 partículas. Cada quadro contém 3 câmeras que ilustram as 3 vistas (topo, frontal e lateral) do espaço de parâmetros, representando como uma "caixa" acinzentada, dimensionada a partir dos limites do universo de soluções adotado no problema.

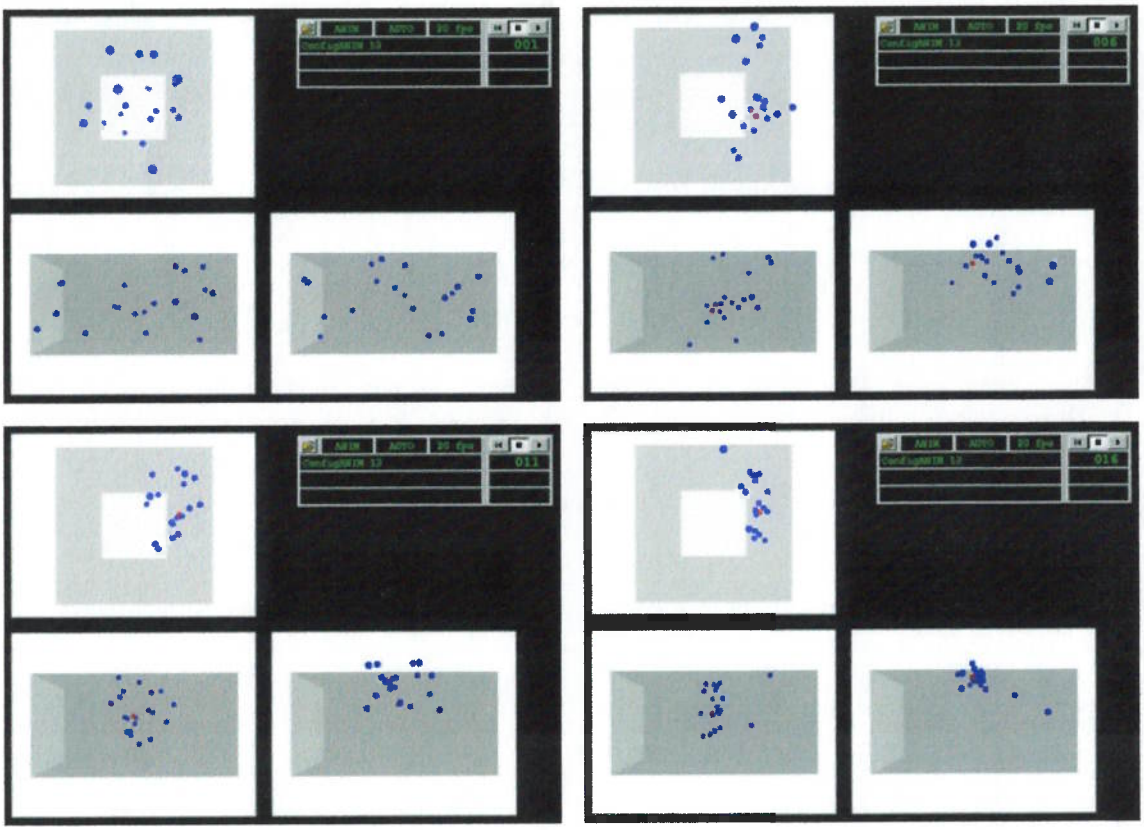

Figura 7.1 – Animação da movimentação das partículas no PSO, em quatro momentos: t=1 (superior esquerda), t=6 (superior direita), t=11 (inferior esquerda) e t=16 (inferior direita).

As bolinhas azuis representam as partículas, e a bolinha vermelha a posição encontrada pelo enxame que obteve o melhor resultado até então. É possível perceber que o enxame rapidamente evolui para uma aglomeração em torno dos pontos mais "interessantes" já encontrados, mas mantendo a típica movimentação aparentemente caótica dos enxames de insetos, a metáfora de origem do algoritmo de otimização. Movimentação que, infelizmente, não é possível de ser observada nestas imagens estáticas.

Esta percepção do comportamento do PSO neste problema contribui, de certa forma, para reduzir um pouco uma insegurança que no fundo remete à uma questão muito importante em otimização. Pode-se indagar se outros algoritmos de busca não seriam mais eficazes, como Algoritmos Genéticos ou Recozimento Simulado. E esta indagação nos leva ao teorema do "*No Free Lunch*"[27] (NFL): nenhum algoritmo pode ser melhor que nenhum outro considerando a média sobre todas as funções objetivas [WOLPERT, MACREADY, 1997].

[27] Equivalente à expressão popular "não existe almoço grátis", que significa que tudo tem o seu preço.

Pelo teorema NFL, para avaliar um algoritmo deveríamos considerar todas as funções objetivas possíveis. O que leva à conclusão de que não faz muito sentido discutir a eficiência de um algoritmo fora de um contexto bem definido, que é problema ao qual está sendo aplicado. Em outras palavras, é fundamental que o algoritmo de otimização tenha "afinidade" com a modelagem e a natureza do problema.

E a visualização proporcionada pelo modo de animação do Programa PREDADOR permite que se obtenha, ainda que de forma indireta e superficial, uma idéia de como é o comportamento geral da função objetiva dentro do espaço de parâmetros, pela simples observação de como o enxame de partículas se comporta ao longo da otimização. Se "desliza" rapidamente para alguma região ou se simplesmente percorre todo o domínio, se produz aglomerados coesos ou se mantém uma "nuvem" difusa. E assim, pode-se obter algum tipo de avaliação sobre a adequação do algoritmo às condições particulares do problema.

Mas vale lembrar que neste trabalho o espaço de parâmetros é tridimensional, e portanto passível de visualização (o programa disponibiliza três câmeras para observar o enxame ao longo dos três eixos ortogonais). Em outras aplicações, envolvendo problemas de maior dimensionalidade do espaço de parâmetros, este tipo de acompanhamento pode não ser tão simples assim. Mas nada impede que outras alternativas possam ser adotadas com o intuito de verificar como o enxame percorre e se distribui ao longo do domínio.

O uso de meta-heurísticas na solução de problemas, em especial em aplicações de elevado impacto econômico e social pode produzir, em certa medida, algum desconforto por parte dos engenheiros e cientistas que utilizam estes algoritmos. O aparente distanciamento entre o processo computacional de solução numérica e uma metodologia analítica de equacionamento pode provocar no usuário uma sensação de que a técnica de otimização, cega, é capaz de oferecer soluções descabidas devido à total falta de sensibilidade e compreensão do problema por parte do algoritmo, o que a princípio poderia fornecer alguma dica sobre o "mapeamento" das soluções ao longo do domínio.

Esta sensação é difícil de ser anulada, mas pode ser atenuada em grande parte se os usuários puderem acompanhar mentalmente o processo de otimização dentro da sua

lógica metafórica. Neste trabalho, o cuidado com a visualização (tanto da simulação quanto da movimentação do PSO) foi capaz de reestabelecer parte desta segurança. Ao visualizar as partículas varrendo boa parte do universo de parâmetros, pode-se adquirir maior confiança na capacidade de exploração do PSO. A aglomeração rápida das partículas aponta para a existência de verdadeiros "vales" dentro da função objetiva, que são encontrados ainda que o PSO não dependa de informações do gradiente desta função.

O fato do PSO ser, no final das contas, uma meta-heurística e, como toda meta-heurística, não garantir a obtenção de um ótimo global não deve ser motivo de depreciação desta técnica. A improdutiva obsessão por ótimos globais foi bem contestada por [KENNEDY, EBERHART, 2001]:

> *"Nós reconhecemos que não vivemos num mundo de uns e zeros, de verdadeiro e falso. A vida real, incluindo engenharia e aplicações de ciência da computação, deve lidar de maneira adequada com incerteza e imprecisão, com variáveis lingüísticas e dados cheios de ruído. Nós devemos aprender a viver, mais e mais a cada dia, com respostas quase-ótimas. Devemos compreender que otimização global é geralmente um mito. Nós não fazemos isso como humanos; nem deveríamos, na maioria dos casos, esperar que nossas máquinas façam isso."*

8. CONCLUSÕES

A capacidade dos sistemas multiagente de resolver problemas a partir de estratégias cooperativas expande o universo de soluções de certos problemas para além de alguns limites humanos de raciocínio e intuição. A dificuldade que temos em processar mentalmente, de forma paralela, as diversas "instâncias" sensoriais-cognitivas-atuadoras que constituem os agentes tem o efeito de tornar invariavelmente surpreendente o comportamento do sistema como um todo.

É possível experimentar esta dificuldade procurando simular, mentalmente, os processos de caça simplesmente a partir dos gráficos de influências dos parâmetros de estratégias presentes no capítulo anterior. O resultado desta experiência é um forte argumento a favor da construção de simuladores.

Por extensão, é possível perceber que esta incapacidade de antecipar o comportamento de certos sistemas multiagente, de antecipar a emergência, é responsável por bloquear parte do real universo de soluções de um problema, que ficaria então limitado às abordagens centralizadoras mais facilmente compreendidas pela sua compatibilidade com o processamento humano.

Neste trabalho, um exemplo de aplicação procurou demonstrar que é possível projetar tais sistemas de Inteligência Artificial Distribuída a partir apenas da capacidade de modelagem do problema. O "ajuste fino" dos detalhes de projeto foi realizado por uma ferramenta de síntese, com um sistema de otimização que também tira proveito de um comportamento emergente (das partículas do PSO). Assim, parte da complexidade do problema foi resolvida sem nenhuma abordagem analítica, mas através de um processo inteligente e automatizado de busca de soluções.

Em relação às estratégias de caça é interessante notar que, depois de acompanhar determinado número de simulações, é possível adquirir alguma intuição sobre as táticas geradas para o time de predadores nos diferentes cenários. Algumas possuem uma formulação simples, com contrastes claros entre os valores dos parâmetros. As simulações comprovam que a aparente elegância dessas estratégias muitas vezes também se traduz em capturas praticamente certas. No entanto, no algoritmo de otimização estas estratégias são geralmente preteridas em favor de outras menos intuitivas, que muitas vezes não garantem a captura.

Ocorre que embora estas estratégias menos intuitivas, privilegiadas pelo otimizador, não *garantam* a captura, elas em geral possuem grande probabilidade de alcançar a presa, e quando o fazem, isso é feito rapidamente. As outras, em contrapartida, fazem com que os predadores partam para caçadas certeiras, mas que podem demorar dependendo do posicionamento inicial (aleatório) dos peixes. Portanto às vezes, para uma mesma unidade de tempo, algumas estratégias menos "elegantes" são mais eficientes. E é isso que a função objetiva reflete, e por conseqüência o PSO. Não seria nada estranho se, na natureza, um bando de predadores também optasse por esse tipo de tática.

Os resultados obtidos neste trabalho podem atestar que a ferramenta de síntese desenvolvida é realmente capaz de fornecer, desde que trabalhando a partir de modelos bem elaborados, soluções satisfatórias para problemas de natureza complexa, de difícil resolução a partir de abordagens analíticas.

Numa apresentação feita em 1992 num Workshop de Inteligência Artificial, Mark Millonas citou vários tópicos científicos que poderiam ser iluminados pelo estudo dos comportamentos emergentes dos enxames, como argumentos para defender a pesquisa nesta área, mas admitiu [MILLONAS, 1993]:

> *"No final das contas, talvez o mais insinuante apelo dos enxames reside numa espécie de atratividade emocional ao tema... Mais que um paradigma, enxames são, freqüentemente, quase como um arquétipo"*

Neste trabalho, o estudo da inteligência dos enxames também foi motivado em parte pela suspeita de que há sabedoria a ser obtida a partir dele, e pela sensação de que há algo a respeito das desordenadas interações de elementos muito simples e suas façanhas que é simplesmente fascinante. Parece haver algo profundo e significativo nestes fenômenos, algo que talvez transcenda a nossa tradição intelectual.

Entregar a tarefa de resolver um problema para uma ferramenta nos moldes propostos neste trabalho significa, em última análise, confiar nas propriedades emergentes de um sistema complexo para produzir soluções para outro sistema (não por coincidência, também complexo). Estas soluções não estão pré-definidas no programa. E nem somos capazes de compreender facilmente as soluções geradas em termos de código do programa. Isso tende a ir na direção contrária da essência da engenharia tradicional. Mas que eventualmente descortina um campo ainda muito pouco explorado na área de desenvolvimento de projetos.

Referências Bibliográficas

[Adami, 1998] ADAMI, C. **Introduction to Artificial Life**. Ed. Springer-Verlag. Nova Iorque, 1998.

[Andradóttir, 1998] ANDRADÓTTIR, S. **A Review of Simulation Optimization Techniques**. Proceedings of the 1998 Winter Simulation Conference, ed. D. J. Medeiros, E. F. Watson, J. S. Carson e M. S. Manivannan, pp. 151-158, 1998.

[Benda, *et al*, 1985] BENDA, M; JAGANNATHAN, V; DODHIAWALA, R. **On Optimal Cooperation of Knowledge Sources**. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue. Washington, 1985.

[Bonabeau, *et al*, 1999] BONABEAU, E; DORIGO, M; THERAULAZ. G. **Swarm Intelligence: From Natural to Artificial Systems**. Ed. Oxford University Press. Nova Iorque, 1999.

[Bond, Gasser, 1988] BOND, A. H; GASSER, L. **An analysis of problems and research in DAI**. Readings in Distributed Artificial Intelligence, ed. A. H. Bond e L. Gasser, pp. 3-35. Ed. Morgan Kaufmann. San Mateo, California, 1988 *apud* [Weiss, 2000].

[Casti, 1997] CASTI, J. L. **Would-be Worlds**. Ed. Wiley. Nova Iorque, 1997 *apud* [Crichton, 2002].

[Coelho, Krohling, 2005] COELHO, L. S; KROHLING, R. **Learning of B-Spline Neural Networks using new Particle Swarm approaches**. Proceedings of the 18th International Congress of Mechanical Engineering, pp. 1-6. Ouro Preto, MG, 2005.

[Crichton, 2002] CRICHTON, M. **Prey**. 1ª edição. Ed. Harper Collins. Nova Iorque, 2002.

[Eberhart, Shi, 1998] EBERHART, R. C; SHI, Y. H. **Evolving Artificial Neural Networks**. Proceedings International Conference on Neural Networks and Brain, pp. 5-13. Publishing House of Electronics Industry. Pequim, 1998 *apud* [Magoulas, *et al*, 2002].

[Grandi, 2003] GRANDI, F. K. **Otimização por Inteligência de Enxames: Algoritmos Ant Colony e Particle Swarm**. Dissertação (Mestrado). Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

[Grassé, 1959] GRASSÉ, P. P. **La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes et cubitermes sp**. La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insect Societies, vol. 6, pp. 41-80 *apud* [Kennedy, Eberhart, 2001].

[Heppner, Grenander, 1990] HEPPNER, F; GRENANDER, U. **A stochastic nonlinear model dor coordinated bird flocks**. The Ubiquity of Chaos, ed. S. Krasner, Ed. AAAS Publications, Washington, 1990 *apud* [Kennedy, Eberhart, 2001].

[Jennings, *et al*, 1998] JENNINGS, N. R; SYCARA, K; WOOLDRIDGE, M. **A roadmap of agent research and development**. Autonomous Agents and Multi-Agent Systems, vol. 1, pp. 7-38, 1998 *apud* [Weiss, 2000].

[Kennedy, Eberhart, 2001] KENNEDY, J; EBERHART, R. C. **Swarm Intelligence**. Ed. Morgan Kaufmann. São Francisco, 2001.

[Magoulas, *et al*, 2002] MAGOULAS, G. D; ELDABI, T; PAUL, R. J. **Global Search Strategies for Simulation Optimization**. Proceedings of the 2002 Winter Simulation Conference, ed. E. Yücesan, C. H. Chen, J. L. Snowdon e J. M. Charnes, pp. 1978-1985, 2002.

[Manela, Campbell, 1995] MANELA, M; CAMPBELL, J. A. **Designing Good Pursuit Problems as Testbeds for Distributed AI: a Novel Application of Genetic Algorithms**. Lecture Notes on Artificial Intelligence 957. Ed. Springer. Berlin, 1995.

[Matejcik, Nelson, 1995] MATEJCIK, F. J; NELSON, B. L. **Two-Stage Multiple Comparisons with the Best for Computer Simulation**. Operations Research, vol. 43, pp. 633-640, 1995 *apud* [Ólafsson, Kim, 2002].

[Millonas, 1993] MILLONAS, M. M. **Swarms, phase transitions, and collective intelligence**. Artificial Life III, ed. C. Langton, pp. 417-445. Ed. Addison-Wesley. Reading, MA, 1999.

[Miranda, *et al*, 2004] MIRANDA, F. R; KÖGLER, J. E; HERNANDEZ, E. M; NETTO, M. L. **An artificial life approach for the animation of cognitive characters**. Boletim Técnico PSI/0402. Escola Politécnica, Universidade de São Paulo. São Paulo, 2004.

[Neves, Netto, 2003] NEVES, R. P. O; NETTO, M. L. **A.L.I.V.E. Vida Artificial em Ambientes Virtuais: Uma plataforma experimental em realidade virtual para estudos dos seres vivos e da dinâmica da vida**. Boletim Técnico PSI/0308. Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

[Ólafsson, Kim, 2002] ÓLAFSSON, S; KIM, J. **Simulation Optimization**. Proceedings of the 2002 Winter Simulation Conference, ed. E. Yücesan, C. H. Chen, J. L. Snowdon e J. M. Charnes, pp. 79-84, 2002.

[Parunak, 1997] PARUNAK, H. V. D. **Go to the Ant: Engineering Principles from Natural Agent Systems**. Annals of Operations Research, vol. 75, pp. 69-101, 1997.

[Reynolds, 1999] REYNOLDS, C. W. **Steering Behaviors for Autonomous Characters**. Sony Computer Entertainment America, 1999.

[Robbins, Monro, 1951] ROBBINS, H; MONRO, S. **A Stochastic Approximation Method**. Annals of Mathematical Statistics, vol. 22, pp. 400-407, 1951 *apud* [Ólafsson, Kim, 2002].

[Weiss, 2000] WEISS, G. **Multiagent Systems: a modern approach to Distributed Artificial Intelligence**. Ed. MIT Press. Massachusetts, 2000.

[Wolpert, Macready, 1997] WOLPERT, D. H; MACREADY, W. G. **No Free Lunch theorems for optimization**. IEEE Transactions on Evolutionary Computation, vol. 1, pp. 67-82, 1997 *apud* [Kennedy, Eberhart, 2001].

Anexo

O texto abaixo apresenta o conteúdo das 50 primeiras iterações de um cenário de configuração de otimização. É um exemplo do formato utilizado pelo programa PREDADOR e dos seus métodos de registro.

```
INTELIGÊNCIA DE ENXAMES APLICADA NA SÍNTESE DE ESTRATÉGIAS DE CAÇA EM AMBIENTE TRIDIMENSIONAL
Emiliano de Castro, 2005

ARQUIVO DE CONFIGURAÇÃO DE OTIMIZAÇÃO

Configuração da Simulação
-------------------------

Código do Cenário de Simulação: Básica 03
Número de Predadores: 4
Duração Máxima da Simulação (s): 45
Velocidade de Passeio da Presa (m/s): 0.2
Velocidade de Fuga da Presa (m/s): 1.0
Distância Inicial entre a Presa e os Predadores (m): 10
Raio da "Bolha de Percepção" da Presa (m): 5
Raio da "Bolha de Proximidade" da Presa (m): 3
Raio da "Bolha Vital" da Presa (m): 0.3
Velocidade de Perseguição dos Predadores (m/s): 1.0
Velocidade Crítica de Aproximação dos Predadores (m/s): 0.6
Coeficiente de Inércia do Simulador: 0.95


Configuração do Algoritmo de Otimização (PSO)
---------------------------------------------

Máximo de Iterações: 100
Número de Partículas: 10
Número de Simulações por Partícula: 5
Tamanho da Vizinhança: 5
Momento de Inércia do PSO (ÔMEGA): 1.0
PHI1max: 0.4
PHI2max: 0.2
Vmax: 2.0


Resultados das Simulações
-------------------------

--- Iteração: 1
[01-01] : ( 1.5064, 2.5164, -0.8659 )    -> 79.8238
[01-02] : ( 2.4095, 8.6992, -2.9093 )    -> 211.2542
[01-03] : ( 6.1786, 8.3009, -4.6586 )    -> 219.7778
[01-04] : ( 3.0194, 3.2519, 6.0038 )     -> 60.8875
[01-05] : ( 0.6593, 8.9675, -9.1897 )    -> 234.8842
[01-06] : ( 9.8955, 9.6918, 2.5354 )     -> 76.2881
[01-07] : ( 3.9438, 9.4876, 5.2884 )     -> 66.1726
[01-08] : ( 5.8289, 7.1922, 2.1778 )     -> 70.5738
```

```
[01-09] : ( 6.5351, 8.9979, -8.0960 )   -> 227.1310
[01-10] : ( 8.7363, 9.5564, -3.6559 )   -> 215.1179
--- Velocidades: [vector( 0.1936, -1.8217, -0.1332 ), vector( 0.9470, -1.9681, -0.0907
), vector( -1.5898, 0.7042, 1.4882 ), vector( -1.8275, -1.1898, 1.8516 ), vector( -
1.6017, 1.4173, 0.3660 ), vector( -0.3664, -1.1090, -0.9019 ), vector( 0.2363, 1.1604, -
1.5292 ), vector( -1.7759, -1.6318, -1.1878 ), vector( -0.7872, -1.4177, 0.0246 ),
vector( 1.2723, 1.7916, 0.6982 )]
--- Melhores resultados individuais: [79.8238, 211.2542, 219.7778, 60.8875, 234.8842,
76.2881, 66.1726, 70.5738, 227.1310, 215.1179]
--- Posições dos melhores resultados individuais: [vector( 1.5064, 2.5164, -0.8659 ),
vector( 2.4095, 8.6992, -2.9093 ), vector( 6.1786, 8.3009, -4.6586 ), vector( 3.0194,
3.2519, 6.0038 ), vector( 0.6593, 8.9675, -9.1897 ), vector( 9.8955, 9.6918, 2.5354 ),
vector( 3.9438, 9.4876, 5.2884 ), vector( 5.8289, 7.1922, 2.1778 ), vector( 6.5351,
8.9979, -8.0960 ), vector( 8.7363, 9.5564, -3.6559 )]
--- Código da Melhor Estratégia: 01-04
--- Melhor Estratégia: vector( 3.0194, 3.2519, 6.0038 )
--- Resultado da Melhor Estratégia: 60.8875


--- Iteração: 2
[02-01] : ( 1.7000, 0.6947, -0.9991 )   -> 42.7104
[02-02] : ( 3.4661, 6.6992, -1.3983 )   -> 158.8368
[02-03] : ( 4.5635, 8.9647, -3.0851 )   -> 202.7278
[02-04] : ( 1.1919, 2.0621, 7.8554 )    -> 57.6612
[02-05] : ( -0.7045, 9.8087, -7.2922 )  -> 226.4605
[02-06] : ( 9.4088, 8.4701, 1.6942 )    -> 74.0447
[02-07] : ( 4.1801, 10.6480, 3.7592 )   -> 66.7294
[02-08] : ( 4.0193, 5.6015, 1.0457 )    -> 66.1983
[02-09] : ( 5.7106, 7.5873, -7.8787 )   -> 231.3570
[02-10] : ( 9.8978, 11.2579, -2.7354 )  -> 207.5356
--- Velocidades: [vector( 0.1936, -1.8217, -0.1332 ), vector( 1.0566, -2.0000, 1.5110 ),
vector( -1.6151, 0.6638, 1.5735 ), vector( -1.8275, -1.1898, 1.8516 ), vector( -1.3638,
0.8412, 1.8975 ), vector( -0.4867, -1.2217, -0.8412 ), vector( 0.2363, 1.1604, -1.5292
), vector( -1.8096, -1.5907, -1.1321 ), vector( -0.8245, -1.4106, 0.2173 ), vector(
1.1615, 1.7015, 0.9205 )]
--- Melhores resultados individuais: [42.7104, 158.8368, 202.7278, 57.6612, 226.4605,
74.0447, 66.1726, 66.1983, 227.1310, 207.5356]
--- Posições dos melhores resultados individuais: [vector( 1.7000, 0.6947, -0.9991 ),
vector( 3.4661, 6.6992, -1.3983 ), vector( 4.5635, 8.9647, -3.0851 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 9.8087, -7.2922 ), vector( 9.4088, 8.4701, 1.6942 ),
vector( 3.9438, 9.4876, 5.2884 ), vector( 4.0193, 5.6015, 1.0457 ), vector( 6.5351,
8.9979, -8.0960 ), vector( 9.8978, 10.0000, -2.7354 )]
--- Código da Melhor Estratégia: 02-01
--- Melhor Estratégia: vector( 1.7000, 0.6947, -0.9991 )
--- Resultado da Melhor Estratégia: 42.7104


--- Iteração: 3
[03-01] : ( 1.8936, -1.1270, -1.1323 )  -> 44.2726
[03-02] : ( 4.2156, 4.6992, 0.1821 )    -> 67.9228
[03-03] : ( 2.6197, 8.6791, -1.2721 )   -> 127.9074
[03-04] : ( -0.6356, 0.8723, 9.7070 )   -> 65.7041
[03-05] : ( -1.1418, 9.2066, -5.2922 )  -> 223.9304
[03-06] : ( 7.4088, 6.4701, 2.0095 )    -> 70.2940
[03-07] : ( 4.2973, 11.2237, 3.0006 )   -> 69.5552
[03-08] : ( 2.1962, 4.7013, 0.6675 )    -> 56.8506
[03-09] : ( 4.9994, 6.3695, -7.5107 )   -> 235.0982
[03-10] : ( 9.7223, 10.1838, -1.5318 )  -> 171.4556
--- Velocidades: [vector( 0.1936, -1.8217, -0.1332 ), vector( 0.7495, -2.0000, 1.5804 ),
vector( -1.9438, -0.2856, 1.8130 ), vector( -1.8275, -1.1898, 1.8516 ), vector( -1.1418,
-0.6020, 2.0000 ), vector( -2.0000, -2.0000, 0.3153 ), vector( 0.1172, 0.5757, -0.7586
), vector( -1.8231, -0.9002, -0.3782 ), vector( -0.7112, -1.2178, 0.3680 ), vector( -
0.1755, 0.1838, 1.2037 )]
--- Melhores resultados individuais: [42.7104, 67.9228, 127.9074, 57.6612, 223.9304,
70.2940, 66.1726, 56.8506, 227.1310, 171.4556]
--- Posições dos melhores resultados individuais: [vector( 1.7000, 0.6947, -0.9991 ),
vector( 4.2156, 4.6992, 0.1821 ), vector( 2.6197, 8.6791, -1.2721 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 9.2066, -5.2922 ), vector( 7.4088, 6.4701, 2.0095 ),
vector( 3.9438, 9.4876, 5.2884 ), vector( 2.1962, 4.7013, 0.6675 ), vector( 6.5351,
8.9979, -8.0960 ), vector( 9.7223, 10.0000, -1.5318 )]
--- Código da Melhor Estratégia: 02-01
--- Melhor Estratégia: vector( 1.7000, 0.6947, -0.9991 )
--- Resultado da Melhor Estratégia: 42.7104


--- Iteração: 4
[04-01] : ( 2.0407, -2.5113, -1.2335 )  -> 64.1716
```

```
[04-02] : ( 4.7764, 2.6992, 1.6739 )    -> 68.7903
[04-03] : ( 0.6197, 7.2047, 0.5815 )    -> 56.4691
[04-04] : ( -1.7261, 0.1623, 10.8118 )  -> 60.8931
[04-05] : ( -1.0315, 7.9438, -3.2922 )  -> 213.0270
[04-06] : ( 5.4088, 4.4701, 2.2281 )    -> 67.8924
[04-07] : ( 4.0552, 10.6495, 1.9868 )   -> 66.6657
[04-08] : ( 0.3732, 3.8012, 0.2893 )    -> 55.4140
[04-09] : ( 4.7102, 5.8732, -7.1512 )   -> 228.0037
[04-10] : ( 8.4004, 8.8541, -0.2520 )   -> 80.2857
--- Velocidades: [vector( 0.1471, -1.3843, -0.1012 ), vector( 0.5608, -2.0000, 1.4918 ),
vector( -2.0000, -1.4745, 1.8536 ), vector( -1.0905, -0.7100, 1.1048 ), vector( -1.0315,
-1.2629, 2.0000 ), vector( -2.0000, -2.0000, 0.2186 ), vector( -0.2421, -0.5742, -1.0137
), vector( -1.8231, -0.9002, -0.3782 ), vector( -0.2892, -0.4963, 0.3595 ), vector( -
1.3219, -1.1459, 1.2798 )]
--- Melhores resultados individuais: [42.7104, 67.9228, 56.4691, 57.6612, 213.0270,
67.8924, 66.1726, 55.4140, 227.1310, 80.2857]
--- Posições dos melhores resultados individuais: [vector( 1.7000, 0.6947, -0.9991 ),
vector( 4.2156, 4.6992, 0.1821 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 7.9438, -3.2922 ), vector( 5.4088, 4.4701, 2.2281 ),
vector( 3.9438, 9.4876, 5.2884 ), vector( 0.3732, 3.8012, 0.2893 ), vector( 6.5351,
8.9979, -8.0960 ), vector( 8.4004, 8.8541, -0.2520 )]
--- Código da Melhor Estratégia: 02-01
--- Melhor Estratégia: vector( 1.7000, 0.6947, -0.9991 )
--- Resultado da Melhor Estratégia: 42.7104


--- Iteração: 5
[05-01] : ( 2.0636, -2.7264, -1.2492 )  -> 61.1941
[05-02] : ( 4.7445, 1.1378, 2.3004 )    -> 67.3504
[05-03] : ( -1.2861, 5.2047, 2.2973 )   -> 58.0912
[05-04] : ( -1.9243, 0.9907, 9.6493 )   -> 63.0474
[05-05] : ( -0.9531, 6.5874, -1.2922 )  -> 137.9747
[05-06] : ( 3.4088, 2.4701, 2.0718 )    -> 62.2492
[05-07] : ( 3.3416, 8.8907, 1.8364 )    -> 62.8121
[05-08] : ( -1.4499, 2.9010, -0.0889 )  -> 54.2726
[05-09] : ( 4.5485, 5.5921, -6.2697 )   -> 223.2077
[05-10] : ( 6.4004, 6.8541, 0.8954 )    -> 70.1951
--- Velocidades: [vector( 0.0229, -0.2151, -0.0157 ), vector( -0.0319, -1.5614, 0.6265
), vector( -1.9058, -2.0000, 1.7158 ), vector( -0.1983, 0.8283, -1.1625 ), vector( -
0.9531, -1.3564, 2.0000 ), vector( -2.0000, -2.0000, -0.1563 ), vector( -0.7136, -
1.7588, -0.1505 ), vector( -1.8231, -0.9002, -0.3782 ), vector( -0.1617, -0.2811, 0.8815
), vector( -2.0000, -2.0000, 1.1474 )]
--- Melhores resultados individuais: [42.7104, 67.3504, 56.4691, 57.6612, 137.9747,
62.2492, 62.8121, 54.2726, 223.2077, 70.1951]
--- Posições dos melhores resultados individuais: [vector( 1.7000, 0.6947, -0.9991 ),
vector( 4.7445, 1.1378, 2.3004 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 6.5874, -1.2922 ), vector( 3.4088, 2.4701, 2.0718 ),
vector( 3.3416, 8.8907, 1.8364 ), vector( 0.0000, 2.9010, -0.0889 ), vector( 4.5485,
5.5921, -6.2697 ), vector( 6.4004, 6.8541, 0.8954 )]
--- Código da Melhor Estratégia: 02-01
--- Melhor Estratégia: vector( 1.7000, 0.6947, -0.9991 )
--- Resultado da Melhor Estratégia: 42.7104


--- Iteração: 6
[06-01] : ( 1.9161, -1.3390, -1.1478 )  -> 53.2938
[06-02] : ( 4.2117, -0.4965, 2.3842 )   -> 65.3781
[06-03] : ( -1.9629, 3.2867, 2.8106 )   -> 66.5778
[06-04] : ( -0.5116, 2.9907, 7.6493 )   -> 65.8488
[06-05] : ( -0.9348, 5.2492, 0.7078 )   -> 60.6314
[06-06] : ( 1.4088, 0.5099, 1.7158 )    -> 55.6960
[06-07] : ( 1.9747, 6.8907, 1.3095 )    -> 58.1686
[06-08] : ( -1.8231, 2.0009, -0.4671 )  -> 19.5625
[06-09] : ( 4.0934, 4.8066, -4.8454 )   -> 226.1781
[06-10] : ( 4.4004, 4.8541, 1.9403 )    -> 68.1625
--- Velocidades: [vector( -0.1474, 1.3874, 0.1014 ), vector( -0.5327, -1.6343, 0.0838 ),
vector( -0.6768, -1.9180, 0.5133 ), vector( 1.4127, 2.0000, -2.0000 ), vector( -0.9348,
-1.3382, 2.0000 ), vector( -2.0000, -1.9602, -0.3560 ), vector( -1.3669, -2.0000, -
0.5268 ), vector( -1.8231, -0.9002, -0.3782 ), vector( -0.4551, -0.7855, 1.4244 ),
vector( -2.0000, -2.0000, 1.0449 )]
--- Melhores resultados individuais: [42.7104, 65.3781, 56.4691, 57.6612, 60.6314,
55.6960, 58.1686, 19.5625, 223.2077, 68.1625]
--- Posições dos melhores resultados individuais: [vector( 1.7000, 0.6947, -0.9991 ),
vector( 4.2117, 0.0000, 2.3842 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 5.2492, 0.7078 ), vector( 1.4088, 0.5099, 1.7158 ),
vector( 1.9747, 6.8907, 1.3095 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 4.5485,
5.5921, -6.2697 ), vector( 4.4004, 4.8541, 1.9403 )]
```

```
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 7
[07-01] : ( 1.6848, 0.6610, -0.9887 )   -> 37.8314
[07-02] : ( 3.2191, -1.5071, 1.8485 )   -> 58.5403
[07-03] : ( -1.6531, 2.5550, 2.4476 )   -> 61.5453
[07-04] : ( 1.4884, 4.3222, 5.6493 )    -> 61.7232
[07-05] : ( -0.8197, 3.5238, 2.7078 )   -> 61.0350
[07-06] : ( -0.5912, -1.4037, 1.2917 )  -> 53.8068
[07-07] : ( 0.4528, 4.8907, 0.6432 )    -> 61.8722
[07-08] : ( -1.8231, 1.1007, -0.8453 )  -> 36.1162
[07-09] : ( 3.1762, 3.7692, -3.0552 )   -> 214.0535
[07-10] : ( 2.4004, 2.8541, 2.6655 )    -> 56.9147
--- Velocidades: [vector( -0.2313, 2.0000, 0.1591 ), vector( -0.9926, -1.5071, -0.5357
), vector( 0.3098, -0.7316, -0.3630 ), vector( 2.0000, 1.3315, -2.0000 ), vector( -
0.8197, -1.7254, 2.0000 ), vector( -2.0000, -1.9137, -0.4241 ), vector( -1.5219, -
2.0000, -0.6663 ), vector( -1.8231, -0.9002, -0.3782 ), vector( -0.9171, -1.0374, 1.7902
), vector( -2.0000, -2.0000, 0.7252 )]
--- Melhores resultados individuais: [37.8314, 58.5403, 56.4691, 57.6612, 60.6314,
53.8068, 58.1686, 19.5625, 214.0535, 56.9147]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 3.2191, 0.0000, 1.8485 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 5.2492, 0.7078 ), vector( 0.0000, 0.0000, 1.2917 ),
vector( 1.9747, 6.8907, 1.3095 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 3.1762,
3.7692, -3.0552 ), vector( 2.4004, 2.8541, 2.6655 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 8
[08-01] : ( 1.4535, 2.6610, -0.8295 )   -> 70.7621
[08-02] : ( 2.1024, -1.4536, 1.0832 )   -> 53.9267
[08-03] : ( -0.9869, 2.3963, 1.7795 )   -> 60.3816
[08-04] : ( 3.2854, 4.9356, 3.6493 )    -> 62.1268
[08-05] : ( -1.4444, 1.5238, 4.3214 )   -> 63.7869
[08-06] : ( -2.0000, -1.5259, 0.5268 )  -> 59.1648
[08-07] : ( -0.7812, 2.8907, -0.0705 )  -> 46.4080
[08-08] : ( -3.0815, 0.4794, -1.1063 )  -> 46.0991
[08-09] : ( 1.8112, 2.4824, -1.0552 )   -> 105.3401
[08-10] : ( 0.4004, 0.8541, 2.7986 )    -> 62.4756
--- Velocidades: [vector( -0.2313, 2.0000, 0.1591 ), vector( -1.1167, -1.4536, -0.7652
), vector( 0.6662, -0.1588, -0.6680 ), vector( 1.7970, 0.6134, -2.0000 ), vector( -
0.6246, -2.0000, 1.6136 ), vector( -2.0000, -1.5259, -0.7649 ), vector( -1.2339, -
2.0000, -0.7137 ), vector( -1.2585, -0.6214, -0.2611 ), vector( -1.3650, -1.2867, 2.0000
), vector( -2.0000, -2.0000, 0.1331 )]
--- Melhores resultados individuais: [37.8314, 53.9267, 56.4691, 57.6612, 60.6314,
53.8068, 46.4080, 19.5625, 105.3401, 56.9147]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 2.1024, 0.0000, 1.0832 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 5.2492, 0.7078 ), vector( 0.0000, 0.0000, 1.2917 ),
vector( 0.0000, 2.8907, -0.0705 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 1.8112,
2.4824, -1.0552 ), vector( 2.4004, 2.8541, 2.6655 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 9
[09-01] : ( 1.2854, 4.1146, -0.7139 )   -> 82.4897
[09-02] : ( 0.9083, -1.3311, -0.0659 )  -> 42.7844
[09-03] : ( -0.1348, 2.7745, 0.9713 )   -> 59.3763
[09-04] : ( 4.9672, 5.3764, 1.6493 )    -> 68.8616
[09-05] : ( -1.5970, 0.5184, 4.6805 )   -> 62.7558
[09-06] : ( -3.2408, -2.2819, -0.1154 ) -> 48.6079
[09-07] : ( -1.2339, 0.8907, -0.8106 )  -> 28.6424
[09-08] : ( -3.4321, 0.3062, -1.1791 )  -> 55.1283
[09-09] : ( 0.4408, 1.1943, 0.9448 )    -> 54.5017
[09-10] : ( -1.2587, -0.6371, 2.5519 )  -> 63.5823
--- Velocidades: [vector( -0.1681, 1.4536, 0.1157 ), vector( -1.1941, -1.3311, -1.1492
), vector( 0.8521, 0.3782, -0.8082 ), vector( 1.6818, 0.4408, -2.0000 ), vector( -
0.1526, -1.0054, 0.3591 ), vector( -1.2408, -0.7560, -0.6422 ), vector( -1.2339, -
2.0000, -0.7401 ), vector( -0.3506, -0.1731, -0.0727 ), vector( -1.3704, -1.2882, 2.0000
), vector( -1.6590, -1.4912, -0.2468 )]
```

```
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 57.6612, 60.6314,
48.6079, 28.6424, 19.5625, 54.5017, 56.9147]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 5.2492, 0.7078 ), vector( 0.0000, 0.0000, -0.1154 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 2.4004, 2.8541, 2.6655 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 10
[10-01] : ( 1.2093, 4.7728, -0.6615 )   -> 82.5914
[10-02] : ( -0.1367, -1.2042, -1.3924 ) -> 72.5846
[10-03] : ( 1.0879, 4.3715, -0.1047 )   -> 58.3243
[10-04] : ( 5.3152, 4.5086, 1.2053 )    -> 68.2137
[10-05] : ( -1.0163, 1.1567, 3.0312 )   -> 59.5886
[10-06] : ( -1.2408, -0.6125, -0.7828 ) -> 35.4443
[10-07] : ( -1.2339, -1.0790, -1.5414 ) -> 64.4614
[10-08] : ( -2.1789, 0.9250, -0.9191 )  -> 38.4741
[10-09] : ( -1.0151, 0.0624, 2.6712 )   -> 58.7163
[10-10] : ( -1.3296, -0.3835, 1.8660 )  -> 54.3772
--- Velocidades: [vector( -0.0761, 0.6582, 0.0524 ), vector( -1.0449, -1.2042, -1.3264
), vector( 1.2227, 1.5970, -1.0760 ), vector( 0.3480, -0.8678, -0.4440 ), vector(
0.5807, 0.6383, -1.6493 ), vector( -1.2408, -0.6125, -0.6674 ), vector( -1.2339, -
1.9697, -0.7307 ), vector( 1.2532, 0.6188, 0.2600 ), vector( -1.4559, -1.1318, 1.7264 ),
vector( -0.0709, 0.2536, -0.6858 )]
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 57.6612, 59.5886,
35.4443, 28.6424, 19.5625, 54.5017, 54.3772]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 1.1567, 3.0312 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 0.0000, 0.0000, 1.8660 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 11
[11-01] : ( 1.2689, 4.2571, -0.7025 )   -> 82.1633
[11-02] : ( -0.5987, -1.7831, -2.3585 ) -> 130.6396
[11-03] : ( 2.1551, 6.3715, -0.9528 )   -> 87.6812
[11-04] : ( 4.9401, 3.1313, 1.3003 )    -> 66.8258
[11-05] : ( 0.5807, 1.7430, 1.0312 )    -> 61.6497
[11-06] : ( -1.2408, -0.2966, -1.4003 ) -> 62.8535
[11-07] : ( -2.2122, -2.4383, -2.0581 ) -> 127.3465
[11-08] : ( -0.1789, 1.9178, -0.5020 )  -> 28.0062
[11-09] : ( -2.2711, -0.7350, 3.8593 )  -> 58.1250
[11-10] : ( -0.0709, 0.3477, 1.0706 )   -> 56.7634
--- Velocidades: [vector( 0.0596, -0.5157, -0.0410 ), vector( -0.4621, -0.5790, -0.9661
), vector( 1.0672, 2.0000, -0.8481 ), vector( -0.3751, -1.3773, 0.0949 ), vector(
0.5807, 0.5863, -2.0000 ), vector( -1.2408, -0.2966, -0.6175 ), vector( -0.9783, -
1.3593, -0.5167 ), vector( 2.0000, 0.9928, 0.4171 ), vector( -1.2561, -0.7975, 1.1880 ),
vector( -0.0709, 0.3477, -0.7955 )]
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 57.6612, 59.5886,
35.4443, 28.6424, 19.5625, 54.5017, 54.3772]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.1919,
2.0621, 7.8554 ), vector( 0.0000, 1.1567, 3.0312 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 0.0000, 0.0000, 1.8660 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 12
[12-01] : ( 1.3864, 3.2416, -0.7833 )   -> 81.3523
[12-02] : ( -0.7301, -1.9905, -2.9821 ) -> 212.3989
[12-03] : ( 2.7139, 8.3715, -1.2966 )   -> 138.8434
[12-04] : ( 3.2639, 1.1313, 1.6333 )    -> 55.7439
[12-05] : ( 0.8471, 1.9633, -0.5891 )   -> 56.0172
[12-06] : ( -2.3171, -0.3950, -1.9109 ) -> 145.1521
[12-07] : ( -2.6699, -2.8697, -2.2366 ) -> 196.3204
[12-08] : ( 1.8211, 2.9216, -0.0803 )   -> 50.1812
```

```
[12-09] : ( -2.6833, -0.8280, 3.9652 )  -> 59.5852
[12-10] : ( -0.1195, 0.8298, 0.2407 )   -> 53.8609
--- Velocidades: [vector( 0.1174, -1.0156, -0.0808 ), vector( -0.1314, -0.2074, -0.6236
), vector( 0.5588, 2.0000, -0.3438 ), vector( -1.6762, -2.0000, 0.3330 ), vector(
0.2664, 0.2203, -1.6203 ), vector( -1.0763, -0.0984, -0.5106 ), vector( -0.4577, -
0.4314, -0.1785 ), vector( 2.0000, 1.0038, 0.4217 ), vector( -0.4121, -0.0930, 0.1059 ),
vector( -0.0487, 0.4821, -0.8299 )]
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 55.7439, 56.0172,
35.4443, 28.6424, 19.5625, 54.5017, 53.8609]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 3.2639,
1.1313, 1.6333 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 0.0000, 0.8298, 0.2407 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 13
[13-01] : ( 1.5798, 1.5690, -0.9164 )   -> 67.2216
[13-02] : ( -0.7995, -2.1239, -3.5064 ) -> 205.3897
[13-03] : ( 2.6595, 9.2748, -1.1555 )   -> 115.8031
[13-04] : ( 1.5163, -0.8687, 1.9134 )   -> 54.9913
[13-05] : ( 1.0285, 2.0760, -2.2316 )   -> 177.1956
[13-06] : ( -2.7650, 0.0001, -2.0546 )  -> 127.0005
[13-07] : ( -2.3182, -1.9647, -1.9220 ) -> 164.1381
[13-08] : ( 3.1387, 3.5804, 0.1965 )    -> 60.2094
[13-09] : ( -2.4322, -0.4420, 3.3463 )  -> 58.0129
[13-10] : ( -0.0487, 1.4916, -0.6978 )  -> 35.6651
--- Velocidades: [vector( 0.1934, -1.6726, -0.1331 ), vector( -0.0693, -0.1334, -0.5243
), vector( -0.0544, 0.9032, 0.1411 ), vector( -1.7476, -2.0000, 0.2801 ), vector(
0.1814, 0.1127, -1.6425 ), vector( -0.4479, 0.3951, -0.1437 ), vector( 0.3518, 0.9050,
0.3146 ), vector( 1.3177, 0.6588, 0.2768 ), vector( 0.2511, 0.3860, -0.6189 ), vector( -
0.0487, 0.6619, -0.9385 )]
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 54.9913, 56.0172,
35.4443, 28.6424, 19.5625, 54.5017, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 14
[14-01] : ( 1.7409, -0.1686, -1.0490 )  -> 39.1954
[14-02] : ( -0.1067, -0.8022, -2.2634 ) -> 148.8289
[14-03] : ( 2.1346, 8.2732, -0.7380 )   -> 86.6373
[14-04] : ( -0.3454, -2.0000, 1.9908 )  -> 55.6881
[14-05] : ( 1.0565, 2.0414, -3.2352 )   -> 211.5661
[14-06] : ( -2.6562, 0.5539, -1.9172 )  -> 162.2854
[14-07] : ( -1.0922, 0.0353, -1.1720 )  -> 54.1335
[14-08] : ( 3.6776, 3.8473, 0.3087 )    -> 64.0487
[14-09] : ( -0.9992, 0.6668, 1.6558 )   -> 61.1548
[14-10] : ( -0.0487, 2.1758, -1.6262 )  -> 176.8091
--- Velocidades: [vector( 0.1611, -1.7376, -0.1326 ), vector( 0.6928, 1.3217, 1.2430 ),
vector( -0.5249, -1.0016, 0.4176 ), vector( -1.8617, -2.0000, 0.0774 ), vector( 0.0279,
-0.0346, -1.0036 ), vector( 0.1087, 0.5537, 0.1374 ), vector( 1.2260, 2.0000, 0.7499 ),
vector( 0.5389, 0.2669, 0.1122 ), vector( 1.4330, 1.1089, -1.6905 ), vector( -0.0487,
0.6842, -0.9284 )]
--- Melhores resultados individuais: [37.8314, 42.7844, 56.4691, 54.9913, 56.0172,
35.4443, 28.6424, 19.5625, 54.5017, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.9083, 0.0000, -0.0659 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.4408,
1.1943, 0.9448 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 15
[15-01] : ( 1.7133, -1.6257, -1.1372 )  -> 47.4958
```

```
[15-02] : ( 0.7419, 0.9522, -0.4961 )   -> 42.3828
[15-03] : ( 1.2064, 6.4750, -0.0142 )   -> 57.0051
[15-04] : ( -1.9185, -3.5936, 1.8928 )  -> 57.4672
[15-05] : ( 0.9611, 1.9136, -3.3617 )   -> 214.3770
[15-06] : ( -2.0734, 1.3457, -1.5242 )  -> 167.4312
[15-07] : ( 0.2332, 2.0353, -0.3822 )   -> 35.3646
[15-08] : ( 3.1703, 3.5890, 0.2001 )    -> 62.2904
[15-09] : ( 0.5353, 1.8782, -0.1948 )   -> 37.9037
[15-10] : ( -0.0828, 2.6982, -2.2579 )  -> 207.2633
--- Velocidades: [vector( -0.0276, -1.4571, -0.0882 ), vector( 0.8486, 1.7544, 1.7673 ),
vector( -0.9282, -1.7982, 0.7238 ), vector( -1.5731, -1.5936, -0.0980 ), vector( -
0.0953, -0.1277, -0.1265 ), vector( 0.5828, 0.7918, 0.3930 ), vector( 1.3254, 2.0000,
0.7898 ), vector( -0.5074, -0.2584, -0.1086 ), vector( 1.5345, 1.2114, -1.8506 ),
vector( -0.0341, 0.5224, -0.6317 )]
--- Melhores resultados individuais: [37.8314, 42.3828, 56.4691, 54.9913, 56.0172,
35.4443, 28.6424, 19.5625, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.7419, 0.9522, -0.4961 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 16
[16-01] : ( 1.3656, -1.8910, -1.1048 )  -> 44.9186
[16-02] : ( 1.5633, 2.7264, 1.2638 )    -> 54.9315
[16-03] : ( 0.0592, 4.8355, 0.9213 )    -> 57.7524
[16-04] : ( -2.2241, -3.8400, 1.7667 )  -> 59.5688
[16-05] : ( 0.7531, 1.6794, -2.9947 )   -> 216.4257
[16-06] : ( -1.0068, 1.8676, -0.9512 )  -> 63.2295
[16-07] : ( 1.4809, 3.8149, 0.3146 )    -> 57.5420
[16-08] : ( 1.3983, 2.6971, -0.1746 )   -> 32.3400
[16-09] : ( 2.0198, 3.1011, -2.0709 )   -> 199.7335
[16-10] : ( -0.1016, 3.0564, -2.5749 )  -> 206.1851
--- Velocidades: [vector( -0.3477, -0.2652, 0.0324 ), vector( 0.8214, 1.7742, 1.7599 ),
vector( -1.1472, -1.6396, 0.9355 ), vector( -0.3056, -0.2464, -0.1261 ), vector( -
0.2081, -0.2343, 0.3670 ), vector( 1.0666, 0.5219, 0.5729 ), vector( 1.2477, 1.7796,
0.6969 ), vector( -1.7720, -0.8919, -0.3747 ), vector( 1.4844, 1.2229, -1.8761 ),
vector( -0.0188, 0.3582, -0.3170 )]
--- Melhores resultados individuais: [37.8314, 42.3828, 56.4691, 54.9913, 56.0172,
35.4443, 28.6424, 19.5625, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.7419, 0.9522, -0.4961 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 2.0009, -0.4671 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 06-08
--- Melhor Estratégia: vector( 0.0000, 2.0009, -0.4671 )
--- Resultado da Melhor Estratégia: 19.5625


--- Iteração: 17
[17-01] : ( 0.9667, -1.6649, -1.0341 )  -> 44.2358
[17-02] : ( 1.8702, 3.7137, 2.1315 )    -> 57.0090
[17-03] : ( -0.9669, 3.6858, 1.7815 )   -> 60.7717
[17-04] : ( -1.4188, -2.7796, 1.4026 )  -> 58.1442
[17-05] : ( 0.4556, 1.3658, -2.1618 )   -> 171.8405
[17-06] : ( 0.5398, 1.7509, -0.2582 )   -> 36.4724
[17-07] : ( 2.1679, 4.6075, 0.6227 )    -> 56.3939
[17-08] : ( -0.6017, 1.6027, -0.6344 )  -> 13.6000
[17-09] : ( 2.8152, 3.8570, -3.2455 )   -> 216.3000
[17-10] : ( -0.0705, 2.6930, -1.9483 )  -> 137.3877
--- Velocidades: [vector( -0.3989, 0.2260, 0.0707 ), vector( 0.3069, 0.9872, 0.8677 ),
vector( -1.0261, -1.1497, 0.8602 ), vector( 0.8053, 1.0604, -0.3641 ), vector( -0.2974,
-0.3136, 0.8329 ), vector( 1.5466, -0.1167, 0.6930 ), vector( 0.6870, 0.7926, 0.3080 ),
vector( -2.0000, -1.0945, -0.4598 ), vector( 0.7955, 0.7559, -1.1746 ), vector( 0.0311,
-0.3634, 0.6266 )]
--- Melhores resultados individuais: [37.8314, 42.3828, 56.4691, 54.9913, 56.0172,
35.4443, 28.6424, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 1.6848, 0.6610, -0.9887 ),
vector( 0.7419, 0.9522, -0.4961 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.8471, 1.9633, -0.5891 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
```

```
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 18
[18-01] : ( 0.4763, -1.1084, -0.9295 )  -> 35.6179
[18-02] : ( 1.6358, 3.6338, 1.9056 )    -> 58.1756
[18-03] : ( -1.7288, 2.3445, 2.3751 )   -> 58.5870
[18-04] : ( -0.0093, -1.0699, 1.0130 )  -> 56.1714
[18-05] : ( 0.1666, 1.0968, -0.8378 )   -> 28.1542
[18-06] : ( 1.9127, 1.3891, 0.2954 )    -> 52.6402
[18-07] : ( 2.6059, 4.9779, 0.7672 )    -> 55.7227
[18-08] : ( -2.0000, 0.5082, -1.0942 )  -> 46.2814
[18-09] : ( 3.1560, 4.2347, -3.9128 )   -> 211.7986
[18-10] : ( -0.0119, 1.8817, -0.8212 )  -> 60.2207
--- Velocidades: [vector( -0.4904, 0.5566, 0.1046 ), vector( -0.2345, -0.0798, -0.2259
), vector( -0.7619, -1.3413, 0.5937 ), vector( 1.4095, 1.7097, -0.3896 ), vector( -
0.2890, -0.2690, 1.3240 ), vector( 1.3729, -0.3619, 0.5536 ), vector( 0.4380, 0.3704,
0.1445 ), vector( -2.0000, -1.0945, -0.4598 ), vector( 0.3407, 0.3778, -0.6673 ),
vector( 0.0586, -0.8113, 1.1270 )]
--- Melhores resultados individuais: [35.6179, 42.3828, 56.4691, 54.9913, 28.1542,
35.4443, 28.6424, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.4763, 0.0000, -0.9295 ),
vector( 0.7419, 0.9522, -0.4961 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.5163,
0.0000, 1.9134 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 19
[19-01] : ( -0.0142, 0.5566, -0.8250 )  -> 43.6341
[19-02] : ( 1.1921, 2.9260, 1.1179 )    -> 55.4742
[19-03] : ( -1.9199, 1.2085, 2.1995 )   -> 59.5382
[19-04] : ( 1.9748, 0.9301, 0.8898 )    -> 50.0051
[19-05] : ( -0.1224, 0.8278, 0.4862 )   -> 54.1902
[19-06] : ( 2.5264, 0.5572, 0.4286 )    -> 52.5036
[19-07] : ( 2.5581, 4.6713, 0.6386 )    -> 57.3890
[19-08] : ( -3.7044, -0.4245, -1.4861 ) -> 61.0145
[19-09] : ( 3.1818, 4.3485, -4.2453 )   -> 217.6945
[19-10] : ( 0.0511, 0.9350, 0.3567 )    -> 57.3970
--- Velocidades: [vector( -0.4904, 0.5566, 0.1046 ), vector( -0.4437, -0.7079, -0.7877
), vector( -0.1911, -1.1360, -0.1756 ), vector( 1.9840, 2.0000, -0.1232 ), vector( -
0.2890, -0.2690, 1.3240 ), vector( 0.6137, -0.8319, 0.1332 ), vector( -0.0478, -0.3067,
-0.1286 ), vector( -1.7044, -0.9327, -0.3919 ), vector( 0.0259, 0.1138, -0.3324 ),
vector( 0.0630, -0.9466, 1.1779 )]
--- Melhores resultados individuais: [35.6179, 42.3828, 56.4691, 50.0051, 28.1542,
35.4443, 28.6424, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.4763, 0.0000, -0.9295 ),
vector( 0.7419, 0.9522, -0.4961 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 20
[20-01] : ( -0.2618, 0.8376, -0.7722 )  -> 34.9196
[20-02] : ( 0.6801, 1.9379, 0.1320 )    -> 38.9841
[20-03] : ( -1.4534, 1.3888, 1.5248 )   -> 59.3753
[20-04] : ( 3.6935, 2.9301, 0.5131 )    -> 60.6620
[20-05] : ( -0.2746, 0.6861, 1.1837 )   -> 61.2543
[20-06] : ( 2.1915, -0.3369, 0.1206 )   -> 48.0790
[20-07] : ( 1.6377, 3.1379, 0.0313 )    -> 45.2489
[20-08] : ( -5.0391, -1.1549, -1.7929 ) -> 136.1042
[20-09] : ( 2.8026, 4.1028, -4.0620 )   -> 213.6690
[20-10] : ( 0.0936, 0.2331, 1.1234 )    -> 59.8555
--- Velocidades: [vector( -0.2477, 0.2811, 0.0528 ), vector( -0.5120, -0.9881, -0.9859
), vector( 0.4665, 0.1803, -0.6747 ), vector( 1.7188, 2.0000, -0.3766 ), vector( -
0.1522, -0.1417, 0.6975 ), vector( -0.3349, -0.8940, -0.3080 ), vector( -0.9203, -
1.5333, -0.6073 ), vector( -1.3347, -0.7304, -0.3069 ), vector( -0.3792, -0.2457, 0.1833
), vector( 0.0425, -0.7020, 0.7668 )]
```

```
--- Melhores resultados individuais: [34.9196, 38.9841, 56.4691, 50.0051, 28.1542,
35.4443, 28.6424, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 0.8376, -0.7722 ),
vector( 0.6801, 1.9379, 0.1320 ), vector( 0.6197, 7.2047, 0.5815 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.0000, 0.8907, -0.8106 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 21
[21-01] : ( -0.2477, 1.1187, -0.7194 )  -> 35.8727
[21-02] : ( 0.0982, 0.8367, -0.9468 )   -> 35.0621
[21-03] : ( -0.0843, 3.3888, 0.2865 )   -> 47.9464
[21-04] : ( 5.3501, 4.8674, 0.1410 )    -> 68.0290
[21-05] : ( -0.2496, 0.7093, 1.0693 )   -> 59.1007
[21-06] : ( 0.6848, -0.7966, -0.6469 )  -> 41.7357
[21-07] : ( 0.6211, 1.5065, -0.6172 )   -> 27.3670
[21-08] : ( -5.3932, -1.3486, -1.8743 ) -> 169.4172
[21-09] : ( 1.8321, 3.3149, -3.0765 )   -> 213.3876
[21-10] : ( 0.1134, -0.1630, 1.4477 )   -> 59.8252
--- Velocidades: [vector( -0.2477, 0.2811, 0.0528 ), vector( -0.5819, -1.1012, -1.0788
), vector( 1.3690, 2.0000, -1.2383 ), vector( 1.6565, 1.9373, -0.3721 ), vector( 0.0250,
0.0232, -0.1143 ), vector( -1.5067, -0.4597, -0.7675 ), vector( -1.0166, -1.6315, -
0.6484 ), vector( -0.3541, -0.1938, -0.0814 ), vector( -0.9706, -0.7879, 0.9855 ),
vector( 0.0197, -0.3960, 0.3242 )]
--- Melhores resultados individuais: [34.9196, 35.0621, 47.9464, 50.0051, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 0.8376, -0.7722 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 22
[22-01] : ( -0.3906, 1.2809, -0.6889 )  -> 33.9180
[22-02] : ( -0.4992, -0.2644, -1.9981 ) -> 146.2178
[22-03] : ( 1.3916, 5.0782, -1.1041 )   -> 109.3383
[22-04] : ( 5.9019, 5.7824, -0.2351 )   -> 75.2566
[22-05] : ( -0.0911, 0.8556, 0.5647 )   -> 53.8433
[22-06] : ( -0.9645, -0.9481, -1.4296 ) -> 65.0395
[22-07] : ( -0.4856, -0.1110, -1.2681 ) -> 48.9253
[22-08] : ( -3.6536, -0.3967, -1.4744 ) -> 84.2817
[22-09] : ( 0.5255, 2.2106, -1.6300 )   -> 174.7590
[22-10] : ( 0.0943, 0.0239, 1.0473 )    -> 38.1340
--- Velocidades: [vector( -0.1430, 0.1622, 0.0305 ), vector( -0.5974, -1.1010, -1.0513
), vector( 1.3916, 1.6894, -1.3907 ), vector( 0.5519, 0.9150, -0.3761 ), vector( 0.1586,
0.1463, -0.5047 ), vector( -1.6493, -0.1516, -0.7827 ), vector( -1.1067, -1.6175, -
0.6509 ), vector( 1.7395, 0.9519, 0.3999 ), vector( -1.3065, -1.1044, 1.4465 ), vector(
-0.0191, 0.1869, -0.4004 )]
--- Melhores resultados individuais: [33.9180, 35.0621, 47.9464, 50.0051, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.2809, -0.6889 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 23
[23-01] : ( -0.1430, 1.4432, -0.6584 )  -> 28.8462
[23-02] : ( -0.8279, -0.7479, -2.4923 ) -> 174.0200
[23-03] : ( 2.3653, 6.0942, -2.1409 )   -> 195.9668
[23-04] : ( 5.8691, 6.1731, -0.6290 )   -> 83.9785
[23-05] : ( 0.1469, 1.0752, -0.1985 )   -> 48.6509
[23-06] : ( -2.2671, -0.5414, -1.9596 ) -> 185.4060
[23-07] : ( -1.1967, -1.1497, -1.6861 ) -> 91.9263
[23-08] : ( -1.6536, 0.9273, -0.9181 )  -> 37.2868
```

```
[23-09] : ( -0.8802, 0.9716, 0.0879 )   -> 52.4630
[23-10] : ( 0.0511, 0.5995, 0.2065 )    -> 52.9698
--- Velocidades: [vector( -0.1430, 0.1622, 0.0305 ), vector( -0.3287, -0.4835, -0.4942
), vector( 0.9737, 1.0159, -1.0368 ), vector( -0.0328, 0.3906, -0.3939 ), vector(
0.2380, 0.2196, -0.7632 ), vector( -1.3026, 0.4067, -0.5300 ), vector( -0.7111, -1.0387,
-0.4180 ), vector( 2.0000, 1.3240, 0.5563 ), vector( -1.4058, -1.2389, 1.7179 ), vector(
-0.0432, 0.5756, -0.8408 )]
--- Melhores resultados individuais: [28.8462, 35.0621, 47.9464, 50.0051, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.4432, -0.6584 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 24
[24-01] : ( -0.1430, 1.6054, -0.6280 )  -> 42.8552
[24-02] : ( -0.8896, -0.6525, -2.4696 ) -> 162.3396
[24-03] : ( 2.6758, 6.0540, -2.6115 )   -> 200.9939
[24-04] : ( 4.7146, 5.3131, -0.8282 )   -> 85.0522
[24-05] : ( 0.4027, 1.3122, -1.1659 )   -> 92.0393
[24-06] : ( -2.7484, 0.3190, -2.0395 )  -> 168.6906
[24-07] : ( -1.2474, -1.1561, -1.6925 ) -> 141.8243
[24-08] : ( 0.3464, 2.4465, -0.2799 )   -> 34.4378
[24-09] : ( -2.1134, -0.1493, 1.7111 )  -> 60.2037
[24-10] : ( -0.0120, 1.5422, -0.9746 )  -> 46.5435
--- Velocidades: [vector( -0.1430, 0.1622, 0.0305 ), vector( -0.0617, 0.0954, 0.0226 ),
vector( 0.3106, -0.0402, -0.4706 ), vector( -1.1545, -0.8600, -0.1992 ), vector( 0.2558,
0.2369, -0.9673 ), vector( -0.4812, 0.8604, -0.0798 ), vector( -0.0507, -0.0064, -0.0064
), vector( 2.0000, 1.5192, 0.6383 ), vector( -1.2332, -1.1210, 1.6232 ), vector( -
0.0631, 0.9427, -1.1811 )]
--- Melhores resultados individuais: [28.8462, 35.0621, 47.9464, 50.0051, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.4432, -0.6584 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 25
[25-01] : ( -0.2507, 1.7277, -0.6050 )  -> 27.6475
[25-02] : ( -0.6104, 0.0446, -1.8703 )  -> 162.5616
[25-03] : ( 2.3016, 5.1270, -2.4190 )   -> 198.9926
[25-04] : ( 2.7146, 3.3131, -0.3968 )   -> 71.3004
[25-05] : ( 0.6897, 1.5766, -2.0094 )   -> 168.9841
[25-06] : ( -3.0188, 1.2482, -2.0143 )  -> 153.5511
[25-07] : ( -0.4304, 0.2602, -1.1349 )  -> 32.0772
[25-08] : ( 2.2561, 3.7457, 0.2660 )    -> 56.4020
[25-09] : ( -2.4851, -0.5853, 2.5575 )  -> 56.3270
[25-10] : ( -0.0724, 2.4757, -2.0905 )  -> 167.3654
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.2792, 0.6971, 0.5993 ),
vector( -0.3742, -0.9270, 0.1925 ), vector( -2.0000, -2.0000, 0.4314 ), vector( 0.2870,
0.2644, -0.8435 ), vector( -0.2704, 0.9292, 0.0252 ), vector( 0.8171, 1.4164, 0.5576 ),
vector( 1.9097, 1.2992, 0.5458 ), vector( -0.3717, -0.4360, 0.8464 ), vector( -0.0603,
0.9335, -1.1160 )]
--- Melhores resultados individuais: [27.6475, 35.0621, 47.9464, 50.0051, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.7277, -0.6050 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 1.9748,
0.9301, 0.8898 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 26
[26-01] : ( -0.1077, 1.8499, -0.5820 )  -> 27.2675
```

```
[26-02] : ( -0.1034, 1.0494, -0.9492 )  -> 52.1556
[26-03] : ( 1.3982, 3.6248, -1.6988 )   -> 181.7789
[26-04] : ( 0.7146, 1.3131, 0.0007 )    -> 45.8632
[26-05] : ( 0.7790, 1.6592, -2.3400 )   -> 180.2276
[26-06] : ( -2.6408, 2.0524, -1.7113 )  -> 151.2670
[26-07] : ( 0.7966, 2.2245, -0.3539 )   -> 50.7392
[26-08] : ( 3.8131, 4.7099, 0.6711 )    -> 63.0343
[26-09] : ( -2.4941, -0.7243, 3.0665 )  -> 61.0823
[26-10] : ( -0.1085, 3.1009, -2.7293 )  -> 204.3124
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.5070, 1.0048, 0.9211 ),
vector( -0.9034, -1.5022, 0.7202 ), vector( -2.0000, -2.0000, 0.3975 ), vector( 0.0893,
0.0827, -0.3307 ), vector( 0.3780, 0.8042, 0.3030 ), vector( 1.2269, 1.9643, 0.7810 ),
vector( 1.5571, 0.9643, 0.4051 ), vector( -0.0090, -0.1390, 0.5089 ), vector( -0.0362,
0.6252, -0.6388 )]
--- Melhores resultados individuais: [27.2675, 35.0621, 47.9464, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.8499, -0.5820 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 27
[27-01] : ( -0.1077, 1.9722, -0.5591 )  -> 19.9809
[27-02] : ( 0.4616, 2.0930, 0.0129 )    -> 44.0434
[27-03] : ( -0.0817, 1.9810, -0.1848 )  -> 42.7170
[27-04] : ( -1.2854, -0.6869, 0.3424 )  -> 53.6379
[27-05] : ( 0.6369, 1.5282, -1.9016 )   -> 168.5349
[27-06] : ( -2.0798, 2.7751, -1.3383 )  -> 145.1361
[27-07] : ( 1.8801, 4.0107, 0.3536 )    -> 54.1275
[27-08] : ( 4.2492, 4.7607, 0.6924 )    -> 65.1396
[27-09] : ( -2.3112, -0.6945, 3.3473 )  -> 57.3959
[27-10] : ( -0.1194, 3.3629, -2.8889 )  -> 210.1224
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.5651, 1.0435, 0.9620 ),
vector( -1.4800, -1.6438, 1.5140 ), vector( -2.0000, -2.0000, 0.3418 ), vector( -0.1421,
-0.1310, 0.4384 ), vector( 0.5610, 0.7227, 0.3729 ), vector( 1.0835, 1.7862, 0.7075 ),
vector( 0.4360, 0.0507, 0.0213 ), vector( 0.1829, 0.0298, 0.2808 ), vector( -0.0109,
0.2619, -0.1596 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 28
[28-01] : ( -0.1077, 2.0945, -0.5361 )  -> 35.5529
[28-02] : ( 0.8499, 2.6357, 0.5564 )    -> 56.8285
[28-03] : ( -1.4800, 0.3365, 1.3027 )   -> 60.2578
[28-04] : ( -2.4409, -1.8251, 0.4915 )  -> 46.4022
[28-05] : ( 0.3108, 1.2276, -0.9204 )   -> 55.5120
[28-06] : ( -1.0906, 2.9599, -0.8479 )  -> 77.1103
[28-07] : ( 2.7177, 5.3472, 0.8852 )    -> 57.9519
[28-08] : ( 2.7918, 3.4042, 0.1225 )    -> 54.7071
[28-09] : ( -1.4511, -0.0226, 2.6261 )  -> 52.6405
[28-10] : ( -0.0896, 2.9959, -2.2949 )  -> 199.7207
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.3882, 0.5428, 0.5435 ),
vector( -1.4800, -1.6444, 1.4876 ), vector( -1.1554, -1.1382, 0.1491 ), vector( -0.3261,
-0.3006, 0.9812 ), vector( 0.9892, 0.1848, 0.4904 ), vector( 0.8376, 1.3365, 0.5316 ),
vector( -1.4574, -1.3565, -0.5699 ), vector( 0.8601, 0.6719, -0.7212 ), vector( 0.0299,
-0.3670, 0.5941 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
```

```
[26-02] : ( -0.1034, 1.0494, -0.9492 )  -> 52.1556
[26-03] : ( 1.3982, 3.6248, -1.6988 )   -> 181.7789
[26-04] : ( 0.7146, 1.3131, 0.0007 )    -> 45.8632
[26-05] : ( 0.7790, 1.6592, -2.3400 )   -> 180.2276
[26-06] : ( -2.6408, 2.0524, -1.7113 )  -> 151.2670
[26-07] : ( 0.7966, 2.2245, -0.3539 )   -> 50.7392
[26-08] : ( 3.8131, 4.7099, 0.6711 )    -> 63.0343
[26-09] : ( -2.4941, -0.7243, 3.0665 )  -> 61.0823
[26-10] : ( -0.1085, 3.1009, -2.7293 )  -> 204.3124
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.5070, 1.0048, 0.9211 ),
vector( -0.9034, -1.5022, 0.7202 ), vector( -2.0000, -2.0000, 0.3975 ), vector( 0.0893,
0.0827, -0.3307 ), vector( 0.3780, 0.8042, 0.3030 ), vector( 1.2269, 1.9643, 0.7810 ),
vector( 1.5571, 0.9643, 0.4051 ), vector( -0.0090, -0.1390, 0.5089 ), vector( -0.0362,
0.6252, -0.6388 )]
--- Melhores resultados individuais: [27.2675, 35.0621, 47.9464, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.8499, -0.5820 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 3.3888, 0.2865 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 27
[27-01] : ( -0.1077, 1.9722, -0.5591 )  -> 19.9809
[27-02] : ( 0.4616, 2.0930, 0.0129 )    -> 44.0434
[27-03] : ( -0.0817, 1.9810, -0.1848 )  -> 42.7170
[27-04] : ( -1.2854, -0.6869, 0.3424 )  -> 53.6379
[27-05] : ( 0.6369, 1.5282, -1.9016 )   -> 168.5349
[27-06] : ( -2.0798, 2.7751, -1.3383 )  -> 145.1361
[27-07] : ( 1.8801, 4.0107, 0.3536 )    -> 54.1275
[27-08] : ( 4.2492, 4.7607, 0.6924 )    -> 65.1396
[27-09] : ( -2.3112, -0.6945, 3.3473 )  -> 57.3959
[27-10] : ( -0.1194, 3.3629, -2.8889 )  -> 210.1224
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.5651, 1.0435, 0.9620 ),
vector( -1.4800, -1.6438, 1.5140 ), vector( -2.0000, -2.0000, 0.3418 ), vector( -0.1421,
-0.1310, 0.4384 ), vector( 0.5610, 0.7227, 0.3729 ), vector( 1.0835, 1.7862, 0.7075 ),
vector( 0.4360, 0.0507, 0.0213 ), vector( 0.1829, 0.0298, 0.2808 ), vector( -0.0109,
0.2619, -0.1596 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 28
[28-01] : ( -0.1077, 2.0945, -0.5361 )  -> 35.5529
[28-02] : ( 0.8499, 2.6357, 0.5564 )    -> 56.8285
[28-03] : ( -1.4800, 0.3365, 1.3027 )   -> 60.2578
[28-04] : ( -2.4409, -1.8251, 0.4915 )  -> 46.4022
[28-05] : ( 0.3108, 1.2276, -0.9204 )   -> 55.5120
[28-06] : ( -1.0906, 2.9599, -0.8479 )  -> 77.1103
[28-07] : ( 2.7177, 5.3472, 0.8852 )    -> 57.9519
[28-08] : ( 2.7918, 3.4042, 0.1225 )    -> 54.7071
[28-09] : ( -1.4511, -0.0226, 2.6261 )  -> 52.6405
[28-10] : ( -0.0896, 2.9959, -2.2949 )  -> 199.7207
--- Velocidades: [vector( -0.1077, 0.1223, 0.0230 ), vector( 0.3882, 0.5428, 0.5435 ),
vector( -1.4800, -1.6444, 1.4876 ), vector( -1.1554, -1.1382, 0.1491 ), vector( -0.3261,
-0.3006, 0.9812 ), vector( 0.9892, 0.1848, 0.4904 ), vector( 0.8376, 1.3365, 0.5316 ),
vector( -1.4574, -1.3565, -0.5699 ), vector( 0.8601, 0.6719, -0.7212 ), vector( 0.0299,
-0.3670, 0.5941 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4443, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0000, 0.0000, -0.7828 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
```

```
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 29
[29-01] : ( -0.1920, 2.1901, -0.5181 )  -> 47.0942
[29-02] : ( 1.1547, 3.0226, 0.9517 )    -> 56.0552
[29-03] : ( -2.4168, -0.7057, 2.1895 )  -> 62.3503
[29-04] : ( -2.5680, -1.9020, 0.3721 )  -> 55.1900
[29-05] : ( 0.0061, 0.9462, 0.0821 )    -> 56.4741
[29-06] : ( 0.0384, 2.8924, -0.3374 )   -> 35.4188
[29-07] : ( 3.0318, 5.8097, 1.0711 )    -> 56.8449
[29-08] : ( 0.7918, 1.4042, -0.7359 )   -> 51.1316
[29-09] : ( 0.2915, 1.5245, 0.6261 )    -> 49.7598
[29-10] : ( -0.0389, 2.2878, -1.3253 )  -> 132.7275
--- Velocidades: [vector( -0.0843, 0.0956, 0.0180 ), vector( 0.3048, 0.3869, 0.3953 ),
vector( -0.9368, -1.0422, 0.8868 ), vector( -0.1271, -0.0769, -0.1194 ), vector( -
0.3047, -0.2814, 1.0026 ), vector( 1.1289, -0.0676, 0.5105 ), vector( 0.3141, 0.4625,
0.1859 ), vector( -2.0000, -2.0000, -0.8584 ), vector( 1.7426, 1.5471, -2.0000 ),
vector( 0.0507, -0.7080, 0.9696 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 30
[30-01] : ( -0.2040, 2.2037, -0.5156 )  -> 42.3485
[30-02] : ( 1.3137, 3.1306, 1.0947 )    -> 55.8561
[30-03] : ( -2.2267, -0.4962, 1.9234 )  -> 57.8066
[30-04] : ( -0.9734, -0.2290, -0.1208 ) -> 51.4940
[30-05] : ( -0.1913, 0.7628, 0.9423 )   -> 46.7816
[30-06] : ( 1.1632, 2.6865, 0.1413 )    -> 46.0031
[30-07] : ( 2.5142, 4.8951, 0.7119 )    -> 58.5819
[30-08] : ( -1.2082, -0.5170, -1.5539 ) -> 127.1016
[30-09] : ( 2.0083, 3.0874, -1.3739 )   -> 148.4137
[30-10] : ( 0.0288, 1.2527, -0.0697 )   -> 51.5811
--- Velocidades: [vector( -0.0120, 0.0136, 0.0025 ), vector( 0.1590, 0.1080, 0.1430 ),
vector( 0.1901, 0.2095, -0.2662 ), vector( 1.5945, 1.6729, -0.4929 ), vector( -0.1974, -
0.1835, 0.8602 ), vector( 1.1248, -0.2058, 0.4787 ), vector( -0.5176, -0.9146, -0.3592
), vector( -2.0000, -1.9212, -0.8181 ), vector( 1.7168, 1.5629, -2.0000 ), vector(
0.0677, -1.0352, 1.2556 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 31
[31-01] : ( -0.1795, 2.1759, -0.5208 )  -> 34.6425
[31-02] : ( 1.1526, 2.6734, 0.7163 )    -> 55.2572
[31-03] : ( -1.5019, 0.3080, 1.1486 )   -> 46.9128
[31-04] : ( 1.0266, 1.7710, -0.6060 )   -> 50.2495
[31-05] : ( -0.1961, 0.7564, 1.2815 )   -> 57.4214
[31-06] : ( 1.9979, 2.3891, 0.4652 )    -> 54.2040
[31-07] : ( 1.8277, 3.7004, 0.2417 )    -> 52.9211
[31-08] : ( -2.9177, -1.9284, -2.1509 ) -> 196.9584
[31-09] : ( 3.1739, 4.2044, -2.9675 )   -> 211.8499
[31-10] : ( 0.0878, 0.2901, 0.9990 )    -> 48.7860
--- Velocidades: [vector( 0.0245, -0.0278, -0.0052 ), vector( -0.1611, -0.4572, -0.3784
), vector( 0.7248, 0.8042, -0.7748 ), vector( 2.0000, 2.0000, -0.4852 ), vector( -
0.0048, -0.0063, 0.3392 ), vector( 0.8347, -0.2975, 0.3239 ), vector( -0.6865, -1.1948,
-0.4701 ), vector( -1.7094, -1.4114, -0.5969 ), vector( 1.1656, 1.1170, -1.5936 ),
vector( 0.0591, -0.9626, 1.0687 )]
```

```
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 32
[32-01] : ( -0.1178, 2.1060, -0.5339 )  -> 35.4171
[32-02] : ( 0.8979, 2.0727, 0.1983 )    -> 45.1383
[32-03] : ( -0.2383, 1.7124, -0.1107 )  -> 53.4733
[32-04] : ( 2.8630, 3.6222, -1.0689 )   -> 100.5292
[32-05] : ( -0.0778, 0.8643, 1.1316 )   -> 50.5548
[32-06] : ( 2.1584, 2.0940, 0.4760 )    -> 54.6671
[32-07] : ( 0.8070, 1.9367, -0.4529 )   -> 45.3113
[32-08] : ( -3.5309, -2.0132, -2.1780 ) -> 173.5603
[32-09] : ( 3.3707, 4.5042, -3.7317 )   -> 214.0060
[32-10] : ( 0.1074, -0.1156, 1.3153 )   -> 57.9853
--- Velocidades: [vector( 0.0616, -0.0699, -0.0131 ), vector( -0.2547, -0.6007, -0.5180
), vector( 1.2637, 1.4043, -1.2593 ), vector( 1.8364, 1.8513, -0.4628 ), vector( 0.1183,
0.1079, -0.1499 ), vector( 0.1605, -0.2951, 0.0108 ), vector( -1.0207, -1.7637, -0.6946
), vector( -0.6133, -0.0848, -0.0272 ), vector( 0.1968, 0.2998, -0.7641 ), vector(
0.0196, -0.4057, 0.3163 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 33
[33-01] : ( -0.0481, 2.0268, -0.5488 )  -> 41.2417
[33-02] : ( 0.1628, 0.9827, -0.9036 )   -> 36.8706
[33-03] : ( 1.1257, 3.2293, -1.4180 )   -> 148.5934
[33-04] : ( 4.1842, 4.9267, -1.2962 )   -> 138.4830
[33-05] : ( 0.1555, 1.0782, 0.6446 )    -> 56.6125
[33-06] : ( 1.7621, 1.7759, 0.2228 )    -> 52.2502
[33-07] : ( -0.2765, 0.0811, -1.1845 )  -> 40.4926
[33-08] : ( -3.5729, -1.5129, -1.9554 ) -> 97.6889
[33-09] : ( 2.4187, 3.7736, -3.2315 )   -> 209.0920
[33-10] : ( 0.0874, 0.0769, 0.8908 )    -> 55.4993
--- Velocidades: [vector( 0.0698, -0.0792, -0.0149 ), vector( -0.7351, -1.0900, -1.1020
), vector( 1.3639, 1.5170, -1.3072 ), vector( 1.3212, 1.3045, -0.2274 ), vector( 0.2334,
0.2139, -0.4870 ), vector( -0.3963, -0.3182, -0.2532 ), vector( -1.0835, -1.8557, -
0.7316 ), vector( -0.0419, 0.5003, 0.2226 ), vector( -0.9520, -0.7306, 0.5001 ), vector(
-0.0201, 0.1925, -0.4245 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6651]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0000, 1.4916, -0.6978 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 34
[34-01] : ( 0.0262, 1.9425, -0.5646 )   -> 21.2915
[34-02] : ( -0.6179, 0.0323, -1.9550 )  -> 169.9291
[34-03] : ( 2.0732, 4.2834, -2.3208 )   -> 203.1499
[34-04] : ( 4.6566, 5.4115, -1.3955 )   -> 153.3630
[34-05] : ( 0.4097, 1.3129, -0.2087 )   -> 52.9959
[34-06] : ( 1.0171, 1.6652, -0.1477 )   -> 43.3752
[34-07] : ( -1.0632, -1.2452, -1.7083 ) -> 115.5706
[34-08] : ( -2.6644, -0.1839, -1.3815 ) -> 77.6607
```

```
[34-09] : ( 0.4187, 1.9050, -1.2315 )   -> 123.3416
[34-10] : ( 0.0395, 0.7264, -0.0355 )   -> 35.6056
--- Velocidades: [vector( 0.0742, -0.0842, -0.0158 ), vector( -0.7807, -0.9503, -1.0514
), vector( 0.9476, 1.0540, -0.9029 ), vector( 0.4724, 0.4848, -0.0992 ), vector( 0.2541,
0.2348, -0.8533 ), vector( -0.7450, -0.1107, -0.3705 ), vector( -0.7866, -1.3262, -
0.5239 ), vector( 0.9084, 1.3290, 0.5740 ), vector( -2.0000, -1.8686, 2.0000 ), vector(
-0.0479, 0.6495, -0.9263 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6056]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0395, 0.7264, -0.0355 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 35
[35-01] : ( 0.0905, 1.8695, -0.5784 )   -> 35.3586
[35-02] : ( -1.1289, -0.5361, -2.5934 ) -> 201.7750
[35-03] : ( 2.5968, 4.8652, -2.8453 )   -> 210.0590
[35-04] : ( 4.1674, 4.9205, -1.2243 )   -> 129.8126
[35-05] : ( 0.6580, 1.5431, -1.1781 )   -> 112.6140
[35-06] : ( 0.0725, 1.7887, -0.5604 )   -> 48.7420
[35-07] : ( -1.5715, -2.0668, -2.0346 ) -> 187.0578
[35-08] : ( -1.6502, 1.2160, -0.7778 )  -> 28.7070
[35-09] : ( -1.5636, 0.0314, 0.7685 )   -> 57.8208
[35-10] : ( -0.0102, 1.4161, -0.9893 )  -> 53.1241
--- Velocidades: [vector( 0.0644, -0.0730, -0.0137 ), vector( -0.5110, -0.5684, -0.6383
), vector( 0.5236, 0.5818, -0.5245 ), vector( -0.4892, -0.4910, 0.1712 ), vector(
0.2484, 0.2302, -0.9693 ), vector( -0.9446, 0.1236, -0.4126 ), vector( -0.5083, -0.8217,
-0.3263 ), vector( 1.0142, 1.3999, 0.6036 ), vector( -1.9823, -1.8735, 2.0000 ), vector(
-0.0497, 0.6897, -0.9538 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6056]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0395, 0.7264, -0.0355 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 36
[36-01] : ( 0.1353, 1.8186, -0.5879 )   -> 20.8016
[36-02] : ( -1.1876, -0.4726, -2.5724 ) -> 185.9160
[36-03] : ( 2.4299, 4.6791, -2.7040 )   -> 196.0073
[36-04] : ( 3.3330, 4.0917, -0.9968 )   -> 86.7271
[36-05] : ( 0.7300, 1.6128, -1.9678 )   -> 169.4020
[36-06] : ( -0.8840, 2.0416, -0.9506 )  -> 75.8193
[36-07] : ( -1.4902, -1.8063, -1.9377 ) -> 162.8230
[36-08] : ( -0.1160, 2.7378, -0.1290 )  -> 53.7872
[36-09] : ( -3.3268, -1.6453, 2.6536 )  -> 57.1855
[36-10] : ( -0.0401, 1.8632, -1.5494 )  -> 151.9047
--- Velocidades: [vector( 0.0448, -0.0509, -0.0096 ), vector( -0.0588, 0.0635, 0.0210 ),
vector( -0.1669, -0.1861, 0.1414 ), vector( -0.8344, -0.8288, 0.2275 ), vector( 0.0719,
0.0696, -0.7898 ), vector( -0.9565, 0.2529, -0.3902 ), vector( 0.0813, 0.2605, 0.0969 ),
vector( 1.5342, 1.5218, 0.6488 ), vector( -1.7632, -1.6767, 1.8851 ), vector( -0.0300,
0.4471, -0.5601 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 45.8632, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6056]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 0.7146,
1.3131, 0.0007 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0395, 0.7264, -0.0355 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 37
[37-01] : ( 0.1323, 1.8221, -0.5873 )   -> 27.2268
```

```
[37-02] : ( -1.0937, -0.1691, -2.3233 ) -> 147.2624
[37-03] : ( 1.8647, 4.0505, -2.1622 )   -> 188.5642
[37-04] : ( 1.3330, 2.0917, -0.3710 )   -> 41.3478
[37-05] : ( 0.6385, 1.5323, -2.3687 )   -> 176.3669
[37-06] : ( -1.6766, 2.3218, -1.2586 )  -> 104.4917
[37-07] : ( -0.6310, -0.2610, -1.3319 ) -> 59.1030
[37-08] : ( 1.4704, 3.7490, 0.2925 )    -> 56.6319
[37-09] : ( -3.8971, -2.2324, 3.6540 )  -> 61.2453
[37-10] : ( -0.0452, 1.9862, -1.6198 )  -> 158.0214
--- Velocidades: [vector( -0.0031, 0.0035, 0.0007 ), vector( 0.0939, 0.3035, 0.2491 ),
vector( -0.5652, -0.6286, 0.5418 ), vector( -2.0000, -2.0000, 0.6258 ), vector( -0.0915,
-0.0804, -0.4009 ), vector( -0.7926, 0.2801, -0.3081 ), vector( 0.8592, 1.5453, 0.6058
), vector( 1.5864, 1.0112, 0.4215 ), vector( -0.5703, -0.5871, 1.0004 ), vector( -
0.0050, 0.1230, -0.0705 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 41.3478, 28.1542,
35.4188, 27.3670, 13.6000, 37.9037, 35.6056]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.6211, 1.5065, -0.6172 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0395, 0.7264, -0.0355 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 38
[38-01] : ( 0.1170, 1.8394, -0.5840 )   -> 35.1060
[38-02] : ( -0.6739, 0.6029, -1.6182 )  -> 133.6047
[38-03] : ( 0.7135, 2.7707, -1.0320 )   -> 100.7547
[38-04] : ( -0.6670, 0.0917, 0.2318 )   -> 46.1043
[38-05] : ( 0.3688, 1.2861, -1.9485 )   -> 168.3778
[38-06] : ( -1.9595, 2.5165, -1.3481 )  -> 155.7572
[38-07] : ( 0.5146, 1.7390, -0.5224 )   -> 20.6034
[38-08] : ( 2.7519, 4.3152, 0.5219 )    -> 59.4637
[38-09] : ( -3.4827, -1.8678, 3.6417 )  -> 61.6084
[38-10] : ( -0.0165, 1.6478, -1.0416 )  -> 70.7711
--- Velocidades: [vector( -0.0153, 0.0173, 0.0033 ), vector( 0.4198, 0.7720, 0.7050 ),
vector( -1.1513, -1.2798, 1.1302 ), vector( -2.0000, -2.0000, 0.6027 ), vector( -0.2697,
-0.2463, 0.4202 ), vector( -0.2829, 0.1947, -0.0894 ), vector( 1.1456, 2.0000, 0.8095 ),
vector( 1.2816, 0.5662, 0.2294 ), vector( 0.4144, 0.3646, -0.0123 ), vector( 0.0286, -
0.3384, 0.5783 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 42.7170, 41.3478, 28.1542,
35.4188, 20.6034, 13.6000, 37.9037, 35.6056]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.9810, -0.1848 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.5146, 1.7390, -0.5224 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0395, 0.7264, -0.0355 )]
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 39
[39-01] : ( 0.0799, 1.8815, -0.5761 )   -> 44.0670
[39-02] : ( 0.0616, 1.5136, -0.6212 )   -> 35.5456
[39-03] : ( -0.6991, 1.2001, 0.3393 )   -> 39.5976
[39-04] : ( -2.4456, -1.6532, 0.6070 )  -> 55.0094
[39-05] : ( 0.0758, 1.0745, -1.0023 )   -> 53.8750
[39-06] : ( -1.2823, 2.6873, -0.9961 )  -> 84.3564
[39-07] : ( 1.5776, 3.7171, 0.2691 )    -> 54.9999
[39-08] : ( 3.0712, 3.9329, 0.3469 )    -> 55.2058
[39-09] : ( -1.7131, -0.2270, 2.2811 )  -> 64.1791
[39-10] : ( 0.0136, 1.2966, -0.4306 )   -> 18.8779
--- Velocidades: [vector( -0.0371, 0.0421, 0.0079 ), vector( 0.7355, 0.9107, 0.9970 ),
vector( -1.4125, -1.5706, 1.3713 ), vector( -1.7786, -1.7449, 0.3752 ), vector( -0.2930,
-0.2116, 0.9462 ), vector( 0.6772, 0.1708, 0.3520 ), vector( 1.0630, 1.9781, 0.7915 ),
vector( 0.3192, -0.3823, -0.1750 ), vector( 1.7696, 1.6408, -1.3606 ), vector( 0.0301, -
0.3512, 0.6110 )]
--- Melhores resultados individuais: [19.9809, 35.0621, 39.5976, 41.3478, 28.1542,
35.4188, 20.6034, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0000, 1.9722, -0.5591 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.5146, 1.7390, -0.5224 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
```

```
--- Código da Melhor Estratégia: 17-08
--- Melhor Estratégia: vector( 0.0000, 1.6027, -0.6344 )
--- Resultado da Melhor Estratégia: 13.6000


--- Iteração: 40
[40-01] : ( 0.0130, 1.8327, -0.5370 )   -> 12.4000
[40-02] : ( 0.8057, 2.1968, 0.2845 )    -> 47.1386
[40-03] : ( -1.4125, -0.2818, 1.6074 )  -> 54.5430
[40-04] : ( -2.4121, -1.5753, 0.3849 )  -> 54.8042
[40-05] : ( -0.1492, 0.9358, 0.0351 )   -> 54.8619
[40-06] : ( -0.4839, 2.8388, -0.5919 )  -> 67.1297
[40-07] : ( 2.3322, 5.1709, 0.8473 )    -> 59.0384
[40-08] : ( 2.3229, 2.7406, -0.1692 )   -> 38.1748
[40-09] : ( 0.2869, 1.7730, 0.2811 )    -> 52.7391
[40-10] : ( 0.0426, 0.9711, 0.1633 )    -> 53.4421
--- Velocidades: [vector( -0.0670, -0.0488, 0.0391 ), vector( 0.7441, 0.6832, 0.9057 ),
vector( -1.4125, -1.4818, 1.2680 ), vector( 0.0334, 0.0779, -0.2221 ), vector( -0.2251,
-0.1386, 1.0374 ), vector( 0.7984, 0.1515, 0.4042 ), vector( 0.7545, 1.4538, 0.5782 ),
vector( -0.7483, -1.1923, -0.5161 ), vector( 2.0000, 2.0000, -2.0000 ), vector( 0.0290,
-0.3255, 0.5938 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
35.4188, 20.6034, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.0384, 2.8924, -0.3374 ),
vector( 0.5146, 1.7390, -0.5224 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 41
[41-01] : ( -0.0540, 1.7839, -0.4979 )  -> 28.4388
[41-02] : ( 1.2465, 2.5097, 0.7647 )    -> 56.4507
[41-03] : ( -2.7105, -1.6231, 2.7442 )  -> 56.6654
[41-04] : ( -0.6204, 0.2519, -0.3245 )  -> 43.9869
[41-05] : ( -0.2263, 0.9268, 0.8124 )   -> 56.9153
[41-06] : ( 0.4017, 2.7743, -0.1939 )   -> 34.9016
[41-07] : ( 2.1889, 5.0494, 0.7886 )    -> 57.4709
[41-08] : ( 0.6559, 1.0982, -0.8693 )   -> 43.2940
[41-09] : ( 2.2869, 3.7730, -1.7189 )   -> 177.2139
[41-10] : ( 0.0598, 0.8268, 0.5076 )    -> 57.5019
--- Velocidades: [vector( -0.0670, -0.0488, 0.0391 ), vector( 0.4408, 0.3130, 0.4803 ),
vector( -1.2980, -1.3413, 1.1368 ), vector( 1.7918, 1.8273, -0.7094 ), vector( -0.0771,
-0.0090, 0.7773 ), vector( 0.8855, -0.0644, 0.3980 ), vector( -0.1433, -0.1215, -0.0588
), vector( -1.6670, -1.6424, -0.7001 ), vector( 2.0000, 2.0000, -2.0000 ), vector(
0.0172, -0.1443, 0.3443 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 20.6034, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.5146, 1.7390, -0.5224 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 42
[42-01] : ( -0.1052, 1.7466, -0.4680 )  -> 19.6112
[42-02] : ( 1.2762, 2.3583, 0.6967 )    -> 53.4429
[42-03] : ( -3.3152, -2.2069, 3.2164 )  -> 57.9497
[42-04] : ( 1.3484, 2.2519, -1.0686 )   -> 105.6070
[42-05] : ( -0.1320, 1.0825, 1.1973 )   -> 50.7642
[42-06] : ( 1.2170, 2.5051, 0.1270 )    -> 43.4349
[42-07] : ( 1.3697, 3.6394, 0.2164 )    -> 54.7988
[42-08] : ( -1.1641, -0.4265, -1.5146 ) -> 67.7035
[42-09] : ( 3.7002, 5.1858, -3.2812 )   -> 205.2124
[42-10] : ( 0.0540, 1.0188, 0.3667 )    -> 55.1138
--- Velocidades: [vector( -0.0512, -0.0373, 0.0299 ), vector( 0.0297, -0.1515, -0.0680
), vector( -0.6047, -0.5838, 0.4722 ), vector( 1.9687, 2.0000, -0.7441 ), vector(
0.0943, 0.1557, 0.3849 ), vector( 0.8153, -0.2693, 0.3210 ), vector( -0.8191, -1.4100, -
0.5722 ), vector( -1.8199, -1.5247, -0.6453 ), vector( 1.4132, 1.4128, -1.5624 ),
vector( -0.0058, 0.1921, -0.1409 )]
```

```
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 20.6034, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.5146, 1.7390, -0.5224 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 43
[43-01] : ( -0.1115, 1.7421, -0.4643 )  -> 44.2703
[43-02] : ( 0.7239, 1.6395, -0.0951 )   -> 48.1550
[43-03] : ( -2.4364, -1.1712, 2.2685 )  -> 60.1585
[43-04] : ( 3.1370, 4.0644, -1.7229 )   -> 181.6294
[43-05] : ( 0.1222, 1.2936, 0.6980 )    -> 44.9156
[43-06] : ( 1.7340, 2.1268, 0.2857 )    -> 52.6881
[43-07] : ( 0.1520, 1.6394, -0.6545 )   -> 14.8000
[43-08] : ( -2.8531, -1.7230, -2.0608 ) -> 142.6671
[43-09] : ( 3.8225, 5.3089, -3.6857 )   -> 214.0217
[43-10] : ( 0.0372, 1.3711, -0.0070 )   -> 47.9440
--- Velocidades: [vector( -0.0063, -0.0046, 0.0037 ), vector( -0.5523, -0.7187, -0.7918
), vector( 0.8789, 1.0357, -0.9478 ), vector( 1.7886, 1.8125, -0.6542 ), vector( 0.2542,
0.2111, -0.4993 ), vector( 0.5170, -0.3783, 0.1586 ), vector( -1.2177, -2.0000, -0.8709
), vector( -1.6890, -1.2964, -0.5463 ), vector( 0.1224, 0.1231, -0.4045 ), vector( -
0.0168, 0.3522, -0.3737 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 44
[44-01] : ( -0.0917, 1.7565, -0.4759 )  -> 42.4645
[44-02] : ( 0.1031, 0.8346, -0.9794 )   -> 39.9412
[44-03] : ( -1.0398, 0.4156, 0.8449 )   -> 55.1259
[44-04] : ( 4.0608, 4.9873, -1.9969 )   -> 194.1890
[44-05] : ( 0.3970, 1.4928, -0.5685 )   -> 28.2461
[44-06] : ( 2.0773, 1.7901, 0.3725 )    -> 54.2850
[44-07] : ( -1.0912, -0.3606, -1.5220 ) -> 117.5728
[44-08] : ( -3.8134, -2.1700, -2.2428 ) -> 154.0153
[44-09] : ( 2.3395, 3.8517, -2.5562 )   -> 196.5985
[44-10] : ( 0.0147, 1.7605, -0.4919 )   -> 34.1100
--- Velocidades: [vector( 0.0197, 0.0144, -0.0115 ), vector( -0.6208, -0.8049, -0.8843
), vector( 1.3966, 1.5867, -1.4236 ), vector( 0.9238, 0.9229, -0.2741 ), vector( 0.2748,
0.1991, -1.2665 ), vector( 0.3433, -0.3368, 0.0868 ), vector( -1.2432, -2.0000, -0.8675
), vector( -0.9603, -0.4471, -0.1820 ), vector( -1.4830, -1.4572, 1.1295 ), vector( -
0.0225, 0.3894, -0.4848 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 45
[45-01] : ( -0.0562, 1.7823, -0.4966 )  -> 44.6041
[45-02] : ( -0.5313, 0.1661, -1.7950 )  -> 155.8889
[45-03] : ( 0.7411, 2.4113, -0.9318 )   -> 81.2722
[45-04] : ( 3.7383, 4.6218, -1.6939 )   -> 173.8085
[45-05] : ( 0.6152, 1.6271, -1.8896 )   -> 185.1272
[45-06] : ( 1.9310, 1.4688, 0.2349 )    -> 46.9637
[45-07] : ( -2.0278, -1.8556, -2.1685 ) -> 174.3724
[45-08] : ( -4.1338, -1.9841, -2.1549 ) -> 200.6452
```

```
[45-09] : ( 0.3395, 1.8517, -0.7140 )   -> 45.7934
[45-10] : ( -0.0082, 2.0843, -0.9737 )  -> 69.6068
--- Velocidades: [vector( 0.0355, 0.0259, -0.0208 ), vector( -0.6343, -0.6685, -0.8156
), vector( 1.7809, 1.9957, -1.7767 ), vector( -0.3225, -0.3655, 0.3030 ), vector(
0.2182, 0.1344, -1.3212 ), vector( -0.1464, -0.3212, -0.1376 ), vector( -0.9366, -
1.4951, -0.6465 ), vector( -0.3204, 0.1860, 0.0879 ), vector( -2.0000, -2.0000, 1.8422
), vector( -0.0230, 0.3238, -0.4819 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 46
[46-01] : ( -0.0066, 1.8185, -0.5256 )  -> 28.9653
[46-02] : ( -1.0538, -0.2967, -2.4182 ) -> 173.7934
[46-03] : ( 2.2083, 3.9673, -2.2725 )   -> 192.1724
[46-04] : ( 2.5170, 3.3290, -0.9826 )   -> 83.1074
[46-05] : ( 0.5882, 1.5566, -2.6131 )   -> 206.7549
[46-06] : ( 1.3039, 1.4934, -0.0515 )   -> 45.8163
[46-07] : ( -1.7874, -1.4266, -1.9751 ) -> 157.9709
[46-08] : ( -2.7610, -0.3290, -1.4442 ) -> 81.5683
[46-09] : ( -1.6605, -0.1483, 1.2289 )  -> 53.6299
[46-10] : ( -0.0239, 2.1903, -1.2828 )  -> 136.5381
--- Velocidades: [vector( 0.0496, 0.0362, -0.0290 ), vector( -0.5225, -0.4628, -0.6232
), vector( 1.4672, 1.5561, -1.3407 ), vector( -1.2212, -1.2928, 0.7113 ), vector( -
0.0270, -0.0706, -0.7235 ), vector( -0.6270, 0.0246, -0.2863 ), vector( 0.2404, 0.4291,
0.1934 ), vector( 1.3728, 1.6551, 0.7107 ), vector( -2.0000, -2.0000, 1.9429 ), vector(
-0.0157, 0.1060, -0.3091 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 47
[47-01] : ( 0.0479, 1.8581, -0.5574 )   -> 34.2811
[47-02] : ( -1.3608, -0.4574, -2.7228 ) -> 175.8778
[47-03] : ( 2.5572, 4.2377, -2.4521 )   -> 190.6797
[47-04] : ( 1.2104, 1.9491, -0.2372 )   -> 30.0881
[47-05] : ( 0.4138, 1.3637, -2.7077 )   -> 180.0119
[47-06] : ( 0.4095, 1.6837, -0.4263 )   -> 26.2776
[47-07] : ( -0.7123, 0.3304, -1.2082 )  -> 40.6705
[47-08] : ( -0.7610, 1.6710, -0.4696 )  -> 44.2181
[47-09] : ( -3.4079, -1.8512, 2.9082 )  -> 61.2346
[47-10] : ( -0.0218, 1.9486, -1.2006 )  -> 116.2908
--- Velocidades: [vector( 0.0544, 0.0396, -0.0318 ), vector( -0.3069, -0.1608, -0.3046
), vector( 0.3489, 0.2704, -0.1796 ), vector( -1.3067, -1.3800, 0.7454 ), vector( -
0.1744, -0.1928, -0.0946 ), vector( -0.8944, 0.1903, -0.3749 ), vector( 1.0751, 1.7570,
0.7669 ), vector( 2.0000, 2.0000, 0.9745 ), vector( -1.7474, -1.7029, 1.6793 ), vector(
0.0021, -0.2417, 0.0822 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 30.0881, 28.1542,
26.2776, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.2104,
1.9491, -0.2372 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4095, 1.6837, -0.4263 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 48
[48-01] : ( 0.0440, 1.8554, -0.5552 )   -> 20.4674
```

```
[48-02] : ( -1.0593, -0.0361, -2.2779 ) -> 200.3525
[48-03] : ( 2.8761, 4.6268, -2.8146 )   -> 196.8393
[48-04] : ( 0.5170, 1.3290, -0.0531 )   -> 46.2371
[48-05] : ( 0.4383, 1.4292, -2.8021 )   -> 211.6159
[48-06] : ( 0.3504, 1.6473, -0.4617 )   -> 43.0247
[48-07] : ( -0.8489, 0.1288, -1.2925 )  -> 57.7453
[48-08] : ( -0.7833, 1.6710, -0.5561 )  -> 27.5739
[48-09] : ( -3.3009, -1.7769, 2.8770 )  -> 62.7222
[48-10] : ( -0.0249, 1.9633, -1.2611 )  -> 141.9009
--- Velocidades: [vector( 0.0506, 0.0369, -0.0296 ), vector( -0.0055, 0.2606, 0.1403 ),
vector( 0.6678, 0.6595, -0.5421 ), vector( -2.0000, -2.0000, 0.9295 ), vector( -0.1499,
-0.1274, -0.1890 ), vector( -0.9535, 0.1539, -0.4102 ), vector( 0.9385, 1.5554, 0.6826
), vector( 1.9777, 2.0000, 0.8881 ), vector( -1.6404, -1.6286, 1.6481 ), vector( -
0.0010, -0.2270, 0.0217 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 49
[49-01] : ( 0.0832, 1.8840, -0.5781 )   -> 30.1430
[49-02] : ( -0.9708, 0.3286, -2.0136 )  -> 165.2742
[49-03] : ( 3.0827, 4.7550, -2.8764 )   -> 207.6065
[49-04] : ( -1.3460, -0.5263, 0.6625 )  -> 55.7104
[49-05] : ( 0.1418, 1.2027, -1.9206 )   -> 166.2299
[49-06] : ( -0.6116, 1.8702, -0.8609 )  -> 54.6090
[49-07] : ( 0.5145, 2.1288, -0.3347 )   -> 50.7142
[49-08] : ( 1.2167, 3.6574, 0.3165 )    -> 56.4148
[49-09] : ( -4.4414, -2.9043, 4.0830 )  -> 50.5738
[49-10] : ( -0.0117, 1.4901, -0.9328 )  -> 53.2691
--- Velocidades: [vector( 0.0392, 0.0286, -0.0229 ), vector( 0.0885, 0.3647, 0.2643 ),
vector( 0.2066, 0.1282, -0.0618 ), vector( -1.8629, -1.8553, 0.7156 ), vector( -0.2964,
-0.2265, 0.8815 ), vector( -0.9620, 0.2229, -0.3992 ), vector( 1.3634, 2.0000, 0.9579 ),
vector( 2.0000, 1.9864, 0.8725 ), vector( -1.1404, -1.1274, 1.2060 ), vector( 0.0132, -
0.4731, 0.3284 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000


--- Iteração: 50
[50-01] : ( 0.0917, 1.8902, -0.5830 )   -> 41.8220
[50-02] : ( -0.6690, 0.8241, -1.5224 )  -> 127.2116
[50-03] : ( 2.5340, 4.0138, -2.1524 )   -> 202.7982
[50-04] : ( -2.7987, -1.9631, 1.1083 )  -> 58.8430
[50-05] : ( -0.1506, 0.9834, -0.8214 )  -> 36.0883
[50-06] : ( -1.3919, 2.1941, -1.1538 )  -> 113.3544
[50-07] : ( 1.6443, 3.8447, 0.4457 )    -> 56.0081
[50-08] : ( 2.7168, 4.7995, 0.7983 )    -> 53.4076
[50-09] : ( -3.9394, -2.4253, 3.8285 )  -> 60.5012
[50-10] : ( 0.0065, 1.0514, -0.5174 )   -> 43.1852
--- Velocidades: [vector( 0.0085, 0.0061, -0.0049 ), vector( 0.3018, 0.4955, 0.4912 ),
vector( -0.5487, -0.7412, 0.7240 ), vector( -1.4528, -1.4368, 0.4458 ), vector( -0.2924,
-0.2193, 1.0992 ), vector( -0.7803, 0.3238, -0.2929 ), vector( 1.1298, 1.7159, 0.7804 ),
vector( 1.5000, 1.1421, 0.4818 ), vector( 0.5020, 0.4791, -0.2546 ), vector( 0.0182, -
0.4387, 0.4154 )]
--- Melhores resultados individuais: [12.4000, 35.0621, 39.5976, 41.3478, 28.1542,
34.9016, 14.8000, 13.6000, 37.9037, 18.8779]
--- Posições dos melhores resultados individuais: [vector( 0.0130, 1.8327, -0.5370 ),
vector( 0.0982, 0.8367, -0.9468 ), vector( 0.0000, 1.2001, 0.3393 ), vector( 1.3330,
2.0917, -0.3710 ), vector( 0.1666, 1.0968, -0.8378 ), vector( 0.4017, 2.7743, -0.1939 ),
vector( 0.1520, 1.6394, -0.6545 ), vector( 0.0000, 1.6027, -0.6344 ), vector( 0.5353,
1.8782, -0.1948 ), vector( 0.0136, 1.2966, -0.4306 )]
```

```
--- Código da Melhor Estratégia: 40-01
--- Melhor Estratégia: vector( 0.0130, 1.8327, -0.5370 )
--- Resultado da Melhor Estratégia: 12.4000
```