# BUSCA POR CARDUMES PARA PROBLEMAS DISCRETOS APLICADA AO PROBLEMA DE CARREGAMENTO DE PALETE

Trabalho de Conclusão de Curso

Engenharia da Computação

Aluno: Tarcísio Mendes de Farias

Orientador: Prof. Carmelo José Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

**Tarcísio Mendes de Farias**

# *Busca por Cardumes para Problemas Discretos Aplicada ao Problema de Carregamento de Palete*

Monografia apresentada para obtenção do grau de Bacharel em Engenharia da Computação pela Universidade de Pernambuco

Orientador:
Carmelo José Albanez Bastos Filho

ESCOLA POLITÉCNICA DE PERNAMBUCO
UNIVERSIDADE DE PERNAMBUCO

Recife - PE, Brasil

15 de maio de 2011

# *Resumo*

Problemas de busca em espaços de alta dimensionalidade e multimodais são difíceis de resolver. Metaheurísticas bioinspiradas tem sido desenvolvidas para solucionar esses problemas. Uma dessas metaheurísticas é o algoritmo de busca por cardumes (FSS, *Fish School Search*) que supera a otimização por enxame de partículas em alguns casos. Contudo, o FSS é uma técnica de inteligência de enxames que foi desenvolvido a princípio para resolver apenas problemas cujos espaços de busca são contínuos. Este trabalho propõe uma abordagem de busca por cardumes em ambientes discretos. Ademais, o operador de movimento de fuga inspirado no nado frenético dos peixes na presença de predadores é proposto neste trabalho. Esse novo operador melhora a capacidade de exploração em amplitude de forma significativa no espaço de busca discreto. Esta monografia introduz também uma técnica híbrida onde o FSS é um operador do algoritmo genético (GA, *Genetic Algorithm*), substituindo o operador de mutação que é essencialmente aleatório. Por fim, a busca por cardumes como operador do GA foi comparada com o GA por meio de simulações para resolver o problema de carregamento de palete (PLP, *Pallet Loading Problem*). Constatou-se que a técnica híbrida proposta obteve melhores resultados do que o GA no PLP. A técnica híbrida convergiu para a solução ótima em todas as simulações.

**Palavras-chave**: Busca por Cardumes, Inteligência de Enxames, Algoritmo Genético, Problema de Carregamento de Palete.

# *Abstract*

Search problems in high-dimensional and multimodal spaces are often difficult to solve. Many bio-inspired metaheuristics have been developed to solve these problems. One of these metaheuristics is the Fish School Search (FSS), which overcomes the Particle Swarm Optimization in some cases. However, the FSS is a swarm intelligence technique which was developed initially proposed to solve problems in continuous search spaces. This work proposes an approach to FSS in a discrete environment. Moreover, the escape movement operator inspired on swimming of the fishes in the presence of predators is proposed in this monograph. This new operator improves the capability of exploration significantly in the discrete search space. This work presents a hybrid technique where the FSS is introduced as an operator in the genetic algorithm (GA). In this hybrid algorithm, FSS replaces the GA mutation operator which is essentially random. Finally, the FSS as operator of the GA is compared with the GA using simulations to solve the pallet loading problem (PLP). Our proposal outperformed the GA in these trials and this algorithm converged to the optimal solution in all simulations for the PLP problem.



**Keywords**: Fish School Search, Swarm Intelligence, Genetic Algorithm, Pallet Loading Problem.

# *Agradecimentos*

A melhor forma de um pai expressar o seu amor para o seu filho é através da educação. Por isso, este trabalho é fruto desse amor dos meus pais: Bartolomeu Muniz de Farias e Elizabeth Mendes de Sousa Farias. Agradeço aos meus pais por terem me educado, ensinado-me os valores morais e éticos necessários para eu tomar minhas próprias decisões em sociedade. Sou grato aos meus irmãos Tássia Elizabeth e Tácito Mendes e a amiga e secretária Vânia pelo apoio que me deram para conclusão deste trabalho.

Durante a graduação tive a oportunidade de conhecer pessoas com visões de mundo diferentes, dentre elas se destacam os meus mentores e tutores da Universidade de Pernambuco que lhes digo meus sinceros agradecimentos: Carmelo Bastos (Poli), Carlos Alexandre (UFPE), Tiago Massoni (UFCG), Fernando Buarque (Poli), Raul Melo (FCM), Sérgio Campelo (Poli), Francisco Madeiro (Poli), Mêuser Valença (Poli), Sérgio Murilo (Poli), etc.

Agradeço ao professor Fernando Buarque em especial pelas orientações e ensinamentos que obtive através de iniciações científicas e atividades de extensão universitária. Agradeço também ao meu orientador Carmelo Bastos o incentivo, o apoio e as contribuições para elaboração deste trabalho.

Por fim, agradeço aos meus colegas da UPE, especialmente George Cavalcanti por ter me fornecido o código fonte do seu Trabalho de Conclusão de Curso (TCC). O problema abordado por George Cavalcanti em seu TCC foi usado neste trabalho como função de teste para as abordagens propostas.

# *Sumário*

# *Lista de Figuras*

# *Lista de Tabelas*

# *Lista de Algoritmos*

# *Lista de Símbolos e Siglas*

ABC *Artificial Bee Colony* (Colônia Artificial de Abelhas)

ACO *Ant Colony Optimization* (Otimização por Colônia de Formigas)

FSS *Fish School Search* (Busca por Cardumes)

GA *Genetic Algorithm* (Algoritmo Genético)

GAFSS *Genetic Algorithm with Fish School Search* (Algoritmo Genético com Busca por Cardumes)

gFSS *Greedy Fish School Search* (Busca por Cardumes Gulosa)

NP *Non-Deterministic Polynomial Time* (Tempo Polinomial Não-determinístico)

PLP *Pallet Loading Problem* (Problema de Carregamento de Palete)

PSO *Particle Swarm Optimization* (Otimização por Enxame de Partículas)

SIP Sistema Inteligente de Carregamento de Paletes

# 1 Introdução

Espaços de buscas de alta dimensionalidade se tornam computacionalmente caros de serem explorados quando se utiliza técnicas tradicionais. Por isso, heurísticas [1] são desenvolvidas. Heurísticas são algoritmos que buscam boas soluções a um custo computacional aceitável sem a garantia de que a solução ótima sempre será encontrada, ou seja, fornecem soluções sem um limite formal de qualidade. A classe metaheurística é um tipo de heurística que é aplicável a vários tipos de problemas que não são conhecidos algoritmos eficientes para resolvê-los.

Metaheurísticas inspiradas no comportamento coletivo de alguns animais surgiram como alternativa para resolver problemas complexos (e.g.: problemas NP-completos [2]) de forma mais eficiente e eficaz quando comparadas com as técnicas tradicionais [3]. Nas últimas décadas, surgiram várias técnicas bioinspiradas, como por exemplo: otimização por exame de partículas (PSO, *Particle Swarm Optimization* [4]) inspirada no comportamento social de bando de pássaros na busca por comida ou ninho, otimização por colônia de formigas (ACO, *Ant Colony Optimization* [5]) inspirada no comportamento das formigas na busca por comida que depositam ferormônio no ambiente para estabelecer rotas em direção aos alimentos encontrados, colônia de abelhas artificiais (ABC, *Artificial Bee Colony* [6]) inspirada na inteligência coletiva emergente das abelhas no processo de busca por alimentos, entre outras.

Muitos algoritmos inspirados na natureza são baseados em população tais como o algoritmo genético (GA, *Genetic Algorithm* [7] [8]), PSO e busca por cardumes (FSS, *Fish School Search*) [9] . Técnicas baseadas em populações permitem o paralelismo do processo de busca, conduzindo a uma convergência mais rápida. Por outro lado, existe um aumento do custo de comunicação entre as entidades da população. Além disso, os indivíduos possuem alguma habilidade de memorização das experiências passadas.

Efeitos de agregação ocorrem tanto em pequenos organismos, como as bactérias, quanto em grandes como os peixes e as baleias. Mais da metade das espécies de peixes vivem em cardumes, isto é, mostram movimentos síncronos e coordenados, em algum momento de suas vidas e outras espécies de peixes se agregam de forma grosseira [10]. A agregação de seres

simples com inteligência individual limitada, como a de muitas espécies de peixes, faz emergir uma inteligência coletiva.

A técnica FSS proposta por Bastos Filho *et al* [11] é uma metaheurística inspirada no comportamento emergente de cardumes [10] que está relacionada com o aumento da habilidade de sobrevivência dos peixes, como por exemplo, proteção mútua e obtenção de comida. Nessa técnica, cada peixe é uma metáfora para uma possível solução.

O processo de busca ocorre através da combinação de operadores que representam a movimentação do peixe e sua alimentação que modifica o peso do peixe [9]. Vale salientar que o peso do peixe representa a sua memória, ou seja, o sucesso da busca pela solução ótima ao longo das iterações. A movimentação dos peixes é realizada por três operadores de movimento, são eles: individual, coletivo-instintivo e coletivo-volitivo.

Ademais, o FSS foi elaborado para espaço de busca contínuo. Contudo, há também problemas discretos complexos, por exemplo: maximizar a quantidades de caixas dispostas no interior de um contêiner, e consequentemente, minimizar o custo do transporte. Outro problema de logística é o problema de carregamento de paletes (PLP, *Pallet Loading Problem* [12]) que se trata da otimização da quantidade de caixas dispostas sobre um palete [13]. O palete (ver figura 1.1) retirada de [14]) é um estrado de madeira, metal ou plástico que é utilizado para movimentação de cargas por meio de empilhadeiras. Vários padrões de paletes são fabricados, ou seja, com diferentes dimensões. O PLP na visão do produtor [15] é mais simples, pois os produtos são empacotados em caixas de iguais dimensões que são arranjadas sobre o palete, de tal forma que haja um maior número de caixas dado uma altura máxima aceitável de empilhamento.

Figura 1.1: Ilustração de palete de madeira à esquerda e de plástico à direita.

Adicionalmente, ambientes discretos possuem um conjunto finito de estados, percepções e ações [16]. O algoritmo FSS possui características de natureza discreta que implicam um maior potencial para essa técnica ser aplicada em ambientes discretos com sucesso, ao contrario do PSO que não detêm essas características.

Além disso, algumas técnicas de inteligência computacional fazem uso de abordagens pseudo-aleatórias para gerar diversidade durante a busca, por exemplo, o operador de mutação do GA [7]. Isso possivelmente é ruim, pois faz com que a busca se torne cada vez mais aleatória. A elaboração do FSS discreto permitirá uma hibridização dessas duas heurísticas através da substituição do operador de mutação por uma metaheurística inteligente.

Poucos trabalhos foram publicados a respeito do FSS, por esta ser uma técnica recente. Em geral, alguns pesquisadores propuseram modificações dos operadores e adição de novos com o intuito de melhorar o desempenho do algoritmo [17] e encontrar múltiplos ótimos globais durante a busca em ambientes contínuos. Entretanto, ainda não foram propostas abordagens discretas para o FSS. Por isso, este trabalho tem como objetivo a proposição de uma nova abordagem do FSS para buscar soluções ótimas em ambientes discretos. É proposto também, um novo operador chamado de movimento de fuga para melhorar o processo de busca do FSS em ambientes discretos. Esse operador é inspirado no comportamento dos peixes devido à presença de predadores, como por exemplo, tubarões.

Ademais, uma heurística híbrida é proposta, combinando as técnicas GA e FSS. Outras metas principais deste trabalho propõem resolver o PLP do produtor com algumas simplificações; analisar o desempenho do FSS para problemas discretos e do GA híbrido na solução do PLP do produtor; e por fim, desenvolver uma ferramenta com a abordagem híbrida proposta e interface gráfica simples para exibir ao usuário de que forma dispor as caixas com o intuito de otimizar a área ocupada do palete.

## 1.1 Estrutura da monografia

O capítulo 2 aborda os conceitos fundamentais e algoritmos escolhidos de inteligência computacional para o estudo de caso proposto nesta monografia. As abordagens PSO, FSS e GA são discutidas. O algoritmo genético, seus operadores e um pseudocódigo são descritos. Algumas definições para inteligência de enxames e a técnica PSO são apresentadas. Por fim, ainda no capítulo 2, a metaheurítica FSS é discutida em detalhes, pois é o foco deste trabalho.

O capítulo 3 apresenta a contribuição deste trabalho. A técnica de busca por cardumes para ambientes discretos é explicada juntamente com o operador de fuga que é proposto. As modificações dos operadores são mostradas. Além disso, a hibridização do FSS e do GA é detalhada.

O capítulo 4 mostra a modelagem adotada e a ferramenta desenvolvida para solucionar o problema multimodal PLP. O capítulo 4 também contém os experimentos realizados e os resul-

tados obtidos com a ferramenta desenvolvida. Uma análise paramétrica simples das técnicas propostas também é discutida. Diversos experimentos foram realizados com o objetivo de avaliar o desempenho entre as abordagens propostas para solucionar o PLP do produtor.

O capítulo 5 estabelece a conclusão deste trabalho, detalha as principais contribuições, elege a melhor abordagem dentre as propostas para solucionar o PLP e divulga possíveis linhas de pesquisa a serem desenvolvidas em trabalhos futuros.

# 2 *Computação Bioinspirada*

Neste capítulo são abordados os conceitos fundamentais e algoritmos escolhidos de inteligência computacional para o estudo de caso proposto nesta monografia. As abordagens *Particle Swarm Optimization* (PSO) [18], *Fish School Search* (FSS) [19] e *Genetic Algorithm* (GA) são discutidas. A seção 2.4 descreve o algoritmo genético, seus operadores e um pseudocódigo. Na seção 2.1, são apresentadas algumas definições para inteligência de enxames. Na seção 2.2, a técnica PSO é detalhada. Na seção 2.3, a metaheurística FSS é discutida em detalhes, pois é o foco deste trabalho.

## 2.1 Inteligência de Enxames

O comportamento emergente das interações entre entidades simples de um enxame caracteriza a inteligência coletiva, ou seja, a inteligência de enxames. O conhecimento local de cada entidade, por exemplo, a sua posição no espaço de busca compartilhado com as outras entidades. Dessa maneira, formam uma rede de conhecimento da qual emerge uma visão global da função objetivo. Esse comportamento permite que problemas complexos sejam resolvidos pelas entidades, a partir de ações individuais regidas por regras simples.

Bonabeau *et al.* [20] define inteligência de enxames como uma tentativa de elaborar algoritmos para resolução de problemas inspirados no comportamento coletivo de insetos e outros animais sociáveis.

A modelagem da comunicação entre as entidades artificiais e a interação com o espaço de busca é inspirada no comportamento observado desses indivíduos na natureza. Por exemplo, algoritmos baseados no comportamento de abelhas em colmeias utilizam a dança e o mecanismo de divisão de trabalhos entre elas como metáfora para comportamentos presentes em uma colmeia artificial.

## 2.2 Otimização por Enxame de Partículas

Otimização por enxame de partículas (PSO) é uma metaheurística proposta por James Kennedy e Russel Eberhart em 1995 [4], e inspirada no comportamento social dos pássaros estudados por Heppner [21]. A busca pelo alimento ou pelo ninho e a interação entre os pássaros ao longo do voo são modelados como um mecanismo de otimização. O PSO é uma técnica de otimização para funções contínuas e de alta dimensionalidade.

Ademais, o PSO é bio-inspirado com as seguintes analogias: o termo partícula foi adotado para simbolizar os pássaros que é uma metáfora para as possíveis soluções do problema de otimização; a área sobrevoada pelos pássaros equivalente ao espaço de busca; encontrar o local com comida, ou o ninho, corresponde a encontrar a solução ótima. A função de avaliação *fitness*, a qual avalia o desempenho das partículas, é utilizada para que o bando de pássaros sempre se aproxime do objetivo. Os pássaros fazem uso de suas experiências e da experiência do próprio bando a fim de alcançar o alvo focado, sejam os alimentos ou os ninhos.

Para cada partícula, há o termo cognitivo ($\vec{p}_{best}$) e o social ($\vec{g}_{best}$) que são valores que representam um posição no espaço de busca. O primeiro exerce uma influência sobre o movimento da partícula na direção e sentido da melhor posição encontrada pela partícula até o momento. O segundo atrai todas as partículas para a melhor posição já visitada pelo enxame. Adicionalmente, a cada partícula é associada a uma velocidade, que lhe permite se movimentar pelo espaço de busca. Essa velocidade é acelerada na direção de seu $\vec{p}_{best}$ e do $\vec{g}_{best}$.

A interação social entre as partículas é fundamental para o bom funcionamento do PSO. As partículas mais aptas influenciam as menos aptas em sua vizinhança. Isso, implica uma convergência mais rápida das partículas nessa vizinhança para uma solução comum.

As partículas que formam o enxame se movimentam pelo espaço de busca à procura da solução ótima. Cada partícula $i$ possui uma posição e uma velocidade. O processo de atualização de uma partícula possui dois termos: o termo cognitivo e o termo social. A cada ciclo, são atualizadas a velocidade e a posição (nessa ordem) de cada partícula do enxame. O vetor velocidade e posição são descritos, respectivamente, em (2.1) e (2.2) [22].

$$\vec{v}_i(t+1) = \omega\vec{v}_i + c_1 r_1(t)(p_{lbest_i}(t) - \vec{x}_i) + c_2 r_2(t)(g_{best}(t) - \vec{x}_i), \tag{2.1}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1), \tag{2.2}$$

onde $\vec{v}_i(t)$ é o vetor velocidade da partícula $i$, $\vec{x}_i(t)$ o vetor posição atual da partícula, $\omega$ o valor

da inércia, $r_1$ e $r_2$ são números aleatórios gerados a partir de uma distribuição uniforme no intervalo [0,1] e $c_1$ e $c_2$ são os fatores de aceleração social e cognitivo, respectivamente.

Todas as partículas no espaço de busca tem a possibilidade de descobrir um ótimo global e compartilhar com as demais. Os coeficientes aleatórios dão estocacidade à busca. O fator de inércia controla o grau de exploração da busca. Quanto maior for o fator de inércia haverá mais exploração em amplitude. É aconselhável inicializar $\omega$ com valores próximos a "1,0" e decrescer esse valor ao longo do tempo a fim de realizar uma busca em profundidade (*exploitation*) na melhor região encontrada pelo enxame durante a exploração em amplitude (*exploration*) [22].

As partículas no PSO interagem entre si, assim como animais que vivem em comunidade na natureza e aprendem umas com as outras. Elas tendem a ser influenciadas mais por outras da vizinhança que encontraram as melhores soluções [3]. A interação é determinada pela topologia de comunicação do algoritmo. As topologias estrela e anel são dois exemplos clássicos.

Na topologia em anel também conhecida como *Local Best PSO* ($L_{best}$), uma partícula se relaciona com os seus $n_i$ vizinhos imediatos e atualiza sua posição de acordo com a solução mais apta encontrada na vizinhança dessa partícula. A melhor posição encontrada pelo enxame é propagada para todas as partículas através das sobreposições de vizinhanças. Quanto maior for $n_i$, mais rápido é o processo de convergência. Contudo, o enxame fica mais susceptível a ficar preso em ótimos locais, prejudicando a precisão do algoritmo. Esse tipo de vizinhança é recomendado quando se quer encontrar o ponto de ótimo global de funções multimodais, haja vista, uma maior cobertura do espaço de busca (maior exploração) quando comparados a maiores valores de $n_i$.

A topologia em estrela, isto é, *Global Best PSO* ($G_{best}$) é um caso particular da topologia em anel, onde $n_i$ igual ao número de partículas menos um (a própria partícula). Nessa abordagem, as partículas são guiadas pela solução mais apta de todo enxame ($\vec{g}_{best}$). Todas as partículas conhecem a posição das outras e ajustam suas posições baseando-se na melhor solução já encontrada pelo enxame. Essa abordagem é recomendável para otimização de funções unimodais por motivos já mencionados anteriormente.

O conceito de PSO para a sua versão local consiste em, a cada instante de tempo, mudar a velocidade de cada partícula por meio de suas posições $\vec{p}_{best}$ e $\vec{l}_{best}$. Já para versão global de PSO, o ajuste ocorre de acordo com as posições $\vec{p}_{best}$ e $\vec{g}_{best}$.

## 2.3 Busca baseada em Cardumes

Nesta seção, é descrito o algoritmo de busca por cardumes (FSS, *Fish School Search*) proposto por Bastos Filho *et al* [11]. Os conceitos básicos são discutidos e, em seguida, o modelo matemático é apresentado. Ademais, o pseudocódigo do algoritmo é mostrado no final desta seção.

### 2.3.1 Introdução

O FSS é uma metaheurística inspirada no comportamento emergente de cardumes que está relacionado com o aumento da habilidade de sobrevivência dos peixes, como por exemplo, proteção mútua e busca por comida. Cada peixe representa uma possível solução para o problema. A principal característica deste paradigma é que os peixes contêm uma memória inata limitada de seu sucesso, isto é, seus pesos. Outro atributo importante do FSS é a ideia de evolução por meio de uma combinação de operadores que representam o nado coletivo e individual durante o processo de busca. O FSS é uma boa opção para busca em espaços de alta dimensionalidade e multimodais [11].

Dois grupos compreendem os principais operadores do algoritmo FSS:

- Alimentação: comida é uma metáfora para indicar aos peixes as regiões do aquário que são provavelmente bons locais do espaço de busca.
- Natação: na realidade uma coleção de operadores que são responsáveis por guiar o esforço da busca globalmente em direção a subespaços do aquário os quais influenciam todos os indivíduos (peixes) por serem mais promissores em relação ao processo de busca.

### 2.3.2 Operador de Alimentação

Com o intuito de encontrar mais comida, os peixes no cardume podem desempenhar movimentos independentes. Como resultado, cada peixe pode aumentar ou diminuir de peso, dependendo do seu sucesso ou falha na busca por comida. O peso atual do peixe é dado por (2.3).

$$W_i(t+1) = W_i(t) + \frac{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]}{\forall i, max\{|f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]|\}}, \tag{2.3}$$

onde $W_i(t)$ é o peso do peixe $i$, $\vec{x}_i(t)$ é a posição do peixe $i$ e $f[\vec{x}_i](t)$ avalia a função *fitness*, isto é, a quantidade de comida, em $\vec{x}_i(t)$.

A fim de que a convergência em direção as áreas mais promissoras do aquário ocorra rapidamente algumas características foram observadas [9]:

- O parâmetro do peso foi criado ($W_{scale}$) para limitar o peso do peixe no intervalo [1,$W_{scale}$];
- O peso inicial de todos os peixes é igual a $\frac{W_{scale}}{2}$;
- A varição do peso do peixe é avaliada uma vez em todo ciclo do FSS.

### 2.3.3 Operadores de Natação

No FSS, os padrões de natação de um cardume são resultados de três diferentes movimentos que são classificados em três classes [9]: (i) individual, (ii) coletivo instintivo e (iii) coletivo volitivo. Além disso, o nado dos peixes está diretamente relacionado a todo importante comportamento coletivo e individual, tal como comer, procriar, escapar de predadores, ir para regiões mais habitáveis.

#### (i) Movimento individual

O movimento individual é aplicada a cada peixe no aquário em todas as iterações do algoritmo FSS. Neste movimento, é definido o parâmetro $step_{ind}$ que representa o deslocamento do peixe no aquário. O parâmetro $step_{ind}$ é multiplicado por um número gerado aleatoriamente por uma distribuição uniforme no intervalo $[-1,1]$ com o intuito de atribuir aleatoriedade no processo de busca (ver equação 2.4). O peixe se desloca de $step_{ind}.rand[-1,1]$ caso esteja dentro das fronteiras do aquário (espaço de busca). Adicionalmente, o deslocamento do peixe no espaço de busca de um ponto $A$ para um ponto $B$, somente ocorre caso $B$ possua mais comida do que em $A$, ou seja, apenas se a função de avaliação (*fitness*) indicar que $B$ é uma região mais promissora do que $A$. Vale ressaltar que a comida é uma metáfora para avaliação das soluções candidatas no processo de busca.

$$\Delta\vec{x}_{ind_i} = step_{ind}.rand[-1,1], \tag{2.4}$$

onde $\Delta\vec{x}_{ind_i}$ é o deslocamento do peixe $i$ em determinado ciclo.

Ademais, com intuito de promover habilidades de busca em profundidade o passo ($step_{ind}$) decai linearmente nas próximas iterações do algoritmo. Uma alternativa para diminuição do passo com o decorrer das iterações é apresentado em (2.5).

$$step_{ind}(t+1) = step_{ind}(t) - \frac{(step_{ind_i} - step_{ind_f})}{iteracoes}, \tag{2.5}$$

onde *iteracoes* é o número de iterações usado na simulação. $step_{ind_i}$ e $step_{ind_f}$ são os parâmetros do movimento individual no início e no final da execução, respectivamente.

**(ii) Movimento Coletivo-instintivo**

Os peixes mais bem sucedidos em seus movimentos individuais influenciam no resultado da direção do movimento coletivo mais do que os outros. Quando a direção global é computada, cada peixe é reposicionado (ver equação 2.6). Este movimento é baseado na variação da quantidade de comida, ou seja, avaliação da função *fitness*, encontrada pelo peixe. O deslocamento é uma média ponderada dos movimentos individuais de cada peixe, onde essas variações são os pesos (ver equação 2.7).

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{I}(t), \tag{2.6}$$

onde $\vec{I}(t)$ a direção resultante que é avaliada por meio de (2.7) que é dada por:

$$\vec{I} = \frac{\sum_{i=1}^{N} \Delta\vec{x}_{ind_i} \cdot \{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}}{\sum_{i=1}^{N} \{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}}, \tag{2.7}$$

onde $\Delta\vec{x}_{ind_i}$ é o deslocamento do peixe $i$ devido ao movimento individual no ciclo do FSS.

**(iii) Movimento Coletivo-volitivo**

O cardume desempenha outro ajuste de suas posições no espaço de busca após os movimentos individual e coletivo-instintivo que é chamado de movimento coletivo-volitivo. Este movimento é baseado no desempenho global do cardume. A ideia é simples: se o cardume aumenta de peso, significa que a busca foi bem sucedida, então o raio do cardume deve contrair; senão, ele deve aumentar. Este operador regula a capacidade de exploração do algoritmo [11][19].

A dilatação ou contração do cardume é aplicada como um pequeno ajuste na posição de todos os peixes de/para o baricentro do cardume que é obtido como mostrado em (2.8). O baricentro é uma média ponderada das posições dos peixes cujos pesos são os pesos ($W_i$) dos peixes.

$$\overrightarrow{Bari}(t) = \frac{\sum_{i=1}^{N} \vec{x}_i(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)}. \tag{2.8}$$

O parâmetro passo volitivo ($step_{vol}$) é definido para especificar a intensidade do deslocamento do peixe de/para o baricentro. A nova posição é determinada como em (2.9), se o peso global do cardume aumenta em relação ao peso dele no ciclo anterior do FSS; ou como em (2.10), se o peso global diminui.

$$\vec{x}_i(t+1) = \vec{x}_i(t) - step_{vol}.rand.[\vec{x}_i(t) - \overrightarrow{Bari}(t)], \tag{2.9}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + step_{vol}.rand.[\vec{x}_i(t) - \overrightarrow{Bari}(t)], \tag{2.10}$$

onde *rand* um número aleatório gerado de uma distribuição uniforme no intervalo $[0,1]$ e o valor do $step_{vol}$ deve ser duas vezes o valor do $step_{ind}$ [19].

### 2.3.4 Pseudocódigo do FSS

O pseudocódigo do algoritmo FSS, apresentado no Algoritmo 1, começa gerando aleatoriamente um cardume de acordo com restrições que controlam o tamanho dos peixes e suas posições iniciais. O processo de busca ocorre até que pelo menos uma condição de parada seja encontrada. Alguns critérios de parada para o FSS são os seguintes: limite de tempo, máximo raio do cardume, limite de iterações (ciclos), mínimo peso do cardume e número máximo de peixes [9].

## 2.4 Algoritmo Genético

A ideia de aplicar os princípios de Darwin para automatizar a solução de problemas surgiu nos anos quarenta, muito antes dos avanços dos computadores. Em 1948, Turing propôs uma busca evolucionária ou genética [23][24], e Bremermann em 1962 [25] realizou de fato experimentos de otimização por meio de computação evolutiva. O algoritmo genético é inspirado na teoria da evolução de Darwin [26] que oferece um explanação da diversidade biológica e seus mecanismos subjacentes, como por exemplo, a seleção natural.

O GA foi inicialmente proposto por Holland [27] como um estudo do comportamento adaptativo. Este tem uma representação binária, seleção proporcionada pela aptidão (*fitness*), uma baixa probabilidade de mutação, e um ênfase na recombinação inspirada geneticamente como

**Algoritmo 1**: Pseudocódigo do Algoritmo de Busca por Cardumes

```
Inicialize todos os peixes aleatoriamente;
enquanto critério de parada faça
    para cada peixe no cardume faça
        avalie a função fitness;
        execute o operador de movimento individual (2.4);
        execute o operador de alimentação (2.3);
    calcule o valor de I(t) por meio de (2.7);
    para cada peixe no cardume faça
        execute o operador de movimento coletivo-instintivo (2.6);
    calcule o valor de Bari através de (2.8);
    para cada peixe no cardume faça
        se pesoCardume(t+1) > pesoCardume(t) então
            atualize a posição do peixe usando (2.9);
        senão
            se pesoCardume(t+1) < pesoCardume(t) então
                atualize a posição do peixe usando (2.10);
    atualize a variável step usando (2.5).
```

um meio de gerar novas soluções candidatas. No GA, existe uma população de indivíduos que são chamados de cromossomos. O cromossomo é uma metáfora para uma possível solução de um problema. Em geral, essa população é gerada inicialmente de forma aleatória, o que permite uma melhor exploração de um espaço de busca totalmente desconhecido.

O conceito de evolução é aplicado à população por meio de critérios de avaliação e seleção de indivíduos que evoluem com o passar das iterações do algoritmo. Nessa técnica, a evolução é mapeada através de critérios de avaliação e seleção de indivíduos. A avaliação de cada indivíduo é feita por meio do cálculo da função de aptidão (função *fitness*) que indica o quão boa é uma solução para o problema a ser solucionado. Vale salientar que a avaliação dos indivíduos dependem do tipo de problema podendo variar. A seleção dos indivíduos mais aptos é realizada de maneira determinística ou probabilística. Portanto, a tendência é que haja uma convergência para uma boa solução devido à sobrevivência dos indivíduos (soluções candidatas) mais aptos ao longo das gerações.

O GA pode ser aplicado à problemas com condições que não possuem representação formal; problemas com diferentes entradas que são combinadas para buscar uma solução ótima; problemas de alta dimensionalidade, ou seja, com um grande número de entradas. Os operadores do GA não se limitam a apenas o operador de seleção e avaliação, é necessário que também haja operadores que garantam a diversidade na busca. Caso não houvesse tais operado-

res a busca seria condicionada as soluções candidatas iniciais. Portanto, a existência de outros operadores é imprescindível, são eles: operador de recombinação (cruzamento) e operador de mutação. Esses operadores são responsáveis por aumentar a capacidade de exploração desta técnica.

Uma solução para o problema é codificada através de um cromossomo que possui tamanho finito. Segundo Serrada em 2000 [28]: "os GAs não trabalham sobre o domínio do problema, mas sim sobre representações de seus elementos". O cromossomo é uma sequência de valores representados por números binários, inteiros, reais, símbolos ou outro tipo de dado usado em um problema. Além disso, as várias entradas (genes de um cromossomo) de um problema podem ser combinadas. Vale ressaltar que toda representação por parte do algoritmo é baseada em seu genótipo (conjunto de genes que um indivíduo possui) e toda a avaliação é baseada em seu fenótipo (resultado do processo de decodificação dos genes).

A solução representada por um indivíduo possui sua qualidade mensurada por meio da função de aptidão (ou fitness). Durante a seleção, os indivíduos mais aptos são mais prováveis de serem escolhidos e combinarem-se para gerar novas soluções do que os menos aptos. Adicionalmente, uma abordagem do GA é o uso de elitismo [29], onde o operador mutação é somente aplicado a uma parte dos indivíduos (os menos aptos) para não perder as melhores soluções ao longo das iterações.

O operador de seleção conduz as soluções para as melhores regiões do espaço de busca, responsável pela busca em profundidade. A seleção é um processo também cumulativo, pois os benefícios da seleção permanecem de uma geração para outra. Adicionalmente, os indivíduos são escolhidos para posterior cruzamento por meio desse operador. Um método de seleção é o de exclusão social [13], onde os indivíduos da população são ordenados de acordo com o seu valor de aptidão, formando uma lista de indivíduos ordenada do mais apto ao menos apto. Uma alternativa utilizada no método de exclusão social é a seguinte: o primeiro indivíduo dessa lista cruza com o segundo, o terceiro cruza com o quarto e assim por diante. Existem outros métodos de seleção, como por exemplo, seleção por giro de roleta [30], seleção por torneio [31], seleção estocástica remanescente *remainder stochastic selection* [31] que não serão detalhados por não fazerem parte do escopo deste trabalho.

O operador de recombinação é responsável pela busca em amplitude, ou seja, explorar o espaço de busca. No cruzamento, dois cromossomos (pais) são combinados para gerar "1" ou mais filhos. Alguns exemplos de recombinação são: cruzamento de ponto único, $n$-pontos e uniforme [7]. A figura 2.1 mostra como é feito um cruzamento de 2-pontos, onde os dois pontos são escolhidos aleatoriamente. Os retângulos preenchidos podem ser interpretados como o valor

lógico verdadeiro e os em branco como falso. Cada retângulo representa um número, letra ou qualquer outro tipo de dado que compõe o cromossomo. Para gerar o Filho 1 basta aplicar os operadores lógicos *AND* e *OR* da seguinte forma: [(Pai 1) *AND* (Máscara B)] *OR* [(Pai 2) *AND* (Máscara A)]. O Filho 2 é obtido de forma análoga.

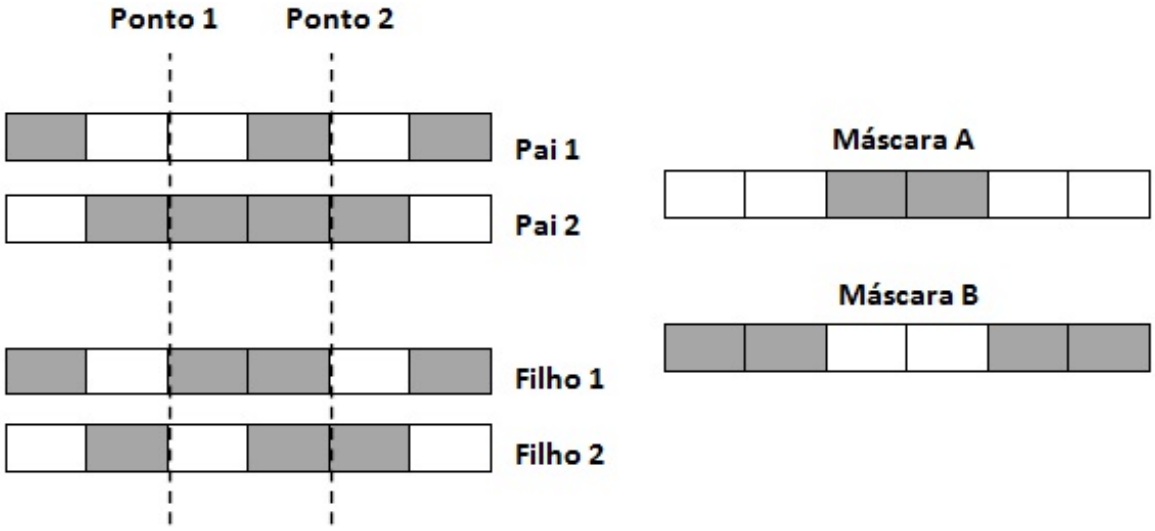


Figura 2.1: Cruzamento de 2-pontos.

O operador de mutação cria diversidade na busca após o cruzamento escolhendo de acordo com um probabilidade pré-determinada, os indivíduos que sofrerão alterações em sua estrutura de forma aleatória. Além disso, a mutação pode ser aplicada apenas a uma parte do indivíduo em vez de alterar aleatoriamente todos os elementos do cromossomo. Exemplos de operadores de mutação são: mutação *flip*, onde cada gene a ser mutado recebe um valor válido; mutação por troca, onde $n$ pares de genes são escolhidos aleatoriamente, e em seguida, trocam de lugar no cromossomo; mutação *creep* um valor aleatório é somado do valor do gene.

Uma análise paramétrica é indispensável para aplicação do GA em diferentes problemas. Os parâmetros do GA influenciam no comportamento do algoritmo, é preciso realizar uma análise para descobrir quais são os valores mais adequados para eles em determinados problemas. Os principais parâmetros do GA são:

- Quantidade de indivíduos, onde uma população pequena implica possível baixa diversidade, e consequente, convergência prematura e uma população grande aumenta o custo computacional;

- Taxa de cruzamento que indica a quantidade de indivíduos da população que se recombinarão. Uma taxa alta permite que boa parte da população seja substituída, podendo ser retirados indivíduos bastante aptos, já uma taxa baixa fará com que o algoritmo demore a convergir, devido poucos indivíduos novos serem gerados;

- Taxa de mutação que indica o número de indivíduos que sofrerão mutação. Uma taxa mutação alta torna a busca essencialmente aleatória. Previne a estagnação da solução em ótimos locais.

O pseudocódigo de versão simples do GA é apresentado em (Algoritmo 2).

**Algoritmo 2**: Pseudocódigo de um Algoritmo Genético Simples

```
1 Gere uma população inicial;
2 Avalie de acordo com a função fitness cada indivíduo;
3 enquanto critério de parada não é alcançado faça
4     Selecione os indivíduos mais aptos da geração anterior para o cruzamento;
5     Recombine com certa probabilidade os indivíduos obtendo os descendentes;
6     Mute os descendentes com certa probabilidade;
7     Calcule o valor de aptidão (fitness) dos descendentes;
8     Insira os novos descendentes na população;
9 Apresente o indivíduo mais apto como solução.
```

# 3 *Busca por cardumes em ambientes discretos*

Este capítulo tem como objetivo apresentar uma nova abordagem para busca em ambientes discreto onde o espaço de busca é finito. A discretização dos operadores do algoritmo FSS é detalhada na seção 3.1. Um novo operador de movimento para o FSS é proposto em 3.1.4. Os parâmetros e o pseudocódigo da busca por cardumes para problemas discretos é mostrada em 3.2.

## 3.1 Versão discreta dos operadores do FSS original

Os princípios computacionais do FSS são mantidos nesta abordagem discreta. Alguns desses princípios são: o processo de busca baseado numa população de indivíduos; peixes com memória inata limitada (peso dos peixes) que representa o sucesso na busca; e movimentação dos peixes no espaço de busca. No entanto, os operadores de movimento do FSS precisam ser mapeados para o universo dos números inteiros. Eles são encarregados de alterar a posição do peixe no espaço de busca discreto. O operador de alimentação o qual foi visto em 2.3.2 permanece inalterado, pois apenas é responsável pelo peso do peixe que pode assumir qualquer valor real entre "1" e $W_{scale}$. Embora o peso do peixe faça parte do cálculo do baricentro (ver equação 2.8) e seja um número real, o operador de movimento coletivo-volitivo que será proposto em (3.1.3) mapeia os resultados para o universo dos inteiros. As modificações nas equações dos operadores de movimento são explicadas e mostradas nas subseções 3.1.1, 3.1.2 e 3.1.3.

### 3.1.1 Operador de Movimento Individual

O princípio deste movimento em ambiente discreto é o mesmo que o do FSS original contínuo, o que muda é a forma de executá-lo. Por exemplo, continua sendo verdade que o peixe somente se desloca de uma posição A para uma posição B, caso B possua mais comida que A. Senão, o peixe permanece em A. A intensidade e sentido do deslocamento do peixe por meio

deste operador são definidos pela equação (3.1). O peixe se desloca por si só de $\Delta\vec{x}_{ind_i}$ a cada ciclo do algoritmo da versão discreta do FSS. Este movimento é essencialmente discreto mesmo para o FSS original, pois a ideia de passo ($step_{ind}$) remete a um espaço de busca enumerável.

$$\Delta\vec{x}_{ind_i} = step_{ind}.randInt[-1,1], \tag{3.1}$$

onde $step_{ind}$ é um número inteiro maior ou igual a "1", $randInt[-1,1]$ é um número inteiro gerado aleatoriamente no intervalo $[-1,1]$, ou seja, assume o valor "$-1$", "0" ou "1".

Esse número inteiro gerado possui duas funções: atribuir aleatoriedade à busca e estabelecer o sentido do deslocamento do peixe (avança ou retrocede) no espaço.

O $step_{ind}$ é um parâmetro que pode ser fixo ou mudar ao longo das iterações. Para um $step_{ind}$ fixo, recomenda-se que assuma o valor "1", pois ao assumir outros valores, como por exemplo, "2", limita-se a busca em valores múltiplos de "2". Possivelmente, a variação do $step_{ind}$ durante o processo de busca é recomendável para problemas discretos com espaço de busca grande. Propõem-se duas abordagens de variação de $step_{ind}$, são elas:

- Abordagem aleatória, onde o passo individual assume um valor aleatório, o qual garante que o peixe não saia do espaço de busca;
- Abordagem de decaimento por estagnação, onde o valor do passo individual decai de uma unidade após um certo número de iterações pré-fixado sem que haja melhorias no processo de busca.

A nova posição do peixe é obtida de acordo com (3.2).

$$x_i(t+1) = x_i(t) + \Delta\vec{x}_{ind_i}. \tag{3.2}$$

A equação (3.3) mostra que as dimensões do $\Delta\vec{x}_{ind_i}$ assumem o valor "0" ou "1" de forma aleatória para valores binários (ambiente ou dimensão específica binária). A operação de soma para obter a nova posição do peixe é substituída pelo operador lógico XOR ($\oplus$) quando se trata de valores binários, como pode ser visto na equação (3.4).

$$\Delta\vec{x}_{ind_i} = randInt[0,1], \tag{3.3}$$

$$x_i(t+1) = x_i(t) \oplus \Delta\vec{x}_{ind_i}. \tag{3.4}$$

### 3.1.2 Operador de Movimento Coletivo-instintivo

O vetor direção resultante $\vec{I}(t)$ que é dado por (2.7) no FSS é calculado da mesma forma na versão discreta do FSS. A diferença está no fato de que os valores do $\vec{I}(t)$ são arredondados, consequentemente, obtém-se um vetor de valores discretos. Outra abordagem é aplicar a função sinal (ver figura 3.1 e equação (3.5)) em $\vec{I}(t)$, com isso, mapeia-se os valores contínuos para valores discretos pertencentes ao conjunto $S = \{-1, 0, 1\}$. Os valores do vetor de movimento instintivo são desejavelmente "$-1$", "0" ou "1", porque esses valores permitem uma mudança de posição menos brusca, diminuindo as chances dos peixes (soluções candidatas) saírem do espaço de busca nas primeiras iterações.

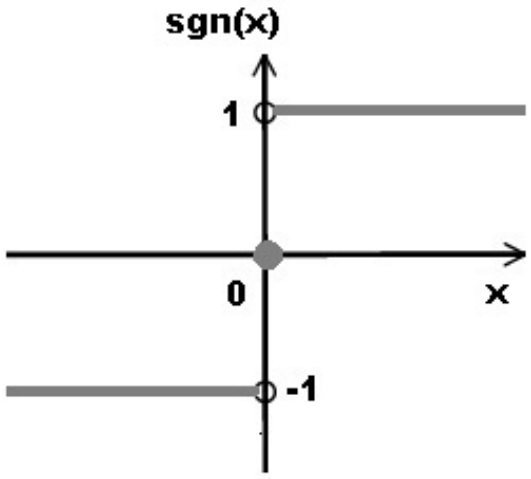


Figura 3.1: Função sinal.

$$sgn(x) = \begin{cases} -1 & se\ x < 0 \\ 0 & se\ x = 0 \\ 1 & se\ x > 0 \end{cases} . \tag{3.5}$$

A probabilidade do movimento instintivo utilizando o arredondamento invalidar uma solução por estar fora do espaço de busca aumenta em ambientes discretos pequenos. Vale salientar que o espaço de busca discreto é pequeno quando comparado ao ambiente contínuo que é infinito. Adicionalmente, os elementos do conjunto $S$ são suficientes para descrever se o peixe deve permanecer naquele ponto ("0"), avançar ("1") ou retroceder ("$-1$").

Caso o $step_{ind}$ seja igual a "1" então é aconselhável o arredondamento do vetor $\vec{I}(t)$ para o universo dos inteiros, pois os valores desse vetor arredondado certamente pertencem ao conjunto $S$ (ver prova 1). Além disso, aplicar o arredondamento em vez da função sinal permite mensurar a intensidade do deslocamento, não apenas o sentido, como por exemplo:

$$\text{sgn(-0,06)} = \text{-1},$$

$$\text{round(0,06)} = 0;$$

em um ambiente discreto, o deslocamento de "$-0,06$", preferivelmente, deve ser interpretado como "0", se for entendido como "$-1$" então está longe da realidade que de fato propiciaria esse deslocamento, caso o ambiente fosse contínuo.

**Prova 1:** Para $step_{ind} = 1$, o $\Delta\vec{x}_{ind_i}$ máximo é igual a $(1,1,...,1)$, supondo que todos os peixes se desloquem o máximo que puderem, então:

$$\vec{I} = \frac{\sum_{i=1}^{N} \overbrace{\Delta\vec{x}_{ind_i}}^{(1,1,...,1)} .\{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}}{\sum_{i=1}^{N} \{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}} \Leftrightarrow \vec{I} = \Delta\vec{x}_{ind} . \overbrace{\frac{\sum_{i=1}^{N} \{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}}{\sum_{i=1}^{N} \{f[\vec{x}_i(t+1)] - f[\vec{x}_i(t)]\}}}^{1}$$

$$\text{Logo, } \vec{I} = (1,1,...,1).$$

Analogamente, obtém-se o resultado de $\vec{I} = (-1,-1,...,-1)$ quando $\Delta\vec{x}_{ind_i} = (-1,-1,...,-1)$ (deslocamento mínimo) para todo peixe $i$.

Portanto, $(-1,-1,...,-1) \leq \vec{I} \leq (1,1,...,1)$. Dessa forma, os valores desse vetor arredondado no universo dos inteiros pertencem ao conjunto $S = \{-1,0,1\}$.

Este movimento na versão discreta do FSS desempenha a mesma função que executa no FSS original. No caso de valores binários, é utilizado o arredondamento, pois a função sinal só mapeia para o número "0" se e somente se o valor de entrada da função sinal for "0". O operador de soma aritmética na equação (2.6) é substituída pelo operador XOR ($\oplus$) quando os valores são binários.

### 3.1.3 Operador de Movimento Coletivo-volitivo

A ideia principal deste movimento no FSS é mantida em sua versão discreta. O cálculo do baricentro é feito da mesma forma que em (2.8). As mudanças estão nas equações para deslocamento do peixe em direção (busca em profundidade) ou oposição (exploração em amplitude) ao baricentro.

O parâmetro passo volitivo ($step_{vol}$) é definido para especificar a intensidade do deslocamento do peixe de/para o baricentro. O valor desse parâmetro é estabelecido da mesma forma que o $step_{ind}$, podendo ser fixo ou variável. A nova posição é determinada como em (3.6), se o

peso global do cardume aumenta em relação ao peso dele no ciclo anterior do FSS discreto; ou como em (3.7), se o peso global diminui.

$$\vec{x}_i(t+1) = \vec{x}_i(t) - step_{vol}.sgn[\vec{x}_i(t) - \overrightarrow{Bari}(t)], \tag{3.6}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + step_{vol}.sgn[\vec{x}_i(t) - \overrightarrow{Bari}(t)], \tag{3.7}$$

onde $sgn(x)$ é a função sinal.

A função sinal, apresentada na equação (3.5), foi utilizada para indicar o sentido do deslocamento no movimento volitivo. O menor ajuste possível nas dimensões da posição do peixe é de uma unidade. Por isso, a função sinal é uma alternativa interessante para ambientes discretos, pois aplica pequenos ajustes nas posições dos peixes.

As equações (3.8) e (3.9), que exprimem o movimento coletivo-volitivo, são aplicadas no vetor $\vec{x}_i$ composto por valores binários. Na equação (3.8), a posição dos peixes passam a ser aproximadamente o baricentro do cardume, representando a diminuição do raio, pois o baricentro pertence ao conjunto dos reais. Já na equação (3.9), os peixes se afastam do baricentro, ou seja, é aplicado o operador lógico de negação ($\sim$) ao baricentro aproximado que é interpretado como um vetor composto de valores booleanos.

$$\vec{x}_i(t+1) = round[\overrightarrow{Bari}(t)], \tag{3.8}$$

$$\vec{x}_i(t+1) = \sim round[\overrightarrow{Bari}(t)], \tag{3.9}$$

onde $round(x)$ é responsável por arredondar o valor real $x$ no conjunto dos números binários ($B = 0, 1$) que deve ser interpretado da seguinte forma: “0” para representar o valor lógico falso e “1” para o valor lógico verdadeiro.

### 3.1.4 Operador de Movimento de Fuga

O operador de movimento de fuga é inspirado no comportamento dos peixes quando são ameaçados por predadores, como por exemplo, tubarões. As presas tentam escapar dos seus predadores naturais nadando em direção contrária, ou seja, fugindo. Em um cardume, isso representa uma desordem, onde alguns peixes, os que foram influenciados pela presença do predador, passam a explorar outras áreas do espaço de busca ou são mortos pelos predadores.

O operador de movimento de fuga proposto melhora a capacidade de exploração do FSS, fazendo com que ele seja menos propenso a ficar preso em ótimos locais. Apenas a fuga bem sucedida é mapeada neste operador, ou seja, o peixe não morre, somente se afasta do predador ou permanece onde estava. Essa característica permite aprimorar a busca em amplitude.

Duas abordagens deste operador são propostas, são elas:

- Operador de movimento de fuga aleatório: o predador é um peixe gerado aleatoriamente a cada iteração do FSS no espaço de busca, que possui uma certa probabilidade de afugentar os peixes do cardume. Essa probabilidade também é concedida de forma aleatória quando o predador é gerado. É importante elucidar que há algumas discrepâncias quando é feito uma comparação com a realidade na natureza, como por exemplo, o predador dificilmente afugentaria um peixe que está a uma distância consideravelmente segura. No entanto, é possível que um peixe numa posição distante do predador fuja nessa modelagem aleatória. A posição dos peixes que são influenciados pelo predador é ajustada de acordo com equação (3.10);

$$\vec{x}_i(t+1) = \vec{x}_i(t) + sgn[\vec{x}_i(t) - \overrightarrow{predador}(t)], \tag{3.10}$$

  onde $\overrightarrow{predador}(t)$ é o vetor posição do predador gerado de forma aleatório ou não a cada iteração do algoritmo.

- Operador de movimento de fuga dos excluídos: somente os $N$ (valor pré-determinado) peixes com os piores valores de *fitness* fogem do predador gerado no baricentro desses peixes que estão nas regiões menos favorecidas do espaço de busca numa dada iteração. Em outras palavras, os piores peixes ajustam sua posição de acordo com (3.10), onde o $\overrightarrow{predador}(t)$ é o baricentro local dos piores peixes no instante de tempo $t$.

### 3.1.5 Busca por Cardumes Gulosa

Ambientes discretos são, em geral, bastante limitados, ou seja, pequenos quando comparados aos espaços de busca contínuos, que são infinitos. Pequenos deslocamentos dos peixes no espaço de busca discreto são mais susceptíveis a conduzir a busca para regiões menos promissoras, que possuem soluções ruins, do que um ambiente contínuo, pois não há valores intermediários entre dois inteiros. Uma alternativa de contorno para esse problema é a busca por cardumes gulosa (gFSS, *Greedy Fish School Search*).

A busca por cardumes gulosa é uma busca onde todos os operadores do FSS somente são

executados sobre um dado peixe caso a nova posição $\vec{x}(t+1)$ seja mais promissora do que a atual $\vec{x}(t)$. Em outras palavras, o deslocamento do peixe no espaço de busca de um ponto $A$ para um ponto $B$, somente ocorre caso $B$ possua mais comida do que em $A$, ou seja, apenas se a função de avaliação (*fitness*) indicar que $B$ é uma região mais promissora do que $A$. Até mesmo o operador de movimento de fuga é guiado pela função de avaliação (*fitness*).

## 3.2 Parâmetros e Pseudocódigo do FSS Discreto

A maioria dos parâmetros do FSS foram mantidos em sua versão discreta. O $step_{ind,inicial}$ e $step_{vol,inicial}$ permanecem, contudo podem ser desprezados em uma abordagem onde assumem o valor mínimo fixo "1", e também, são desprezados quando são gerados ao acaso como foi visto na subseção 3.1.1. A influência do $step$ no processo de busca discreta será explicado com mais detalhes no capítulo 4. Os parâmetros $step_{ind,final}$ e $step_{vol,final}$ não são mais modificados pelo usuário do FSS na versão discreta, pois assumem o valor fixo "1" e somente são usados em uma modelagem onde o $step$ varia por estagnação do processo de busca.

O peso máximo do peixe ($W_{scale}$) continua sendo um parâmetro modificável na busca por cardumes discreta. O peso do peixe deve ser no mínimo igual a "2", consequentemente, ($W_{scale}$) é igual a no mínimo "4", pois a importância do peso de um peixe no cálculo do baricentro pode ser comprometida. Isso ocorre, porque o peso é restringido, obtendo-se um centro de massa irreal nas primeiras iterações. Essa situação é exemplificada a seguir: supondo um ambiente de busca composto por dois peixes ($p_1$ e $p_2$) cada um com peso inicial igual a "1"($W_{scale} = 2$), na primeira iteração, $p_1$ é o mais bem sucedido na busca, e tem seu peso aumentado de "1", o outro de "0,5", logo $W_1 = 2$ e $W_2 = 1,5$; na segunda iteração, $p_2$ é o mais bem sucedido na busca, e tem seu peso aumentado de "0,5", pois o peso máximo permitido é "2", o peso de $p_1$ não pode mais aumentar, porque já está no valor máximo permitido; quando for calculado o baricentro na segunda iteração, os pesos dos peixes não corresponderam a realidade e darão a mesma importância a todos os peixes no cálculo do baricentro nas iterações seguintes, conduzindo o processo de convergência para uma solução ótima local.

Novos parâmetros foram introduzidos no FSS discreto, são eles:

- Taxa de movimento coletivo-instintivo (taxa instintiva): é a probabilidade de um peixe ser influenciado pelo sucesso instantâneo do movimento individual executado por todos os peixes, ou seja, a probabilidade de um dado peixe executar o movimento coletivo-instintivo;

- Taxa de movimento coletivo-volitivo (taxa volitiva): é a probabilidade de um dado peixe se aproximar ou afastar-se do baricentro do cardume;

- Limiar de mudança de ótimo local: é o número de vezes máximo que a posição do peixe *i* permanece sem a possibilidade de ser alterada quando o valor de *fitness* de *i* não muda ao longo das iterações. Esse limiar é responsável por alternar as posições dos peixes entre as regiões com a mesma quantidade de comida (*fitness*). Ademais, diminui as chances do algoritmo ficar preso em ótimos locais e melhora a forma de lidar com problemas com múltiplos ótimos globais e locais. No Algoritmo 3 é mostrado como este parâmetro é usado no algoritmo do FSS;

- "Número de Iterações para Decaimento": é quantidade de iterações ($N$) que o algoritmo FSS precisa permanecer estagnado, isto é, sem melhorar o processo de busca, para promover o decremento de uma unidade dos passos individuais e volitivos.

**Algoritmo 3**: Pseudocódigo da Função que Atualiza a Posição do Peixe

```
Entrada: peixe_i, fitness(t+1), x⃗(t+1), limiar
se fitness(t)=fitness(t+1) então
    Incrementar contator global do peixe_i;
    se contador global do peixe_i = limiar então
        Atualize a posição do peixe_i;
        Atribua ao contador global do peixe_i o valor "0";
senão
    Atribua ao contador global do peixe_i o valor "0";
    Atualize a posição do peixe_i.
```

O pseudocódigo da busca por cardumes discreta com o operador de movimento de fuga é mostrado em Algoritmo 4.

## 3.3 Busca por Cardumes como Operador do Algoritmo Genético

Os espaços de busca com mais de trinta dimensões e com muitos pontos de ótimos locais e globais são verdadeiros desafios para a busca por cardumes em ambientes discretos, como será visto no capítulo 4. Isso porque essa técnica originalmente não foi concebida para realizar buscas em espaços hiperdimensionais. Por isso, um algoritmo que faça busca em espaços hiperdimensionais precisa possuir excelente capacidade de exploração em amplitude e em profundidade.

**Algoritmo 4**: Pseudocódigo da Busca por Cardumes para Problemas Discretos

```
Inicialize todos os peixes aleatoriamente;
enquanto critério de parada faça
    para cada peixe no cardume faça
        avalie a função fitness;
        execute o operador de movimento individual (3.2) (3.4);
        execute o operador de alimentação (2.3);
    calcule o valor de I(t) por meio de (2.7);
    aplique a função sinal (3.5) ou arredonde para números inteiros o vetor I(t);
    para cada peixe no cardume faça
        execute o operador de movimento coletivo-instintivo (2.6);
    gere um predador de forma aleatória ou não;
    para cada peixe escolhido do cardume faça
        execute o operador de movimento de fuga (3.10);
    calcule o valor de Bari através de (2.8);
    para cada peixe no cardume faça
        se pesoCardume(t+1) > pesoCardume(t) então
            atualize a posição do peixe usando (3.6) (3.8);
        senão
            se pesoCardume(t+1) < pesoCardume(t) então
                atualize a posição do peixe usando (3.7) (3.9);
    decremente de "1" o step_ind e o step_vol após N iterações de estagnação.
```

O operador de mutação no GA é uma abordagem pseudo-aleatória para gerar diversidade durante a busca. Isso faz com que a busca se torne cada vez mais aleatória demorando mais tempo para convergir para um ótimo global. Hibridizar o GA com o FSS é uma solução que pode resolver dois problemas: a dificuldade de convergência do FSS em espaços hiperdimensionais e o operador de mutação que é essencialmente aleatório.

A hibridização do GA e FSS ocorre por meio da substituição do operador de mutação pela metaheurística FSS discreto com o operador de fuga guiado, ou seja, o peixe somente foge para região que é mais promissora. O GA e o FSS são algoritmos baseados em população, o que facilita a transição entre as metaheurísticas durante a busca. Os indivíduos da população são cromossomos durante os operadores de cruzamento e seleção do GA. Além disso, antes que uma iteração do algoritmo FSS seja executado a população de cromossomos é convertida para peixes e o seus pesos são locais na interação. O cardume é convertido em uma população de cromossomos após o término de uma iteração do FSS e a memória (informação do peso) do peixe é perdida.

A busca por cardumes como operador do algoritmo genético (GAFSS, *Genetic Algorithm with Fish School Search*) é visto como uma opção para fazer buscas em espaços discretos com alta dimensionalidade, conduzindo o processo de busca para uma solução ótima global mais rapidamente. O pseudo-código do GAFSS é mostrado no Algoritmo 5.

A ideia de elitismo no GA é mantida no GAFSS. Os indivíduos mais aptos da população não fazem parte do cardume, e consequentemente, não realizam uma busca por cardumes. Outra alternativa é permitir que os indivíduos mais aptos realizem somente uma busca por cardumes gulosa e os demais um busca não gulosa, com isso o indivíduo que representa a melhor solução encontrada pelo GAFSS não é descartado e pode ser substituído por outro indivíduo mais apto. Da mesma forma que no GA, é recomendado um taxa de elitismo que permita manter pelo menos a melhor solução encontrada até o momento.

**Algoritmo 5**: Pseudocódigo do FSS Discreto como Operador do GA

```
1  Gere uma população inicial com p peixes;
2  Avalie de acordo com a função fitness cada indivíduo;
3  enquanto critério de parada não é alcançado faça
4      Execute uma única iteração do FSS discreto nos indivíduos menos aptos;
5      Execute uma única iteração da versão gulosa do FSS discreto nos indivíduos mais
       aptos;
6      Converta o cardume em uma população de cromossomos;
7      Cruze com certa probabilidade os indivíduos obtendo os descendentes;
8      Calcule o valor de aptidão (fitness) dos descendentes;
9      Insira os novos descendentes na população;
10     Selecione os p indivíduos mais aptos da nova população;
11     Converta a população de cromossomos para um cardume;
12 Apresente o indivíduo mais apto como solução;
```

# 4 Experimentos e Resultados

Este capítulo mostra a modelagem do PLP e da função objetivo para otimização na seção 4.1, esse problema é utilizado para validar a eficácia e eficiência das técnicas propostas no capítulo 3. Adicionalmente, a seção 4.2 apresenta os experimentos realizados e resultados obtidos para resolver o PLP; e uma análise da influência da quantidade de peixes na busca. A seção 4.2 também mostra uma comparação do FSS discreto com a sua versão gulosa; o impacto do movimento de fuga na busca; e por fim, um estudo dos novos parâmetros presentes na abordagem discreta do FSS.

## 4.1 Modelagem do PLP e Ferramenta Computacional

O problema de carregamento de palete na visão do produtor pode ser modelado de diferentes formas. Um modelo é considerar cada caixa como um indivíduo, o qual pode ser um peixe ou cromossomo. Contudo, o modelo escolhido por decisão de projeto considera cada indivíduo como um representação de um conjunto de caixas colocadas sobre o palete, como é visto na figura 4.1 retirada e modificada de [13]. Assim, o indivíduo no modelo escolhido é constituído de um número de caixas menor ou igual à quantidade máxima teórica de caixas sobre o palete que é obtida por meio de (4.1). Portanto, nesse modelo é inserido o número máximo de caixas sobre o palete sem que haja sobreposições.

$$Qtd_{Maxima} = \frac{Area_{Palete}}{Area_{Caixa}}, \tag{4.1}$$

onde $Qtd_{Maxima}$ é a quantidade máxima de caixas sobre o palete, $Area_{Palete}$ é a área do palete e $Area_{Caixa}$ é a área da caixa .

A caixa é modelada da forma que foi proposta em [13], onde é representada por um vetor composto de dois números inteiros $x$ e $y$, e um número binário que representa a orientação ($o$) da caixa no palete dado um referencial (“0” = horizontal e “1” = vertical). ($x$, $y$) é o vértice inferior esquerdo do retângulo que representa a superfície da caixa como mostrado na figura 4.1.

A altura da caixa é desprezada nessa modelagem para simplificação do problema. Ademais, a figura 4.1 mostra o mapeamento do genótipo para o fenótipo do PLP. A unidade de medida não importa nessa modelagem, pois não é relevante para a solução do PLP, o que importa são as proporções entre as medidas. A distância entre dois pontos consecutivos da grade que representa o palete na figura 4.1 corresponde a um unidade ($1u$) e o menor quadrado possível tem área igual a $1u^2$.

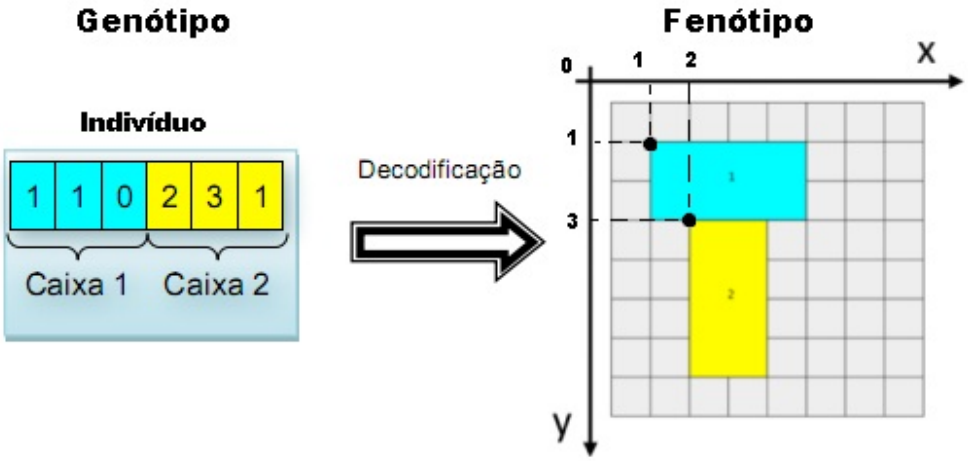


Figura 4.1: Decodificação de indivíduo composto por duas caixas.

O espaço de busca é limitado pelas dimensões do palete. Cada caixa (base retangular) é representada no indivíduo (peixe ou cromossomo) como sendo a tupla $(x, y, o)$. As condições necessárias e suficientes para a caixa não ficar fora do palete são:

- Se $o$ for igual a 1 então o valor de $x$ pertence ao intervalo $[0, C-l]$ e $y$ ao intervalo $[0, L-c]$;
- Se $o$ for igual a 0 então o valor de $x$ pertence ao intervalo $[0, C-c]$ e $y$ ao intervalo $[0, L-l]$;

onde "C" e "L" são o comprimento e a largura do palete respectivamente, e "c" e "l" são o comprimento e a largura da caixa respectivamente. Vale ressaltar que no modelo proposto por Cavalcanti Jr. em [13], somente é considerado as dimensões do palete para delimitar o espaço de busca. Com isso, parte da caixa pode ficar fora do palete. Contudo para que isso não ocorra, ele propõe que a posição da caixa seja modificada aleatoriamente até que ela se encontre completamente dentro do palete. Esse modelo não é o ideal, pois torna a busca pela solução ótima mais aleatória.

Além disso, o indivíduo como um conjunto de caixas não é a representação ideal para resolver o PLP, pois torna o espaço de busca hiperdimensional. Por exemplo, se for possível pôr

15 caixas em um determinado palete então o espaço de busca é constituído por 45 dimensões, já que cada caixa é codificada por 3 números. Logo, o indivíduo é codificado com a quantidade de números igual a três vezes a quantidade de caixas possíveis. Entretanto, essa modelagem propicia testar a busca por cardumes em ambientes discretos grandes e de alta dimensionalidade, o que potencializa o uso dessa modelagem para analisar as abordagens propostas.

A função objetivo também chamada de função de avaliação para a busca por cardumes para o PLP é descrita na equação (4.2) que é dada por:

$$F = \frac{1}{1 + \frac{Sp}{P_A}}, \tag{4.2}$$

onde $Sp$ é o total de unidades de área com sobreposição de caixas (cada sobreposição de caixas em uma mesma unidade de área é levada em conta) e $P_A$ é o percentual de área que as caixas ocupam no palete.

O objetivo do PLP é maximizar a função descrita na equação (4.2), ou seja, é um problema de otimização, onde o ponto de máximo é 1. O algoritmo para cálculo de $Sp$ é mostrado em Algoritmo 6, e foi proposto por Cavalcanti Jr. em [13].

**Algoritmo 6**: Pseudocódigo para Cálculo de Sobreposição Total ($Sp$)

```
Entrada: indivíduo
Saída: Sp
Sp = temp = 0;
Decodifique o indivíduo gerando uma lista de caixas;
para cada caixa na lista faça
    Coloque a caixa no palete;
    para cada unidade de área da caixa posta faça
        se área já ocupada então
            temp = temp + 1;
    Sp = Sp + temp;
    temp = 0;
```

Os peixes somente mudam de posição, caso a nova posição não os deixem fora do espaço de busca em todas as abordagens propostas de busca por cardumes no capítulo 3.

A ferramenta desenvolvida é baseada no SIP (Sistema Inteligente de Carregamento de Paletes) proposto por Cavalcanti Jr. em [13]. A figura 4.2 mostra a tela principal da ferramenta desenvolvida. As principais mudanças em relação ao proposto por Cavalcanti Jr. são a adição da busca por cardumes à ferramenta e a hibridização das duas técnicas FSS e GA para solucionar o PLP do produtor. A implementação da ferramenta foi feita na linguagem orientada

a objetos Java [32]. O software continua sendo dividido em três camadas: Interface Gráfica, Carregamento e Algoritmos de Inteligência Computacional. Os algoritmos propostos e o GA se encontram no módulo Algoritmos de Inteligência Computacional. As classes *Individual* (Individuo), *Fish* (Peixe) e *Chromosome* (Cromossomo) estão nesse módulo. A classe *Fish* e *Chromosome* herdam a classe *Individual*. Portanto, todo peixe e todo cromossomo são indivíduos, isso facilita a transição de peixe para cromossomo e vice-versa no GAFSS, visto que ambas as técnicas (GA e FSS) são baseadas em população.


Figura 4.2: Sistema de Carregamento de Paletes com GAFSS.

## 4.2 Validação da Busca por Cardumes em Ambientes Discretos

As dimensões do palete e da caixa para realizar os experimentos são respectivamente: $12u$ de comprimento e $10u$ de largura; e $4u$ de comprimento e $3u$ de largura. Essas dimensões implicam um espaço amostral (ou de busca) com $3,3.10^{21}$ possibilidades, esse valor foi obtido a partir das equações (4.3) e (4.4), onde $n$ é igual a 10. Portanto, é um espaço discreto relativamente grande proporcionado pela modelagem adotada para o PLP. Além disso, a função objetivo proposta apresenta vários pontos de máximo locais e globais, o que aumenta a probabilidade das metaheurísticas propostas ficarem presas em ótimos locais. A função objetivo com

as dimensões escolhidas (palete 12$u$ x 10$u$ e caixa 4$u$ x 3$u$) possui três ótimos globais que são mostrados na figura 4.3 extraída de [13].

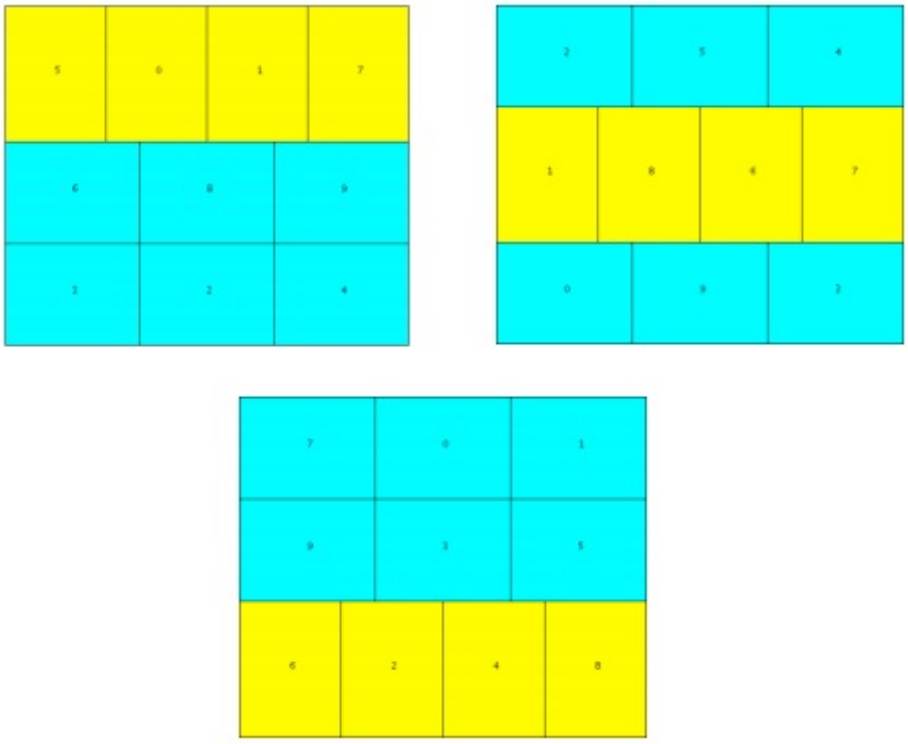

Figura 4.3: Três soluções ótimas globais para o PLP (palete 12x10 e caixa 4x3).

Sendo $C$ e $c$ o comprimento do palete e da caixa, respectivamente; $L$ e $l$ a largura do palete e da caixa, respectivamente; sabendo que o vértice inferior esquerdo da base retangular da caixa (ver figura 4.1) só pode assumir valores os quais garantam que a caixa não fique fora do palete em hipótese alguma; e supondo que as caixas podem se sobrepor, tem-se que o número de possibilidades para dispor um caixa sobre o palete ($P_{caixa}$) é dado por:

$$P_{caixa} = (C - c + 1).(L - l + 1) + (C - l + 1).(L - c + 1). \tag{4.3}$$

Logo, a partir de (4.3), calcula-se o número total de possibilidades ($P_{total}$) para $n$ pela equação (4.4) dada por:

$$P_{total} = (P_{caixa})^n. \tag{4.4}$$

O conjunto de experimentos realizados para busca em cardumes compreendem 20 simulações para cada arranjo de parâmetros diferentes ou técnicas propostas. Experimentos que envolvem somente a busca por cardumes possuem como critério de parada o número máximo de 15.000 iterações ou quando convergir para uma solução ótima (função *fitness* igual a 1). Adicionalmente, a condição de parada para os experimentos com as técnicas GAFSS e GA é o

número máximo de 500 iterações ou quando convergir para a solução ótima.

A quantidade de indivíduos para o FSS em sua versão discreta foi determinada a partir de experimentos com os seguintes parâmetros: $step_{ind}$ igual a 1; $step_{vol}$ igual a 1; peso máximo do peixe ($W_{scale}$) igual a 200 e o peso mínimo do peixe igual a 1. Vale salientar que a versão da busca por cardumes empregada nesses experimentos é baseada na versão original do FSS. Portanto, o algoritmo usado nesses experimentos é uma abordagem discreta do FSS, entretanto sem novos operadores e sem novos parâmetros (FSS original discreto).

Os resultados obtidos dos experimentos para determinar a quantidade de peixes são mostrados na tabela 4.1, que apresenta as médias e os desvios padrão dos valores de *fitness* de três conjuntos composto por vinte simulações da execução do FSS original discreto. A diferença entre esses três conjuntos de experimentos está na variação da quantidade de peixes (10, 15 ou 20 peixes) empregada no processo de busca. A taxa de convergência, a qual é o percentual de simulações que convergiram para a solução ótima, também é mostrada na tabela 4.1. A figura 4.4 apresenta o processo de convergência para solução ótima do melhor experimento de cada conjunto de simulações de 10, 15 ou 20 indivíduos.

Tabela 4.1: Resultados para o FSS original discreto variando-se a quantidade de peixes.

| **Quantidade de Peixes** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| 10 | 0,9535 | 0,0299 | 10% ($\frac{2}{20}$) |
| 15 | 0,955 | 0,0292 | 20% ($\frac{4}{20}$) |
| 20 | 0,951 | 0,019 | 0% ($\frac{0}{20}$) |

O tempo médio de uma iteração do algoritmo com 15 indivíduos é 1,157 vez maior do que com 10 indivíduos. Todavia, o cardume composto por 15 peixes obteve a maior média de valores de *fitness* e a maior taxa de convergência, convergindo em 4 simulações de 20, contra apenas 2 com cardume igual a 10 peixes. O cardume com 20 indivíduos obteve o menor desvio padrão embora tenha a pior média de *fitness*, o que representa uma maior dificuldade de sair de um ótimo local após uma convergência prematura. Essa dificuldade é evidenciada na figura 4.4 a partir da iteração 2500, quando o processo de busca permanece estagnado por muito mais de 7000 iterações. Além disso, uma iteração com 20 indivíduos demora em média 1,075 vez mais do que com 15 indivíduos. Esse aumento da duração de uma iteração ocorre devido a maior quantidade de cálculos que precisam ser feitos. Por isso, a quantidade de peixes escolhida, quando omitida, para fazer os experimentos descritos nesta seção é igual a 15.

É importante salientar que em todos os outros experimentos feitos, o peso máximo do peixe

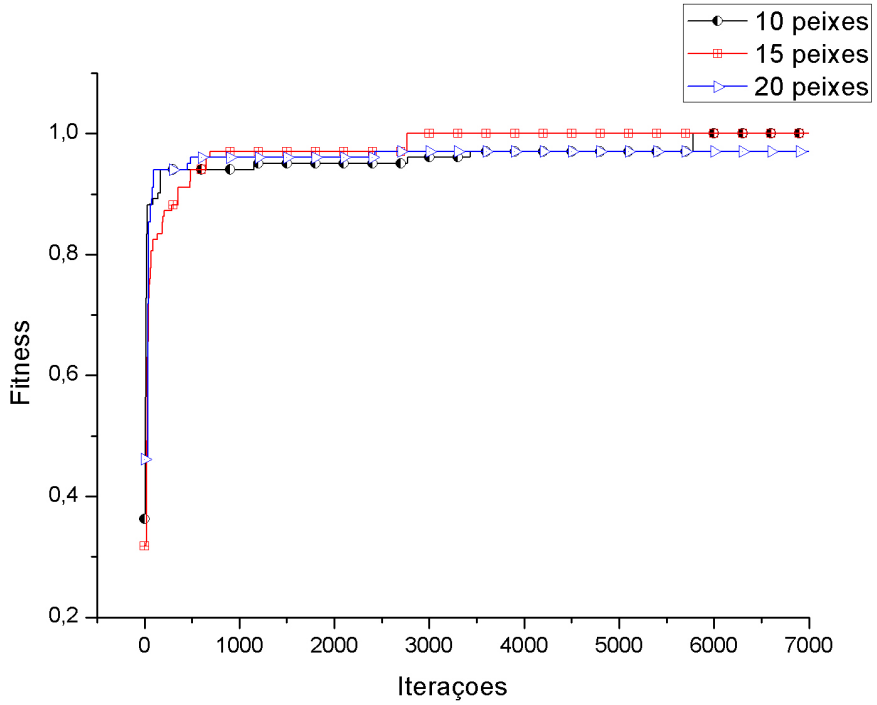


Figura 4.4: Influência da quantidade de peixes no FSS discreto.

($W_{scale}$) é igual a 200 e o seu peso mínimo é igual a 1. Esse valores foram determinados por meio de experimentos pré-liminares. Entretanto, um estudo da influência desses parâmetros na busca é proposto como trabalho futuro. Quando as dimensões das caixas e do palete forem omitidas nos experimentos, deve-se considerar o palete 12$u$ x 10$u$ e as caixas 4$u$ x 3$u$.

### 4.2.1 FSS Original Discreto versus gFSS Discreto

A tabela 4.2 apresenta a média, o desvio padrão do *fitness* e a taxa de convergência para o FSS original discreto e a sua versão gulosa cujo o cardume é composto por 10 ou 15 peixes. A média dos valores de *fitness* do gFSS são maiores do que o FSS original, e o desvio padrão desses valores é menor no gFSS, no entanto a taxa de convergência do FSS original é o dobro dessa mesma taxa para o gFSS. Portanto, o gFSS garante uma maior precisão na busca pela solução ótima do que o FSS apesar de ficar preso em ótimos locais durante mais iterações, como é mostrado na figura 4.5. A figura 4.5 representa o processo de convergência das melhores simulações obtidas durante os experimentos para o gFSS e FSS. A dificuldade de sair de ótimos locais é também ocasionada devido a capacidade de busca em amplitude ter sido reduzida no gFSS para melhorar a busca em profundidade. Por isso, o operador de movimento de fuga foi proposto para reforçar a exploração em amplitude do gFSS. Os experimentos com esse novo operador serão descritos na subseção 4.2.2.

Tabela 4.2: FSS original e gFSS discretos com 10 e 15 peixes.

| **Algoritmo (#Peixes)** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| FSS original (10) | 0,9535 | 0,0299 | 10% ($\frac{2}{20}$) |
| FSS guloso (10) | 0,9618 | 0,0159 | 5% ($\frac{1}{20}$) |
| FSS original (15) | 0,955 | 0,0292 | 20% ($\frac{4}{20}$) |
| FSS guloso (15) | 0,9657 | 0,0162 | 10% ($\frac{2}{20}$) |

É importante ressaltar que a duração de uma iteração no gFSS é um pouco menor do que no FSS original. Esse tempo menor é justificado pela redução da quantidade de instruções de acesso à memória, pois a atualização da posição do peixe somente ocorre, caso essa posição possua uma melhor avaliação (*fitness*). Portanto, o ganho na precisão da busca pela solução ótima no gFSS supera o fato de possuir uma convergência mais lenta para o ótimo global quando comparado ao FSS original. O gFSS detém uma confiabilidade maior, o que é desejável na maioria dos problemas de busca e otimização.

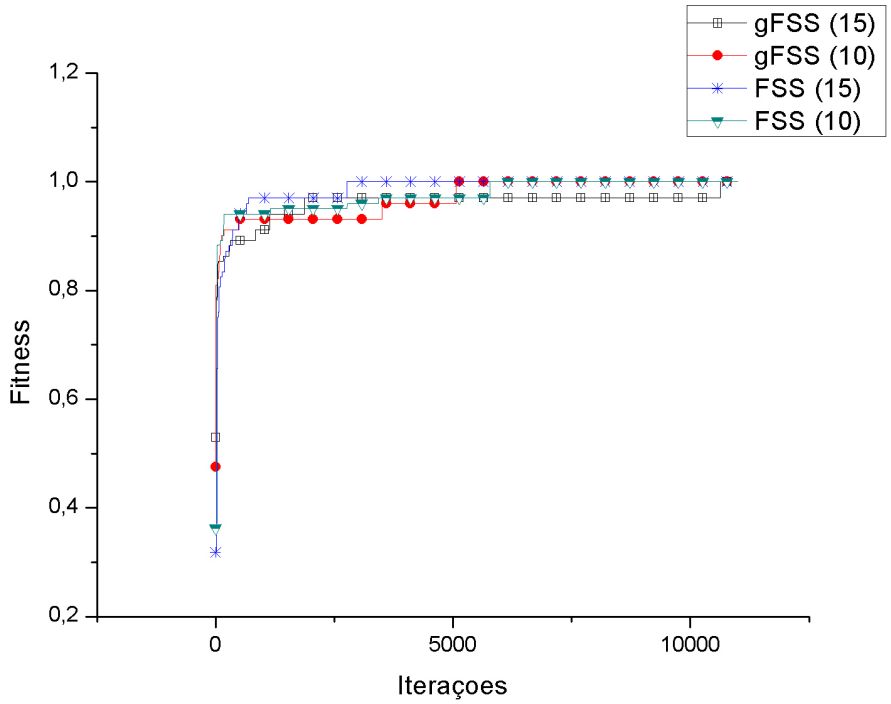


Figura 4.5: Convergência do FSS e do gFSS para cardumes com 10 e 15 peixes.

A figura 4.5 evidência também que o aumento do tamanho do cardume no gFSS, assim como no FSS (ver figura 4.4), torna a busca mais susceptível a uma convergência prematura e a ficar preso em ótimos locais.

### 4.2.2 Simulações com o Operador de Movimento de Fuga

A tabela 4.3 apresenta os resultados obtidos ao executar o movimento de fuga aleatório no FSS original e no gFSS ambos para problemas discretos. Ao comparar os resultados do FSS original sem e com o movimento de fuga (ver tabelas 4.2 e 4.3), percebe-se que este novo operador dificulta o processo de convergência do algoritmo, ficando preso em pontos de máximo locais. A taxa de convergência do FSS original sem esse novo operador é quatro vezes maior do que com ele embora o desvio padrão seja menor para uma mesma média de valores de *fitness*.

Tabela 4.3: Influência do movimento de fuga aleatório na busca.

| **Algoritmo** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| FSS | 0,9554 | 0,0185 | 5% ($\frac{1}{20}$) |
| gFSS | 0,9718 | 0,0145 | 25% ($\frac{5}{20}$) |

Ademais, o algoritmo gFSS com o operador de movimento de fuga aleatório obteve melhores resultados do que sem esse operador. A média dos valores de *fitness* resultantes das 20 simulações com esse novo operador aumentou, o desvio padrão desses valores diminuiu, e a taxa de convergência alcançada foi 2,5 vezes maior em relação ao gFSS sem o operador de movimento de fuga. Observa-se que o gFSS com o operador de movimento de fuga aleatório também superou o FSS original com e sem esse novo operador em todas as instancias (média, desvio padrão e taxa de convergência) avaliadas.

Os resultados obtidos em experimentos realizados do gFSS com o operador de movimento de fuga dos excluídos são mostrados na tabela 4.4. Essa abordagem desse novo operador obteve melhores resultados quando aplicada aos 60% piores peixes do cardume, ou seja, os 9 ($0,6.15 = 9$) menores valores de *fitness*. A tabela 4.4 mostra que as simulações para o movimento de fuga executado por 50% e 60% do cardume possuem a mesma taxa de convergência, contudo para 60% do cardume, obteve-se uma maior média do *fitness* e um menor desvio padrão. Vale salientar que o valor de *fitness* igual a 0,9701 corresponde ao segundo maior valor do conjunto imagem da função objetivo que se quer maximizar para o PLP com as dimensões de palete 12x4 e das caixas 4x3. Nessa mesma tabela, percebe-se que a taxa de 60% conduz a busca em 4 simulações das 20 para obter o ótimo global contra apenas 1 para a taxa de 30%. É importante notar que o desvio padrão dos valores de *fitness* para taxa de 30% é menor do que a da taxa de 60% em relação a uma média de aproximadamente 0,97, porque houve mais eventos de convergência para o ponto de máximo da função objetivo ($fitness = 1,0$).

Por fim, a abordagem aleatória do operador de movimento de fuga atingiu melhores resul-

Tabela 4.4: Influência do movimento de fuga dos excluídos na busca.

| **Taxa de excluídos** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| 30 | 0,9702 | 0,0085 | 5% ($\frac{1}{20}$) |
| 50 | 0,9712 | 0,0176 | 20% ($\frac{4}{20}$) |
| 60 | 0,9717 | 0,0165 | 20% ($\frac{4}{20}$) |

tados em relação a abordagem dos excluídos para o PLP com dimensões 12 x 10 (palete) e 4 x 3 (caixas). Uma análise detalhada dessas duas abordagens é proposta como trabalho futuro. É importante salientar que todos os experimentos descritos até o momento não incluem os novos parâmetros enunciados na seção 3.2 para a busca por cardumes em ambientes discretos. Esses parâmetros melhoram significativamente a capacidade de convergência para o ótimo global, como será visto nos experimentos em (i), (ii) e (iii).

**(i) Limiar de Mudança de Ótimo Local**

A tabela 4.5 mostra os resultados obtidos ao incorporar no gFSS com operador de movimento de fuga aleatório a possibilidade do peixe se deslocar de uma região para outra de mesma quantidade de comida, isto é, mesmo valor de *fitness*. Para uma abordagem onde o valor do parâmetro "Limiar de Mudança de Ótimo Local" assume um número natural aleatório gerado por uma distribuição uniforme no intervalo $[0, diagonal_{caixa}]$ ($diagonal_{caixa} = 5$ para caixa 4 x 3), constatou-se uma taxa de convergência de 75%. Enquanto que para o $limiar = 1$, a taxa de convergência é superior, pois as simulações convergiram para a solução ótima em 90% dos casos (18 de 20 experimentos). Ademais, o $limiar = 1$ proporciona uma média maior e um desvio padrão menor dos valores de *fitness* do que quando o valor desse parâmetro é escolhido de modo aleatório. Isso assegura melhores soluções para o PLP em estudo.

Tabela 4.5: Influência do parâmetro "Limiar de Mudança de Ótimo Local" na busca.

| **Limiar de Mudança** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| Aleatório | 0,9925 | 0,0133 | 75% ($\frac{15}{20}$) |
| Fixo em 1 | 0,997 | 0,0092 | 90% ($\frac{18}{20}$) |

A tabela 4.6 apresenta a média e o desvio padrão para as 15 melhores simulações do limiar igual a 1 e do limiar aleatório, portanto os experimentos que não alcançaram a solução ótima foram desprezados. Com uma média de quase 2600 iterações a menos e um desvio padrão

bastante inferior, o limiar fixo em 1 converge mais rapidamente para solução ótima do que o limiar aleatório.

Tabela 4.6: Número de iterações das 15 melhores simulações com diferentes limiares.

| **Limiar de Mudança** | **Média** (Iterações) | **Desvio Padrão** (Iterações) |
|---|---|---|
| Aleatório | 7359 | 3294 |
| Fixo em 1 | 4792 | 2592 |

Experimentos com o limiar fixo em 3 e 5 também não proporcionaram resultados melhores do que o limiar igual a 1. O gFSS com este novo parâmetro e o operador movimento de fuga foi capaz de convergir em 13 simulações a mais do que o gFSS somente com o operador de fuga (sem o "Limiar de Mudança de Ótimo Local"). Isso mostra que o processo de busca se torna menos susceptível a ficar preso em ótimos locais com este parâmetro.

**(ii) Probabilidades da Execução dos Movimentos Coletivos**

Os parâmetros responsáveis por definir as probabilidades de ocorrência do movimento coletivo instintivo e volitivo foi incluído no algoritmo do gFSS juntamente com o operador de movimento de fuga aleatório e o parâmetro "Limiar de Mudança de Ótimo Local" fixado em 1. Experimentos pré-liminares foram realizados para resolver o PLP com palete (12 x 10) e caixas (4 x 3), determinando-se uma taxa de movimento coletivo-instintivo igual a 80% e de 60% a taxa de movimento coletivo-volitivo. A tabela 4.7 apresenta a quantidade média e o desvio padrão das iterações executadas por 20 experimentos quando essas duas taxas são iguais a 100%, e quando as taxas instintiva e volitiva são iguais a 80% e 60%, respectivamente. Vale ressaltar que essas taxas diminuem a duração média de um iteração do algoritmo, haja vista a não execução dos operadores de movimento coletivos em alguns momentos. A partir da tabela 4.7, pode-se concluir que a solução ótima é encontrada mais rapidamente ao incorporar a probabilidade da ocorrência do movimento coletivo na busca por cardumes para a mesma taxa de convergência. Os gráficos *boxplots* do número de iterações necessárias para encontrar a solução ótima de cada um daqueles conjuntos de experimentos são apresentados na figura 4.6.

A figura 4.6 ilustra dois *boxplots*, onde o da direita representa o conjunto de simulações para o gFSS com a taxa instintiva igual a 80% e a taxa volitiva igual a 60% e o *boxplot* da esquerda representa as simulações com essas taxas iguais a 100%. Os losangos fora da linha vertical representam os *outliers*, o quadrado representa a média e os triângulos apontando para cima significam valores máximos, desconsiderando os *outliers* quando existirem. O triângulo

Tabela 4.7: Influência das taxas instintiva e volitiva no tempo de convergência.

| **Taxa de Movimento Instintivo** | **Taxa de Movimento Volitivo** | **Média** (Iterações) | **Desvio Padrão** (Iterações) | **Taxa de Convergência** |
|---|---|---|---|---|
| 60% | 80% | 5515 | 3983 | 90% ($\frac{18}{20}$) |
| 100% | 100% | 6766 | 4298 | 90% ($\frac{18}{20}$) |

apontando para baixo ilustra os valores mínimos.

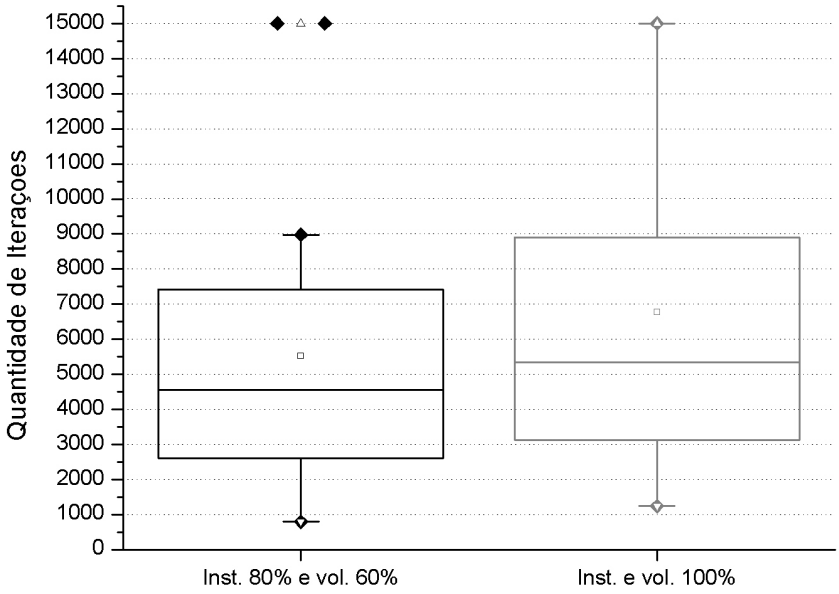


Figura 4.6: Influência das taxas instintiva e volitiva na busca pela solução ótima.

Na figura 4.6, é possível ver claramente que os dados do conjunto representado pelo *boxplot* da direta possuem uma maior mediana, podendo-se afirmar que o número de iterações necessárias para encontrar a solução ótima tendem a ser maior do que o do conjunto representado pelo *boxplot* da esquerda. E também, o *boxplot* da direta possui uma maior dispersão, já que tanto o IIQ (Intervalo Interquartil) quanto o tamanho das linhas verticais são maiores. Os gráficos ainda evidenciam que cerca de 90% dos experimentos convergiram para solução ótima em menos de 9000 iterações para as taxas volitiva e instintiva de 60% e 80%, respectivamente. Entretanto, menos de 75% das simulações encontram a solução ótima quando essas taxas são iguais a 100%.

**(iii) Decrescimento dos Passos Devido à Estagnação da Busca**

Experimentos realizados mostraram que a abordagem do parâmetro número de iterações para decaimento proposta ao ser incluído no gFSS não melhorou o processo de busca pela solução ótima no PLP em estudo. As simulações foram feitas com o $step_{ind}$ e o $step_{vol}$ inicialmente calculados de acordo com a equação (4.5). Logo, o $step_{ind}$ e o $step_{vol}$ (palete 12 x 10 e caixas 4 x 3) são iguais a 3. A quantidade máxima de iterações que o melhor *fitness* não mudou assumiu os valores de 200, 400 e 600 para cada conjunto de 20 experimentos. Dentre esses valores, obteve-se melhores resultados com 400. Contudo, a taxa de convergência é de 85%, o que é 5% menor do que os experimentos com passos fixos. Vale ressaltar que o decaimento dos passos pode promover resultados satisfatórios para outras abordagens do PLP ou outros problemas.

$$step_{inicial} = randInt(\frac{diagonal_{palete}}{diagonal_{caixa}}), \tag{4.5}$$

onde o $randInt(x)$ é uma função que arredonda o valor de $x$ para um número inteiro, $diagonal_{palete}$ é a diagonal do palete, $diagonal_{caixa}$ é a diagonal da caixa, e o $step_{inicial}$ é o valor inicial que é atribuído aos $step_{ind}$ e o $step_{vol}$.

### 4.2.3 GA Hibridizado com FSS versus GA

O PLP com palete (12 x 10) e caixas (4 x 3) foi solucionado utilizando o GA e o GAFSS descrito em 3.3. Para o GA resolver o PLP, restringiu-se o vértice inferior da base retangular da caixa (ver figura 4.1) nas dimensões do palete. Entretanto, mesmo com essa restrição parte da caixa pode ficar fora do palete, se isso ocorrer é aplicado uma mutação na caixa (parte do cromossomo) até que ela seja posta totalmente sobre o palete. Os parâmetros usados no conjunto de 20 experimentos para o GA foram os seguintes: taxa de mutação, taxa de recombinação, taxa de mutação local iguais a 50%; tamanho da população igual a 50; e percentual de elitismo igual a 25%. Esses valores foram escolhidos de acordo com o que foi proposto por Cavalcanti Jr. em [13]. O operador de recombinação adotado consiste em cruzar cada um dos indivíduos com todos os outros, porém o cruzamento tem uma probabilidade igual a taxa de recombinação para ocorrer. O operador de mutação é composto pela probabilidade de escolha de um indivíduo para ser mutado (taxa de mutação) e pela probabilidade para alterar um elemento do cromossomo (taxa de mutação local).

O GAFSS implementado possui os mesmos operadores de cruzamento e seleção do GA a fim de realizar uma comparação justa. Os valores dos parâmetros do GAFSS para o PLP abordado foram determinados empiricamente, são eles: taxa de recombinação igual a 30%; tamanho

da população igual a 50; percentual de elitismo igual a 2%; passos fixos em 1; peso mínimo e máximo do peixe iguais a 1 e 200, respectivamente; "Limiar de Mudança de Ótimo Local" igual a 1; e taxas volitiva e instintiva iguais a 70%. O parâmetro de número de decaimento foi desconsiderado, pois não obteve-se resultados satisfatórios com ele. É importante ressaltar que as versões do FSS que são executadas no GAFSS possuem o operador de movimento de fuga aleatório.

A tabela 4.8 mostra os resultados obtidos pelo GA e GAFSS nas simulações realizadas. O critério de parada da execução desses algoritmos foi o número máximo de iterações igual a 500 ou quando encontrar a solução ótima. Constata-se nessa tabela que o GAFSS convergiu em todas os experimentos, enquanto que o GA convergiu para a solução ótima em apenas 90% dos casos.

Tabela 4.8: Resultados dos experimentos para resolver o problema PLP com GAFSS e GA.

| **Algoritmo** | **Média** (*Fitness*) | **Desvio Padrão** (*Fitness*) | **Taxa de Convergência** |
|---|---|---|---|
| GA | 0,997 | 0,0092 | 90% ($\frac{18}{20}$) |
| GAFSS | 1 | 0 | 100% ($\frac{20}{20}$) |

A figura 4.7 mostra três gráficos do tipo *boxplot* da quantidade de iterações para os conjuntos de experimentos do gFSS, GA e GAFSS. Os parâmetros do gFSS foram escolhidos para obter os melhores resultados de acordo com a seção 4.2.

A tabela 4.9 evidencia que uma iteração do GA demora em média mais tempo para ser executada do que uma iteração do gFSS e do GAFSS no PLP. Porém, o gFSS precisa de uma quantidade muito maior de iterações para encontrar a solução ótima. O GAFSS possui um tempo médio de iteração cerca de 60% do tempo médio de iteração do GA e converge em média em menos iterações, vide figura (4.8). O tempo médio de uma iteração é a média da duração de uma iteração de cada um dos 20 experimentos.

Observa-se na figura 4.7 que o gFSS em quase 75% dos casos somente converge para a solução ótima após 7500 iterações, podendo até mesmo após 9000 iterações não convergir em alguns casos. Ademais, há dois *outliers* para o gFSS que não são mostrados na figura 4.7.

A figura 4.8 mostra que o GAFSS obteve resultados melhores do que o GA para o PLP. A mediana do conjunto de simulações do GAFSS é aproximadamente 30 iterações a menos do que a mediana do GA, logo o número de iterações necessárias para encontrar a solução ótima tendem a ser menor no GAFSS do que no GA. Adicionalmente, percebe-se que por volta de 400 iterações, o GAFSS converge para solução ótima em 100% das simulações, enquanto o GA

converge em cerca de 80% dos experimentos para esse mesmo número de iterações.

Tabela 4.9: Tempo médio de uma iteração em segundos para o gFSS, GA e GAFSS.

| **Algoritmo** | **Média** (segundos) | **Desvio Padrão** (segundos) |
|---|---|---|
| gFSS | 0,0025 | 0,0008 |
| GA | 0,0158 | 0,003 |
| GAFSS | 0,0094 | 0,0016 |

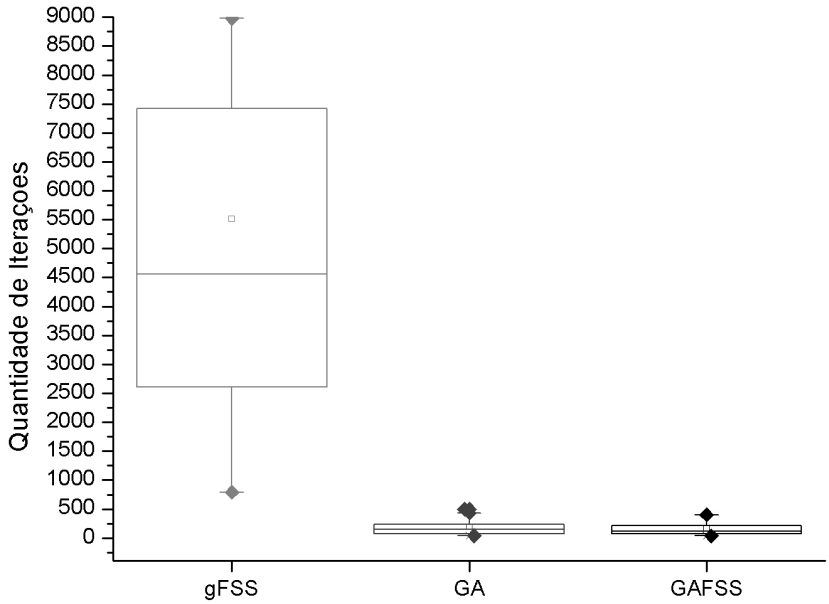


Figura 4.7: *Boxplots* da quantidade de iterações do gFSS, GA e GAFSS.

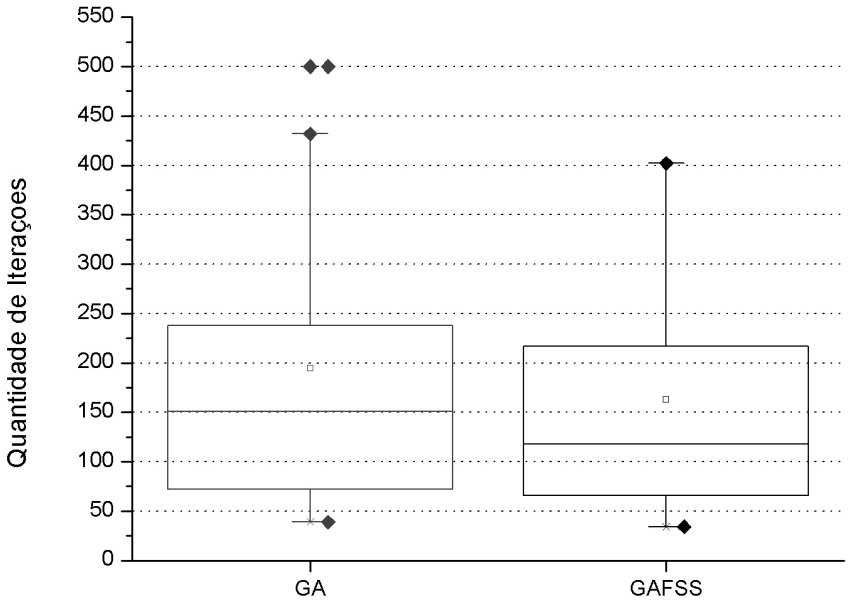


Figura 4.8: *Boxplots* evidenciados da quantidade de iterações do GA e GAFSS.

# 5 Conclusão

Este trabalho constitui uma contribuição para área de inteligência computacional, mostra o potencial do FSS para trabalhar em ambientes discretos e contribui para o amadurecimento dessa técnica recente. A busca por cardumes para problemas discretos proposta pode ser utilizada para encontrar a solução ótima de diversos problemas complexos de alta dimensionalidade, como por exemplo, problemas de otimização combinatória, processamento digital de imagens, logística, etc.

Os resultados discutidos no capítulo 4 mostram que o FSS original discreto apresentou resultados não muito satisfatórios devido aos operadores de movimento coletivo que ocorrem mesmo se a nova posição do peixe for pior que a anterior. Por se tratar de um ambiente discreto, as mudanças são bruscas quando comparadas a um espaço de busca contínuo, o que leva o processo de busca para regiões ruins do espaço. Por isso, o gFSS é mais recomendável para fazer buscas em espaço discreto.

A quantidade de parâmetros é importante, pois quanto maior for mais difícil será para o usuário final da técnica escolher o arranjo de parâmetros ideal para solucionar o problema dele. Embora o número de parâmetros da busca por cardumes em ambiente discreto seja maior do que num ambiente contínuo, alguns parâmetros podem ser suprimidos quando os valores representados pelo vetor são pequenos, como por exemplo, pertencem a um intervalo de [0, 10]. Alguns exemplos desses parâmetros são: os passos e o limiar de mudança de ótimo local quando fixados no valor de 1; e o número de iterações para decaimento.

A introdução de um novo operador, o operador de movimento fuga, inspirado no comportamento dos peixes na presença de predadores, permite que a técnica FSS alcance melhores resultados na busca com um menor custo computacional. Vale salientar que essa contribuição não se trata de alterações nos operadores existentes, mas da importância de modelar outros comportamentos dos cardumes que auxiliam no processo de busca. Esse operador melhorou de forma significativa a busca em amplitude.

Problemas discretos com grandes quantidades de dimensões, maiores que trinta dimensões,

e vários ótimos locais são difíceis de serem resolvidos pelo paradigma FSS discreto. Tendo em vista isso, esse trabalho apresentou outra contribuição que é a hibridização com o algoritmo genético. O FSS como operador do GA sem o operador de mutação mostrou ser uma boa metaheurística híbrida para solucionar problemas discretos onde o espaço de busca é multimodal e hiperdimensional. Além disso, o GAFSS foi capaz de convergir para soluções ótimas em bem menos iterações do que o GA com o operador de mutação para o PLP de palete (12 x 10) e caixas (4 x 3). Portanto, o GAFSS é a melhor técnica para solucionar o PLP abordado.

Ademais, o custo computacional de uma iteração do GAFSS é menor do que o de uma iteração do GA com o operador de mutação essencialmente aleatório, pois a taxa de cruzamento necessária no GA para obter bons resultados no problema abordado é 20% mais alta. Uma taxa de cruzamento mais alta implica um maior número de instruções para serem processadas, o que compensa o aumento do custo computacional devido à hibridização. Adicionalmente, a quantidade de parâmetros do GAFSS pode ser reduzida ao utilizar versões mais simples do FSS e gFSS.

Por fim, a ferramenta "Sistema Inteligente de Carregamento de Palete" proposta por Cavalcanti Jr. em [13] foi melhorada ao incorpora as técnicas desenvolvidas neste trabalho. A ferramente melhorada pode ser usada pelos carregadores que dispõe caixas sobre os paletes diminuindo os custos de trasporte de mercadorias.

A busca por cardumes para problema discreto proposta não é uma abordagem definitiva. Essa abordagem discreta não possui a mesma capacidade de exploração em profundidade presente no FSS original contínuo que é reforçada pelo decaimento do passo ao longo das iterações. Quando o passo é igual a 1, não é possível decrementá-lo, pois é o menor valor em módulo diferente de zero no ambiente discreto. Tendo em vista isso, um trabalho futuro é determinar alternativas para melhorar a capacidade de exploração em profundidade da busca por cardumes discreta.

O operador de movimento de fuga proposto pode ser usado no algoritmo FSS contínuo em trabalhos futuros. Adicionalmente, FSS discreto se apresenta como uma nova alternativa para trabalhar em ambientes discretos e pode obter resultados melhores em determinados problemas do que as técnicas já existentes. Outros trabalhos futuros propostos são: dividir o cardume em subcardumes, permitindo que a técnica convirja para mais de uma solução ótima; trabalhar sob novas modelagens dos operadores, por exemplo, permitir que o peixe emagreça devido ao nado frenético; e análise detalhada das duas abordagens para o operador de movimento de fuga propostas. Outro aspecto que é recomendado como trabalho futuro é uma análise paramétrica aprofundada dos algoritmos propostos.

# *Referências Bibliográficas*


[1] REEVES, C. R. *Modern Heuristic Techniques for Combinatorial Problems*. New York: McGraw - Hill International, 1995. 320 p. 320p.

[2] GAREY, M.; JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. [S.l.]: New York: W. H. Freeman, 1983. 340 p.

[3] ENGELBRECHT, A. P. *Computational Intelligence: An Introduction*. [S.l.]: John Wiley & Sons, 2007. 310 p.

[4] KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. *In Proc. of the IEEE Int. Conf. on Neural Networks*, Piscataway, NJ., USA, IV, p. 1942–1948, 1995. IEEE service center, Piscataway, NJ.

[5] DORIGO, M. *Optimization, Learning and Natural Algorithms*. Tese (Doutorado) — Politecnico di Milano, 1992.

[6] KARABOGA, D.; BASTURK, B. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*, v. 39, p. 459–171, 2007.

[7] EIBEN, A.; SMITH, J. *Introduction to Evolutionary Computing*. [S.l.]: Spring, 2003. 316 p.

[8] GOLDBERG, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. [S.l.]: Addison-Wesley Longman Publishing Co, 1989. 372 p.

[9] CHIONG, R. *Nature-Inspired Algorithms for Optimization*. [S.l.]: Heidelberg: Springer-Verlag, 2009. 536 p.

[10] PARRISH, J.; VISCIDO, S.; GRÜNBAUM, D. Self-organized fish schools: an examination of emergent properties. *Biological Bulletin*, v. 202, p. 296–305, 2002.

[11] BASTOS FILHO, C. J. A. et al. A novel search algorithm based on fish school behavior. *IEEE International Conference on Systems, Man, and Cybernetics*, p. 39–43, 1995.

[12] RAM, B. The pallet loading problem: A survey. *International Journal of Production Economics*, p. 217–225, 1992.

[13] CAVALCANTI JR., G. M. SIP - Sistema Inteligente de Carregamento de Paletes. *Trabalho de Conclusão de Curso de Bacharel em Engenharia da Computação da Universidade de Pernambuco*, p. 49, 2009. Disponível em: `http://tcc.dsc.upe.br/20091/TCC_George_Moraes_SIP.pdf`. Último acesso em 28/5/2011.

[14] FERNANDES, J. C. de F. *Administração de material: um enfoque sistêmico: teoria e prática*. [S.l.]: Rio de Janeiro: Livros Técnicos e Científicos Editora, 1981. 251 p.

[15] HODGSON, T. A combined approach to the pallet loading problem. *IIE Transactions*, v. 14 (3), p. 176–182, 1982.

[16] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Prentice Hall - 3rd edition, 2009. 1152 p.

[17] MADEIRO, S. S. *Buscas Multimodais por Cardumes Baseados em Densidade*. Dissertação (Mestrado) — Universidade de Pernambuco, 2010.

[18] EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. *Micro Machine and Human Science*, p. 39–43, 1995.

[19] BASTOS FILHO, C. J. A. et al. On the influence of the swimming operators in the fish school search algorithm. *IEEE International Conference on Systems, Man, and Cybernetics*, 2009.

[20] BONABEAU, E.; DORIGO, M.; THERAULAZ, G. *Swarm Intelligence: from natural to artificial systems*. [S.l.]: Oxford University Press, 1999. 320 p.

[21] HEPPNER, F.; GRENANDER, U. The ubiquity of chaos. In: ____. [S.l.]: Amer Assn for the Advancement, 1990. cap. A stochastic nonlinear model for coordinated bird flocks, p. 247.

[22] SHI, Y.; EBERHART, R. C. A. Modified particle swarm optimizer. *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, p. 69–73, 1998.

[23] TURING, A. Machine intelligence. In: ____. [S.l.]: Edinburgh University Press, 1969. cap. Intelligent Machinery, p. 3–23.

[24] TURING, A. M. Computing machinery and intelligence. *Mind 59*, p. 433–460.

[25] BREMERMANN, H. J. Self-organizing systems. In: ____. [S.l.]: Spartan Books, 1962. cap. Optimization through Evolution and Recombination, p. 563.

[26] DARWIN, C. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. [S.l.]: John Murray, 1859. 459 p.

[27] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. [S.l.]: University of Michigan Press: Ann Arbor, MI, 1975. 211 p.

[28] SERRADA, A. P. *Una introducción a la Computación Evolutiva*. Tese (Doutorado) — Madri, 1996.

[29] SRINIVAS, M.; PATNAIK, L. M. Genetic algorithms: A survey. *IEEE Computer*, v. 27, p. 17–26, 1994.

[30] DAVIS, L. D. *Handbook of genetic algorithms*. [S.l.]: Van Nostrand Reinhold, 1991. 385 p.

[31] GEYER-SCHULTZ, A. *Fuzzy rule-based expert systems and genetic machine learning*. [S.l.]: PhysicaVerlag Heidelberg, 1997. 432 p.

[32] DEITEL, H. M.; DEITEL, P. J. *Java Como programar*. 6. ed. [S.l.]: Prentice-Hall, 2005. 1152 p.