UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JOSÉ DA SILVA LUCENA

TREINAMENTO POPULACIONAL COM MÚLTIPLAS HEURÍSTICAS

SÃO LUÍS
2014

JOSÉ DA SILVA LUCENA

TREINAMENTO POPULACIONAL COM MÚLTIPLAS HEURÍSTICAS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alexandre César Muniz de Oliveira

SÃO LUÍS

2014

Lucena, José da Silva

Treinamento populacional com múltiplas heurísticas / José da Silva Lucena. - São Luís, 2014.

50 f.

Orientador: Alexandre César Muniz de Oliveira.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2014.

1. Algoritmos Evolutivos 2. Treinamento Populacional 3. Heurísticas 4. Minimização de pilhas abertas

CDU 004.023

JOSÉ DA SILVA LUCENA

TREINAMENTO POPULACIONAL COM MÚLTIPLAS HEURÍSTICAS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em 20 de JANEIRO de 2014.

BANCA EXAMINADORA

Prof. Dr. Alexandre César Muniz de Oliveira
(Orientador)

Prof. Dr. Geraldo Braz Júnior

Prof. Dr. Samyr Béliche Vale

À minha bebezinha de 12 anos, Júlia Sol.

AGRADECIMENTOS

Agradeço a tudo e a todos que contribuíram, de todas as formas, para a conclusão de todo este trabalho.

“Aprendi o segredo da vida vendo as pedras que choram sozinhas no mesmo lugar.”

(Raul Seixas)

RESUMO

Em Algoritmos Evolutivos (AE's), cada solução do espaço de busca possui um conjunto de informações agregadas, que podem ser utilizadas para direcionar o processo evolutivo. A forma como essas informações são geradas e o modo como são utilizadas no AE determinam a sobrevivência da solução e de seus futuros descendentes. O uso de heurísticas é um exemplo de aplicação desse procedimento. O presente trabalho tem como objetivo principal realizar um estudo de caso da utilização de múltiplas heurísticas na estrutura de um Algoritmo Evolutivo. Trata-se de uma extensão do método conhecido como Treinamento Populacional em Heurísticas (TPH), que utiliza os resultados de aplicações de uma função heurística sobre o espaço de soluções para configurar operadores evolutivos e, dessa forma, acelerar a detecção de regiões promissoras. O AE será aplicado sobre o problema de Minimização de Pilhas Abertas, juntamente com uma nova heurística desenvolvida para o mesmo.

Palavras-chave: Algoritmos Evolutivos. Treinamento Populacional. Heurísticas. Minimização de Pilhas Abertas.

ABSTRACT

In Evolutionary Algorithms (EA), each solution of the search space has a set of aggregate information, which can be used to guide the evolutionary process. The way this information is generated and how it is used in AE determine the survival of the solution and their future offspring. The use of heuristics is an example of application of this procedure. This paper aims to carry out a case study of the use of multiple heuristics in the structure of an Evolutionary Algorithm. This is an extension of the method known as Population Training Heuristics (PTH), which uses the results of applications of a heuristic function on the space of solutions to configure evolutionary operators and thus accelerate the detection of promising regions. The EA is applied to the Minimization Of Open Stacks Problem, along with a new heuristic developed for the same.

Keywords: Evolutionary Algorithms. Population Training. Heuristics. Minimization of Open Stacks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AE	Algoritmo Evolutivo
TPH	Treinamento Populacional em Heurísticas
ATP	Algoritmo de Treinamento Populacional
ATPM	Algoritmo de Treinamento Populacional em Múltiplas Heurísticas
AEH	Algoritmo Evolutivo Híbrido
MOSP	*Minimization of Open Stacks Problem*
MD	Minimização de Danos

SUMÁRIO

1 INTRODUÇÃO

O processo de evolução das espécies na natureza, explicado atualmente pela *Teoria Sintética da Evolução*, também conhecida como *Mutacionismo* ou *Neodarwinismo*, é uma forma de otimização em que o objetivo é a manutenção da vida no planeta. A melhor solução é o estado em que a probabilidade de manutenção da vida é a máxima, com os recursos existentes. Tendo em vista o grau de eficiência observado neste processo, nada mais natural do que tentar utilizar seus métodos na resolução de problemas matemáticos de difícil solução, nos casos em que se aplica. Assim foram desenvolvidos os Algoritmos Evolutivos, que utilizam analogias dos mecanismos e conceitos do processo de evolução das espécies, como a seleção natural, recombinação genética, mutação, geração, indivíduo, competição, etc.

Um Algoritmo Evolutivo (AE) tenta encontrar a solução para um problema de otimização através da aplicação sucessiva de operadores evolutivos sobre um conjunto de soluções inicial. Uma instância de um AE é frequentemente caracterizada pela forma como esses operadores trabalham em sua arquitetura.

A correspondência entre termos apresentada na Tabela 1 é essencial para o entendimento dos AE's.

Tabela 1 - Correspondência entre termos do Neodarwinismo e dos AE's

Neodarwinismo	*Algoritmos Evolutivos*
indivíduo	solução
população	conjunto de soluções
geração	uma população em determinado tempo
grau de adaptação ao meio	*fitness*
cromossomo	solução representada computacionalmente
gene	componentes do cromossomo

O objetivo de um AE é, a partir de uma população inicial (geração inicial), formar sucessivas gerações através de operadores evolutivos, de forma que a geração seguinte possua indivíduos melhores que a anterior. O resultado de um AE é sempre um conjuto de soluções – a geração final.

Nem sempre o AE encontra a solução ótima, mas a qualidade das

melhores soluções encontradas (as soluções finais) frequentemente possui excelente relação custo-benefício.

Ao longo da história foram criados vários termos referentes a classes de algoritmos evolutivos. Como por exemplo: algoritmos genéticos, programação evolutiva, estratégias evolutivas, programação genética, sistemas classificadores, etc. As diferenças entre essas classes são muito tênues, residindo apenas nos operadores e parâmetros utilizados. Com o surgimento dos algoritmos híbridos esse sistema de classificação ficou ainda mais nebuloso.

Atualmente, usa-se a denominação “Algoritmo Evolutivo” para todo algoritmo que utiliza a Teoria da Evolução das Espécies como analogia para resolução de problemas. Por conseguinte, Computação Evolutiva refere-se à área da computação que estuda os Algoritmos Evolutivos. Alguns eventos importantes para a consolidação dessa área, segundo (ZUBEN, 2003), foram:

- Encontro de pesquisadores em algoritmos evolutivos na conferência Parallel Problem Solving from Nature (PPSN), realizada em Dortmund em 1990.
- Organização da primeira International Conference on Genetic Algorithms (ICGA’91) em 1991.
- Chegada a um consenso para o nome da área – computação evolutiva – e o estabelecimento de um jornal homônimo pelo MIT Press em 1993.
- Inclusão da área na primeira World Conference on Computational Intelligence (WCCI) em 1994, que incluiu: redes neurais artificiais, sistemas nebulosos e algoritmos evolutivos.

Algoritmos Evolutivos são utilizados quando não há outra técnica específica conhecida para resolver o problema. No entanto, o AE pode ser construído de forma a utilizar conhecimento específico do problema para evitar caminhos inúteis e/ou escolher caminhos mais propícios a resultados satisfatórios em menor tempo. Neste caso, diz-se que o AE utiliza Busca Heurística no seu processo evolutivo. Esta técnica será utilizada neste trabalho.

Em otimização, heurísticas são procedimentos aproximativos, i.e., algoritmos de busca capazes de encontrar soluções aproximadas (sub-ótimas) de boa qualidade em tempo computacional razoável. AE’s são considerados heurísticas de busca (ou otimizadores heurísticos), pois utilizam-se do conhecimento empírico e também teórico para, combinando soluções consideradas de boa qualidade,

gerarem soluções ainda melhores (OLIVEIRA, 2004).

Atualmente, são raros os estudos que utilizam AEs genéricos para resolução de problemas. A popularização do uso de heurísticas em AE's é um passo natural, uma vez que, quanto mais se sabe sobre um problema, provavelmente maiores são as chances de conseguir resolvê-lo.

Utilizar-se-á o termo “Algoritmo Evolutivo Híbrido” (AEH) para designar todo AE que, para acelerar seu desempenho, utiliza heurísticas de busca local – heurísticas capazes de trabalhar intensivamente num pequeno subconjunto do espaço de soluções e, com isso, encontrar as melhores soluções locais. Essas heurísticas demandam alto custo computacional, daí a necessidade de que sejam aplicadas somente em regiões do espaço de busca onde haja alta probabilidade de serem encontradas boas soluções. Tais regiões são denominadas pela expressão “regiões promissoras”. A parte genérica do AEH procura por regiões promissoras que, por sua vez, serão exploradas pela parte específica(a heurística).

O grande desafio, na construção de um AEH, está no balanceamento das parcelas de custo computacional empregadas na parte genérica e na parte específica (heurística). Obviamente que o sucesso do AEH em encontrar regiões promissoras facilita a resolução desse desafio.

1.1 Motivação

Mesmo utilizando aleatoriedade nos seus fundamentos, os algoritmos evolutivos se mostram eficientes na busca por soluções ótimas de problemas de difícil solução. A natureza possui diversos processos que se assemelham a esse tipo de busca por estados ótimos. Por isso, o uso da analogia de processos naturais na construção dos AE's é um procedimento muito interessante.

Os AE's possuem grande utilidade em problemas de otimização combinatória, como escalonamento de tarefas, sequenciamento e corte de padrões, caixeiro viajante, assim como em problemas de processamento de imagem, sistemas de classificação, simulações de fenômenos físicos, telecomunicações e em inúmeros problemas das engenharias.

Os trabalhos que utilizam AEH tem mostrado desempenho muito satisfatório. A possibilidade de diminuir o custo computacional, através da identificação de regiões promissoras para busca local, é um grande motivador para o

estudo dessa área.

O uso de heurísticas no auxílio a AE's não é tão simples. Como as heurísticas direcionam o AE para determinados caminhos, outros caminhos podem ser eliminados, inclusive um que levaria a uma região promissora.

Portanto, a seleção de heurísticas e a forma de incorporá-las à arquitetura do AE é um grande desafio nesta área.

1.2 Objetivos

Este trabalho tem como objetivo principal realizar um estudo de caso da utilização de múltiplas heurísticas na estrutura de um Algoritmo Evolutivo. Trata-se de uma extensão do método conhecido como Treinamento Populacional em Heurísticas (TPH), que utiliza os resultados de aplicações de uma função heurística sobre o espaço de soluções para configurar operadores evolutivos e, dessa forma, acelerar a detecção de regiões promissoras. O AE será aplicado sobre o problema de Minimização de Pilhas Abertas, juntamente com uma nova heurística que será desenvolvida para o mesmo.

1.3 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos fundamentais dos Algoritmos Evolutivos e descreve o problema a ser utilizado nos testes da abordagem utilizada. O Capítulo 3 aborda o Treinamento Populacional em Heurísticas, descrevendo seu funcionamento com múltiplas heurísticas. O Capítulo 4 apresenta uma análise dos resultados obtidos ao se aplicar o TPH com múltiplas heurísticas em um problema concreto – o Problema de Minimização de Pilhas Abertas. O Capítulo 5 apresenta as conclusões obtidas.

2 FUNDAMENTOS DOS ALGORITMOS EVOLUTIVOS

Este capítulo descreve o funcionamento dos Algoritmos Evolutivos. As demais seções apresentarão os componentes da arquitetura e conceitos principais de um AE genérico, a saber: codificação das soluções, função de avaliação, mecanismos de seleção, tipos de cruzamento, mutação, gerações, heurísticas e o conjunto de parâmetros associado. O problema de Minimização de Pilhas Abertas também é abordado de forma detalhada na seção final do capítulo. É muito importante lembrar que o desempenho do AE será tanto melhor quanto melhores forem as escolhas dos componentes e parâmetros utilizados.

2.1 Codificação da solução

Escolher a forma como as soluções serão representadas é uma das etapas mais importantes do AE. Cada solução é representada por uma cadeia de símbolos, onde cada símbolo pode assumir um conjunto de valores. Utilizando a analogia com os conceitos da genética, essa cadeia pode ser chamada de cromossomo, ao passo que os símbolos podem ser chamados de genes.

Na representação binária, cada solução (indivíduo) é uma sequência de bits e os operadores evolutivos são aplicados sobre essa sequência. A idéia aqui é facilitar a execução dos operadores evolutivos e, principalmente, abstrair ao máximo os detalhes do problema a ser resolvido. A idéia é a de que o AE pode ser eficiente mesmo sem utilizar conhecimento sobre o problema, ou seja, prima-se pela generalidade do AE.

A falta de correlação direta, na maioria dos problemas, entre a representação binária e a “representação natural” da solução aumenta o custo computacional do AE, pois exige métodos de transformação entre essas representações. Por “representação natural” entenda-se o formato que as soluções devem apresentar para que possam ser aplicadas diretamente no problema original.

Outro inconveniente da representação binária consiste no grande número de soluções inviáveis geradas durante a evolução devido aos limites das variáveis do problema.

Na codificação real utiliza-se uma sequência de números reais (um vetor real) onde cada posição possui um valor limite inferior e outro superior. Esta é uma

representação muito utilizada porque se aproxima da representação natural de muitos problemas e, assim, oferece maior desempenho na maioria dos casos.

Há outras formas de codificar a solução que, dependendo do problema, podem apresentar melhores resultados. Há a codificação inteira, onde os genes são números inteiros; a codificação de string, onde os genes são letras do alfabeto, etc. Mudanças na forma de codificação do indivíduo exigem mudanças na implementação dos operadores evolutivos.

2.2 Função de avaliação

É a função de avaliação *f* que atribui um valor à qualidade da solução. Ela é aplicada sobre o espaço de soluções *S* e pode ser definida como:

$$f : S \rightarrow R \tag{2.1}$$

O espaço de soluções *S* é o conjunto de soluções possíveis para o problema. A função *f* geralmente está relacionada à função objetivo do problema e também é chamada de função de aptidão, sendo que seu valor é denominado o fitness da solução. É ela que faz o papel do meio ambiente, pois é utilizada no processo de seleção para determinar quais indivíduos continuarão na evolução.

Diz-se que quanto melhor o *fitness* de uma solução, mais adaptada ela está à função de avaliação. O fitness indica a proximidade com a solução ótima.

2.3 Mecanismos de seleção

O AE começa sua execução com uma população inicial de indivíduos, formando a geração inicial. As demais gerações são montadas com novos indivíduos criados a partir de operações evolutivas aplicadas sobre essa geração inicial. As operações mais comuns são a seleção, o cruzamento e a mutação. Esta seção trata dos vários tipos de seleção.

A seleção natural das espécies é simulada nos AE's através de mecanismos que realizam a escolha de soluções baseados no seus *fitness*. Dessa forma, há uma competição pela sobrevivência e reprodução, onde os melhores possuem mais vantagem seletiva.

A preferência dada aos melhores indivíduos se dá para que eles tenham maior chance de transmitir suas características para as próximas gerações.

É importante observar que o operador de seleção, toda vez que é executado, não simplesmente coleta o melhor indivíduo da população. O melhor indivíduo apenas possui **maior probabilidade** de ser selecionado. Isso é feito para que indivíduos ruins também tenham chances de serem selecionados, pois, apesar de serem ruins, podem possuir genes de indivíduos bons.

A literatura apresenta vários métodos de seleção. A seguir são apresentados os mais comuns:

2.3.1 Método da roleta

Neste método, a probabilidade de seleção de cada indivíduo é determinada como sendo o resultado da divisão do *fitness* do indivíduo pela soma dos *fitness* de todos os indivíduos da população. O nome “roleta” se dá pela semelhança entre a forma como o indivíduo é escolhido e a a forma como funciona o tradicional jogo da roleta. A Figura 1 ilustra esse funcionamento, que é descrito da seguinte forma:

1. Divide-se uma roleta em n partes, onde n é a quantidade de indivíduos da população.
2. Constrói-se cada parte com um tamanho diretamente proporcional à probabilidade de seleção do indivíduo.
3. Gira-se a roleta.

A posição final do marcador da roleta indica o indivíduo a ser selecionado.

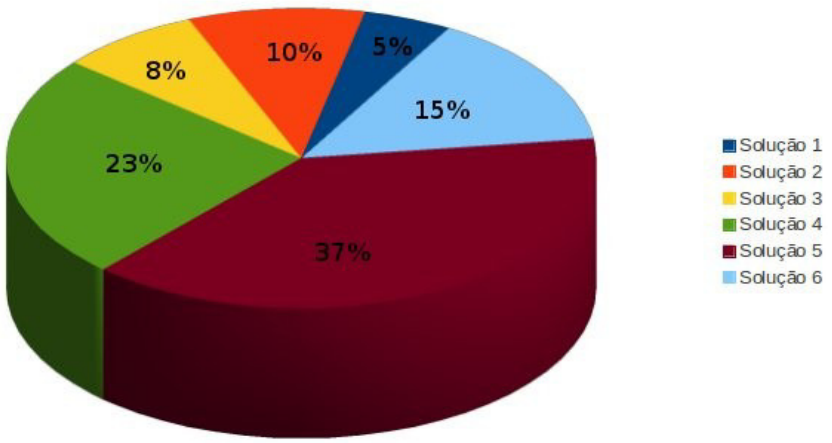

Figura 1 - Roleta com as probabilidades de seleção de cada solução

2.3.2 Método da seleção por torneio

Neste método, o indivíduo é selecionado é o melhor de uma amostra aleatória da população. Ou seja, para cada seleção é necessário coletar aleatoriamente alguns indivíduos.

Outra variante deste método estabelece uma forma de permitir que, em alguma das seleções, o pior indivíduo seja selecionado. Por exemplo:

1. Coleta-se a amostra de p indivíduos;
2. Decide-se entre a escolha do pior ou do melhor, utilizando algum parâmetro contendo a probabilidade de escolha do melhor (ou pior);
3. Seleciona-se o indivíduo com a característica escolhida.

2.3.3 Método da seleção elitista

Neste método, simplesmente seleciona-se o melhor indivíduo da população, que é retirado da geração atual para evitar seleções repetidas. Como se pode observar, é altamente provável que a eliminação dos indivíduos ruins ocorra muito rapidamente e a variabilidade genética da população se torne muito baixa. Esse fenômeno se chama *convergência prematura* e a sua principal consequência é a dificuldade de gerar boas soluções.

A convergência prematura pode ocorrer porque soluções ruins podem possuir genes de soluções boas e, ao serem eliminadas do processo evolutivo, não têm a chance de transmitir suas características aos descendentes. Um dos grandes desafios no projeto de AE's está no controle de convergência da população.

2.4 Mecanismos de cruzamento

Este mecanismo consiste na construção de novas soluções, para próxima geração, a partir de duas ou mais outras soluções da geração atual. A literatura também utiliza os termos *recombinação genética* e *crossover* para se referir ao cruzamento. É este mecanismo que permite que haja troca de partes de soluções, possibilitando assim a criação de indivíduos melhores. Afinal de contas, não adianta apenas identificar o melhor indivíduo, é necessário fornecer meios para produção de indivíduos melhores ainda.

A princípio pode-se pensar que não haverá apenas combinação entre as melhores partes dos indivíduos, mas também entre as partes ruins, entre genes ruins. No entanto, há o processo de seleção que mantém na população uma alta taxa de bons indivíduos. Ou seja, os melhores têm maior probabilidade de participarem do processo de cruzamento e, assim transmitirem suas características para os integrantes da próxima geração.

O algoritmo clássico de cruzamento funciona da seguinte forma: sejam duas soluções s_1 e s_2 representadas por vetores de tamanho n. Escolhe-se uma posição x pertencente ao intervalo [0,n-1]. As novas soluções, s_1' e s_2', resultantes do cruzamento, serão formadas pela justaposição dos *x* primeiros elementos de s_1 e os (n-x) últimos elementos de s_2.

Posição→	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
s_1	2	45	1	17	28	3	7	12
s_2	32	89	80	34	5	15	46	25

s_1'	2	45	1	17	28	3	7	12
s_2'	32	89	80	34	5	15	46	25

Figura 2 - Exemplo de cruzamento entre duas soluções

No cruzamento ilustrado na Figura 2, percebe-se que as soluções foram desmembradas na posição 2. Esta posição é chamada de *ponto de crossover* e, em alguns métodos, é escolhida aleatoriamente. Há algoritmos que utilizam vários *pontos de crossover*.

O cruzamento também pode ser feito com várias soluções, gerando uma ou várias outras soluções. Há também o cruzamento por máscara, em que, a partir de um vetor de bits, escolhidos aleatoriamente ou já definidos, constrói-se a solução final utilizando o seguinte método: se o bit for 1, o gene herdado será o da primeira solução, caso contrário, será o da segunda solução.

2.5 Mecanismos de mutação

A mutação consiste em alterar deliberadamente um ou mais genes de uma solução. É comum utilizar algum processo aleatório para determinar: a quantidade de genes que serão alterados, quais genes serão alterados e o novo valor do gene.

A mutação é importante para que seja possível explorar novas áreas do espaço de soluções, pois muitos pontos (soluções no espaço de soluções) podem não ser formados por recombinações das soluções da geração atual. No entanto, as alterações nos genes frequentemente são baixas, já que não se quer que todo o progresso obtido com a evolução seja desperdiçado.

O parâmetro *taxa de mutação* denomina a probabilidade de ocorrer mutação em um gene. Quanto maior for a taxa de mutação utilizada, maior será a variabilidade genética da população e, por conseguinte, menor será a probabilidade de ocorrer um fenômeno chamado de *convergência prematura*, que consiste na finalização precoce da execução do algoritmo sem encontrar uma solução satisfatória.

2.6 O problema de Minimização de Pilhas Abertas

O problema de Minimização de Pilhas Abertas (Minimization of Open Stacks Problem - MOSP) é comum na indústria e consiste na determinação da melhor permutação de padrões de corte em uma linha de produção.

Cada produto final em uma indústria é composto por um conjunto de peças(ou itens), ou seja, é composto por um padrão. Produtos diferentes possuem padrões diferentes. Para se obter *n* produtos é necessário processar *n* padrões. Cada item, ao ser fabricado, é colocado em uma pilha diferente. Enquanto ainda há itens de determinado tipo a serem produzidos, diz-se que a pilha daquele item está aberta. A solução ótima é a que indica a ordem em que cada padrão deve ser processado para que o máximo de pilhas abertas (mpa) durante a fabricação dos produtos seja o menor possível.

A Tabela 2 ilustra uma linha de produção numa indústria, onde as colunas indicam os produtos(padrões) e as linhas indicam os itens que compõem os produtos. Pode se observar que o Produto 1 é formado apenas por itens do tipo B e

F, enquanto que o Produto 2 é formado por itens D e E, e assim sucessivamente.

Ainda na Tabela 2, observa-se que apenas os produtos 1, 5 e 6 necessitam da fabricação de itens do tipo B. No entanto, a pilha de itens B permanece aberta durante a fabricação dos produtos 1, 2, 3, 4, 5, e 6, ocupando espaço no pátio de armazenamento da fábrica. O objetivo é justamente reduzir ao maximo a necessidade desse espaço, sem limitar a velocidade de produção. Para isso é necessário realocar as colunas da tabela, determinando uma nova ordem de fabricação dos produtos.

Tabela 2 - Instância de um problema de Minimização de Pilhas Abertas

	PRODUTOS									
	1	2	3	4	5	6	7	8	9	10
PEÇA A	0	0	1	0	1	0	1	0	0	0
PEÇA B	1	0	0	0	1	1	0	0	0	0
PEÇA C	0	0	0	0	1	0	0	0	1	1
PEÇA D	0	1	1	0	0	1	0	0	0	1
PEÇA E	0	1	0	1	0	0	0	1	1	0
PEÇA F	1	0	0	0	0	0	1	1	0	0

As células de cor cinza na tabela da Tabela 3 indicam que, naquele momento, a pilha de itens da respectiva linha está aberta. Dessa forma, o número de células de cor cinza em cada coluna indica o número de pilhas abertas naquele momento da produção. Por exemplo, durante o processamento do produto 4 haviam 5 pilhas abertas, mesmo sendo o produto feito de apenas um item. O máximo de pilhas abertas (mpa) é 6, e ocorre durante o processamento dos produtos 5 e 6, como indicado na figura.

Tabela 3 - Quantidades de pilhas abertas

	PRODUTOS									
	1	2	3	4	5	6	7	8	9	10
PEÇA A	0	0	1	0	1	0	1	0	0	0
PEÇA B	1	0	0	0	1	1	0	0	0	0
PEÇA C	0	0	0	0	1	0	0	0	1	1
PEÇA D	0	1	1	0	0	1	0	0	0	1
PEÇA E	0	1	0	1	0	0	0	1	1	0
PEÇA F	1	0	0	0	0	0	1	1	0	0
PILHAS ABERTAS	2	4	5	5	6	6	4	3	2	1
					mpa					

A Tabela 4 mostra um exemplo de rearranjo de padrões que diminui o mpa da instância. No caso, houve apenas uma troca de posição entre as colunas 5 e 8 e, com isso, o mpa diminuiu de 6 para 5.

Tabela 4 - Instância após permutação das colunas 5 e 8

	PRODUTOS									
	1	2	3	4	***8***	6	7	***5***	9	10
PEÇA A	0	0	1	0	***0***	0	1	***1***	0	0
PEÇA B	1	0	0	0	***0***	1	0	***1***	0	0
PEÇA C	0	0	0	0	***0***	0	0	***1***	1	1
PEÇA D	0	1	1	0	***0***	1	0	***0***	0	1
PEÇA E	0	1	0	1	***1***	0	0	***0***	1	0
PEÇA F	1	0	0	0	***1***	0	1	***0***	0	0
PILHAS ABERTAS	2	4	5	5	5	5	4	4	2	1
			mpa							

De acordo com o modelo apresentado, pode-se representar uma instância de um problema MOSP como uma matriz binária *P* de *I* linhas e *J* colunas, $P_{ij} = I \times J$. As linhas representam os itens e as colunas representam os padrões.

P_{ij} = *0, se o item i* ***não ocorre*** *no padrão j*

P_{ij} = *1, se o item i* ***ocorre*** *no padrão j*

Uma determinada permutação de padrões é representada por um vetor com a sequência de números de colunas. Por exemplo, a permutação zero é a apresentada na situação inicial da instância MOSP abordada. Portanto, através da Tabela 2, temos que:

$permuta_0$ = *(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)*

A permutação 1, apresentada na Figura 5, é dada por:

$permuta_1$ = *(1, 2, 3, 4,* ***8****, 6, 7,* ***5****, 9, 10)*

Portanto, dizemos que a permutação 0 possui mpa 6 e a permutação 1 possui mpa 5.

3 TREINAMENTO POPULACIONAL EM HEURÍSTICAS

Em Algoritmos Evolutivos, cada solução do espaço de busca possui um conjunto de informações agregadas, que são utilizadas para direcionar o processo evolutivo. A forma como essas informações são geradas (os cálculos empregados) e o modo como são utilizadas no AE determinam a sobrevivência da solução e de seus futuros descendentes. O uso de heurísticas é um exemplo de aplicação desse procedimento.

O Treinamento Populacional em Heurísticas (TPH) consiste em avaliar a população utilizando não apenas o valor da função objetivo, mas também o seu grau de adaptação a uma ou mais heuristicas. Dessa forma, busca-se nos indivíduos um conjunto extra de características que, mesmo possivelmente não possuindo relação direta com a melhoria do valor da função objetivo, os identifica como pertencentes ou não a regiões promissoras do espaço de busca.

O procedimento utilizado neste trabalho para quantificar o grau de adaptação de um indivíduo a determinada heurística foi desenvolvido por (OLIVEIRA, 2004). O desafio aqui consiste em transpor este procedimento para o uso de múltiplas heurísticas de uma forma que obtenha melhores resultados.

Em (OLIVEIRA, 2004), para quantificar a adaptação de um indivíduo s_k com relação a uma heurística H, deve-se:

a) gerar um conjunto de soluções vizinhas a s_k;

b) avaliar heuristicamente cada uma delas;

c) identificar a melhor, s_m;

d) calcular a distância entre s_k e s_m;

e) calcular a adaptação, considerando a distância entre s_k e s_m.

Como pode ser observado neste método, a adaptação de uma solução à heurística depende das avaliações heurísticas de seus vizinhos. Estas avaliações são realizadas por uma função heurística, que é construída a partir de conhecimento específico do problema.

A literatura apresenta várias formas de se calcular a distância entre duas soluções. Neste trabalho, optou-se por utilizar a diferença entre os valores assumidos pela própria função objetivo.

3.1 Formalização do Treinamento Populacional em Heurísticas (TPH)

O modelo descrito a seguir é a formalização do TPH apresentada por (OLIVEIRA, 2004).

O TPH consiste do seguinte conjunto:

$$TPH = \{ P, \Theta, f, H, p, \delta \} \tag{3.1}$$

O componente P é a população atual de indivíduos s_k, amostrados do espaço de busca S.

O componente Θ é o conjunto de operadores evolutivos utilizados para gerar novas soluções para S.

O componente f é a função objetivo, que estabelece a relação entre S e o conjunto dos números reais, $\mathbb{R}$, ou seja:

$$f : S \rightarrow \mathbb{R} \tag{3.2}$$

O componente H é a heurística de treinamento populacional. É importante lembrar que H é diferente da função heurística, como pode ser observado a seguir:

$$H = \{ \Phi^H, g \} \tag{3.3}$$

O componente g é a função heurística que realiza a nova forma de avaliação de indivíduos e será aplicada sobre os vizinhos de uma solução s_k.

O componente Φ^H é o operador capaz de gerar soluções vizinhas a uma solução s_k, portanto:

$$\Phi^H : S \rightarrow S^v \tag{3.4}$$

onde v é o número de vizinhos de s_k, incluindo ele mesmo. Logo:

$$\Phi^H(s_k) = \{ s_k, s_1, s_2, s_3, s_4, s_5, \ldots, s_{v-1} \} \tag{3.5}$$

Portanto, cada indivíduo s_k possui um valor de avaliação heurística,

determinado pela função g aplicada sobre ele, e também possui um valor de adaptação heurística, que é calculado a partir da avaliação *g* de seus vizinhos.

Neste trabalho, a função *g* será definida como o valor da função *f* aplicado sobre o melhor vizinho, s_m. Logo:

$$g(s_k) = f(s_m) \tag{3.6}$$

A distância entre uma solução s_k e o seu melhor vizinho s_m é denotada por $p(s_k,s_m)$. Como *f* e *g* são funções que possuem os mesmos conjuntos domínio e imagem, a subtração a seguir sempre gerará um valor consistente.

$$p(s_k,s_m) = | f(s_k) - g(s_k) | \tag{3.7}$$

Finalmente, (OLIVEIRA, 2004) define a função δ que estabelece a adaptação total do indivíduo. Para problemas de minimização, o δ é dado por:

$$\delta(s_k) = d \cdot [G_{max} - f(s_k)] - | f(s_k) - g(s_k) | \tag{3.8}$$

O termo G_{max} representa o maior valor que as funções g e f podem assumir durante a execução de determinada instância do AE com TPH. G_{max} pode ser uma constante desde do início do AE ou pode ser determinado por amostragem no início do processo evolutivo e mantido constante a partir de então.

A função δ possui dois componentes. O primeiro, $\{ d \cdot [G_{max} - f(s_k)] \}$, mostra que quanto menor for valor de $f(s_k)$, maior será o valor de $\delta(s_k)$. A constante *d* é utilizada para determinar a parcela de contribuição desse componente para o aumento do valor de $\delta(s_k)$. Geralmente *d* assume um valor no intervalo [0, 1].

O segundo componente da função δ, $\{ - | f(s_k) - g(s_k) | \}$, mostra que quanto maior for a distância entre o valor da função objetivo da solução s_k e o valor da função objetivo da melhor solução vizinha de s_k, menor será o valor de $\delta(s_k)$. Ou seja, este componente avalia a semelhança do indivíduo com seus vizinhos.

Como pode ser observado, a metodologia TPH direciona o AE para a resolução de um problema bi-objetivo, uma vez que visa minimizar a distância $| f(s_k) - g(s_k) |$ e melhorar(maximizar ou minimizar) a função $f(s_k)$.

A Figura 3 mostra um exemplo de cálculo da função δ de uma solução s_k de um problema de minimização. Cada ponto representa uma solução do espaço de busca S. A área cinza destaca o conjunto de soluções vizinhas de s_7, ou seja: $\Phi^H(s_7) = \{ s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10} \}$. A solução vizinha que possui o menor valor de f é a s_5, pois $f(s_5) = 9$. Se considerarmos $G_{max} = 68$, ou seja, o dobro do maior valor de f encontrado na amostra da população indicada na área cinza, e $d = 0.015$, então o valor de $\delta(s_7)$ é dado por:

$$\delta(s_7) = d \cdot [G_{max} - f(s_7)] - | f(s_7) - g(s_7) | \quad (3.9)$$

$$\delta(s_7) = d \cdot [G_{max} - f(s_7)] - | f(s_7) - f(s_5) | \quad (3.10)$$

$$\delta(s_7) = 0.015 \cdot [68 - 32] - | 32 - 9 | \quad (3.11)$$

$$\delta(s_7) = - 22.46 \quad (3.12)$$

Devido à grande distância entre a solução s_7 e a melhor solução, s_5, o delta apresentou um valor muito baixo, significando que s_7 está pouco adaptado à heurística utilizada.

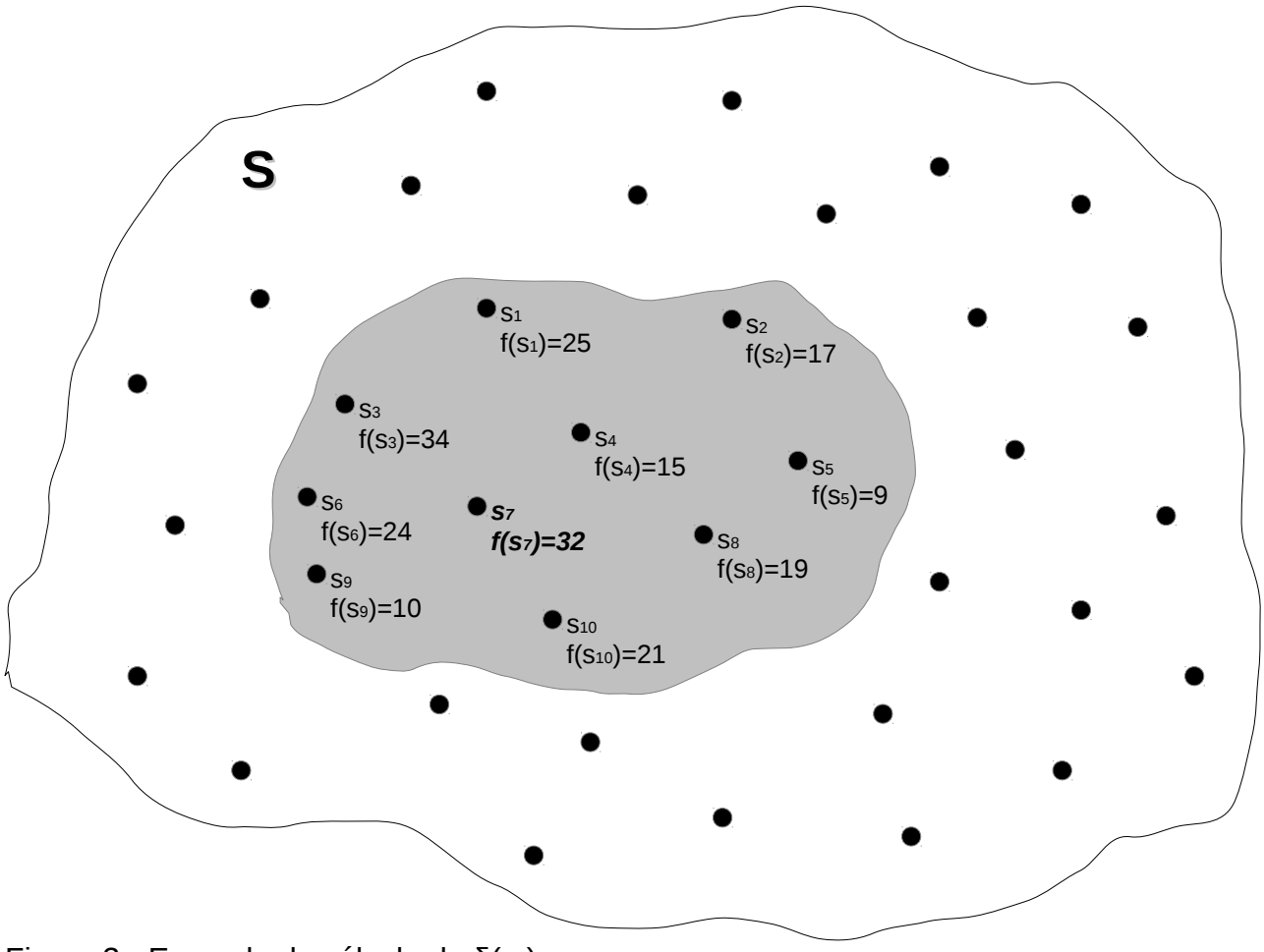

Figura 3 - Exemplo de cálculo de $\delta(s_k)$.

3.2 Extensão da formalização do TPH para múltiplas heurísticas

A formalização do TPH de (OLIVEIRA, 2004), apresentada na seção anterior, faz uso de apenas uma heurística de treinamento no cálculo da adaptação do indivíduo $\delta(s_k)$. No presente trabalho foi realizado um estudo dos efeitos resultantes do uso de mais de uma heurística no referido treinamento. Para isso, foi necessário efetuar uma extensão no modelo formal apresentado anteriormente.

Muitas questões surgem quando se busca uma forma de melhorar o desempenho do TPH incluindo mais heurísticas de treinamento. Algumas destas são apresentadas a seguir:

- A adaptação do indivíduo será definida por qual das heurísticas?
- Forçar o indivíduo a se adaptar igualmente a todas as heurísticas presentes realmente garante resultados melhores?
- Qual heurística em execução é a responsável pelos movimentos em direção às regiões mais promissoras do espaço de busca?
- Há heurísticas se anulando mutuamente, ou seja, realizando movimentos contrários no espaço de busca e, assim, desperdiçando processamento computacional?

(OLIVEIRA, 2004) define três modelos de utilização de múltiplas heurísticas:

a) utilizando uma única população treinada com diferentes heurísticas, competitivamente;
b) utilizando uma única população treinada com diferentes heurísticas, cooperativamente;
c) utilizando múltiplas populações treinadas com diferentes heurísticas, paralelamente;

Neste trabalho será utilizado o modelo de uma única população treinada com diferentes heurísticas de forma cooperativa. As funções heurísticas do conjunto de heurísticas serão aplicadas sobre os indivíduos de forma exclusiva, ou seja, cada indivíduo será treinado com a mesma heurística por toda a sua existência. No momento de sua criação, o indivíduo receberá, aleatoriamente, o tipo de heurística

com a qual será treinada. Dessa forma, mesmo que cada indivíduo possua mais afinidade com determinada heurística, a população em si poderá estar adaptada às múltiplas heurísticas. A população mais adaptada a todas as condições do meio será mais beneficiada no proceso evolutivo.

Do ponto de vista do indivíduo, o caráter múltiplo deste modelo é instigante porque simula com mais realismo o processo evolutivo dos seres vivos. Na natureza, numa determinada região de um ecosistema com grande diversidade, as diversas espécies de seres vivos possuem habilidades diferentes e, com isso, são capazes de se adaptar em maior ou menor grau ao ambiente. No entanto, um conjunto de indivíduos que for altamente desenvolvido em apenas uma habilidade não necessariamente sobreviverá por mais tempo do que outro que possui desempenho médio em várias. Portanto, prevalece de forma mais forte a regra da sobrevivência do que possui maior capacidade de adaptação.

Do ponto de vista do ambiente, sabe-se que seres vivos sempre precisam ser capazes de se adaptar a várias condições ambientais, algumas até antagônicas, como por exemplo o frio do inverno e o calor do verão, ambientes secos em certas épocas do ano e alagados em outras, etc. Somente os que possuem o melhor desempenho médio nas múltiplas avaliações da natureza é que conseguem sobreviver por mais tempo e, assim, gerar mais descendentes.

A formalização do TPH, apresentada anteriormente, será estendida, para incluir este modelo múltiplo, da forma apresentada a seguir.

O componente *H*, a heurística de treinamento populacional, será alterado para:

$$H = \{ \Phi^{H}_{1}, \Phi^{H}_{2}, \Phi^{H}_{3}, \Phi^{H}_{4}, \ldots, \Phi^{H}_{n}, g \} \qquad (3.13)$$

onde Φ^{H}_{k} é uma heurística *k*, de um conjunto contendo *n* heurísticas, cada uma com sua própria forma de gerar soluções vizinhas.

O componente *g*, neste trabalho, é o mesmo para todas as heurísticas. Isto ocorre porque *g* apenas seleciona a melhor solução do conjunto de vizinhos gerado pela heurística.

A adaptação de cada indivíduo será calculada apenas com a sua própria heurística, escolhida no seu nascimento. Assim, a população estará o tempo todo composta por indivíduos adaptados a várias heuristicas de treinamento diferentes.

Há, agora, várias funções delta(δ), uma para cada heurística. Assim, tem-se:

$$\delta_t(s_k) = d \cdot [G_{max} - f(s_k)] \; - \; |f(s_k) - g_i(s_k)| \tag{3.14}$$

onde $g_i(s_k)$ é a aplicação da função heurística g ao conjunto de vizinhos gerado pela heurística Φ^H_i. Ou seja, gera-se um conjunto de vizinhos de s_k com a heurística Φ^H_i, obtendo-se

$$\Phi^H_i(s_k) = \{ s_k, s_1, s_2, s_3, s_4, s_5, \ldots, s_{v-1} \} \tag{3.15}$$

Em seguida, calcula-se o valor da função objetivo f de todas as soluções do conjunto, selecionando o menor valor obtido, que será configurado como o $g_i(s_k)$.

$$g_i(s_k) = f(s_m) \tag{3.16}$$

3.3 Aplicação do Treinamento Populacional em Múltiplas Heurísticas ao problema de Minimização de Pilhas Abertas

Foi construído um algoritmo chamado de *Algoritmo Evolutivo com Treinamento Populacional em Múltiplas Heurísticas (ATPM)* a partir da adaptação do *Algoritmo de Treinamento Populacional (ATP),* desenvolvido por (OLIVEIRA, 2004), a múltiplas heurísticas.

O ATPM é específico para o problema de Minimização de Pilhas Abertas (Minimization of Open Stack Problem - MOSP). As seções seguintes descrevem o seu funcionamento. Inicialmente apenas a estrutura principal do algoritmo é apresentada. Em seguida, o conjunto de heurísticas utilizado é descrito detalhadamente. Por fim, são explicados os aspectos de implementação do código.

3.3.1 Estrutura geral do ATPM

3.3.1.1 A função objetivo

Como explicado na seção 2.6, o problema de minimização de pilhas abertas (MOSP) busca uma permutação de padrões que minimize o número de pilhas abertas na produção de uma fábrica. No entanto, o ATPM foi projetado para

buscar atingir os dois objetivos a seguir:

- minimizar o número máximo de pilhas abertas (mpa);
- minimizar o tempo máximo de pilhas abertas (tpa).

Dessa forma, a função objetivo é determinada por:

$$f(s_k) = I \cdot J \cdot mpa + tpa \tag{3.17}$$

onde *I* denota o número de pilhas (de peças) e *J* denota o número de padrões.

O *tpa* de uma permutação de padrões pode ser melhor entendido através da Tabela 5, que ilustra uma determinada permutação. Como foi anteriormente citado, uma célula cinza indica que a pilha correspondente à respectiva linha ainda está aberta. Dessa forma, o *tpa* da permutação é simplesmente a soma de pilhas abertas em cada momento da produção. Portanto, a permutação da Tabela 5 possui *I = 6, J = 10, mpa = 6* e *tpa = 38.*

Tabela 5 - Exemplo de uma permutação de padrões

	PRODUTOS									
	1	2	3	4	5	6	7	8	9	10
PEÇA A	0	0	1	0	1	0	1	0	0	0
PEÇA B	1	0	0	0	1	1	0	0	0	0
PEÇA C	0	0	0	0	1	0	0	0	1	1
PEÇA D	0	1	1	0	0	1	0	0	0	1
PEÇA E	0	1	0	1	0	0	0	1	1	0
PEÇA F	1	0	0	0	0	0	1	1	0	0
PILHAS ABERTAS	2	4	5	5	6	6	4	3	2	1
					mpa					

Cada sequência de células cinzas em cada linha é denominda *trilha*. Portanto, diz-se que, na Tabela 5, há uma trilha de comprimento 5 na pilha de peças A; há uma trilha de comprimento 8 na pilha de peças E, e assim por diante.

Os termos *I* e *J* foram incluídos no cálculo da função objetivo simplesmente para aplicar um peso maior ao valor do *mpa*, já que a minimização do *tpa* é apenas um objetivo secundário.

3.3.1.2 O controle populacional

O ATPM foi projetado para utilizar população variável. No entanto há duas constantes que determinam os limites populacionais superior e inferior. No início da execução, há uma geração aleatória de soluções até atingir o limite inferior. A partir daí, com a geração inicial completa, começa-se o processo evolutivo.

O controle populacional é feito pelo fator alfa (α), que não tem seu valor alterado durante uma geração, mas apenas na passagem entre gerações. Toda solução precisa ter o seu delta maior do que o alfa da geração atual. Isso determina as duas únicas formas que o algoritmo apresenta para se variar a população:

1. Adicionar uma solução vizinha que apresenta valor de delta (δ) maior que alfa (α), encontrada durante a execução de alguma das heurísticas de treinamento.
2. Eliminar uma solução que apresenta um valor de delta (δ) menor ou igual a alfa (α).

3.3.1.3 O mecanismo de seleção

O mecanismo de seleção do ATPM possui a funcionalidade de coletar dois tipos de soluções: a solução base ou a solução guia. Quando é solicitada uma solução do tipo base, é feita uma escolha aleatória apenas dentre as ρ% melhores soluções da população atual. Já a solução guia é escolhida aleatoriamente na solução inteira.

3.3.1.4 O mecanismo de cruzamento

O mecanismo de cruzamento utilizado pelo ATPM utiliza a solução base e a solução guia, selecionadas previamente, para geração de mais duas soluções que serão inseridas na população se possuírem um bom valor de delta (δ). O genes da solução base serão privilegiados durante o cruzamento. A forma como isso acontece será explicada na seção que abordará os detalhes de implementação.

3.3.1.5 O mecanismo de mutação

Antes de realizar o cruzamento entre os indivíduos base e guia, efetua-se uma sequência de *k* mutações no indivíduo base. Estas mutações seguem o método 2-Opt, que consiste basicamente na busca local, numa vizinhança de tamanho limitado, pelo melhor vizinho gerado a partir de uma árvore de busca, onde cada nó -filho é uma solução que sofreu uma pequena alteração a partir do nó-pai.

O limite da vizinhança na mutação 2-Opt, detalhada na implementação do ATPM, é configurável e foi introduzida no algortimo para evitar o consumo excessivo de processamento.

3.3.2 Heurísticas utilizadas

Foram utilizadas três heurísticas no ATPM: a Heurística 2-Opt, a Heurística de Faggioli e a Heurística MD. Todas são descritas nas seções seguintes. Como poderá ser observado no capítulo 4, foram feitos vários testes com combinações dessas três heurísticas.

3.3.2.1 Heurística 2-Opt

O algoritmo 2-Opt foi proposto primeiramente por (Croes, 1958) e é baseado em trocas entre pares de arestas de grafos que representam soluções para problemas de permutação. O movimento de troca remove duas arestas, quebrando o circuito em dois caminhos, e os reconecta da outra maneira possível (OLIVEIRA, 2004).

O algoritmo 2-Opt, ao ser aplicado sobre uma permutação (uma solução), gera uma nova permutação que pode ser considerada uma solução vizinha. Aplicando-se repetitivamente o algoritmo, mas quebrando as soluções em pontos diferentes, consegue-se gerar várias outras soluções vizinhas. A heurística 2-Opt, portanto, consiste na aplicação sucessiva do Algoritmo 2-Opt para a geração de um conjunto de soluções geneticamente parecidas.

A Figura 4 ilustra o funcionamento do Algoritmo 2-Opt. O primeiro grafo representa uma permutação MOSP de uma solução. Duas arestas são selecionadas, sem repetição, para quebra e troca de pontos de ligação: são as

arestas < 1, 4 > e < 8, 6 >. O segundo grafo mostra a situação final, após reconectar as arestas.

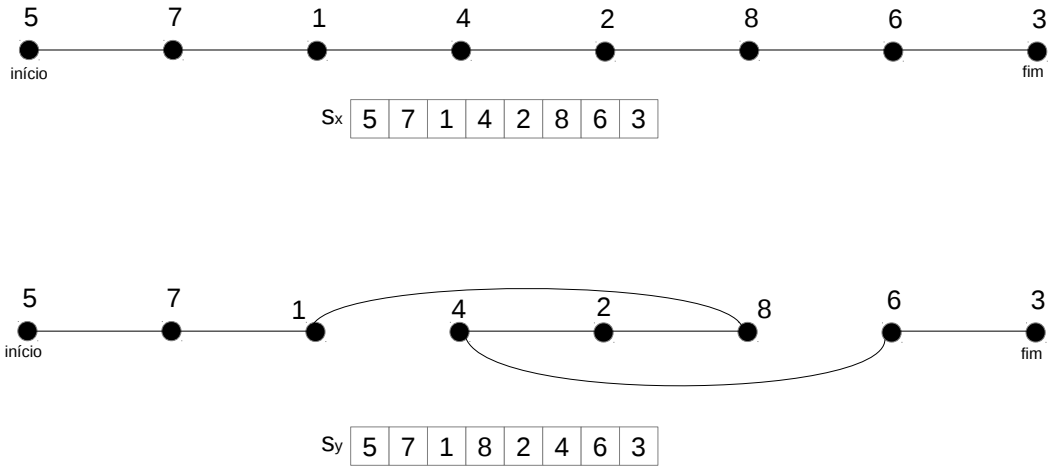

Figura 4 - Exemplo geração de solução vizinha através do algoritmo 2-Opt

O número *v* de vizinhos gerados é configurável no algoritmo. No entanto, deve-se lembrar que haverá uma avaliação de função objetivo para cada vizinho, pois a aplicação da função heurística *g* exige a determinação do melhor vizinho. Portanto, o custo computacional aumenta consideravelmente com o aumento de *v*.

O algoritmo da heurística 2-Opt é definido por (OLIVEIRA, 2004) da seguinte forma:

```
s_m := s_k;
Seja G(s_k) o grafo da solução s_k;
i := gerar_numero_aleatorio( 1, i - 1 );
para( p := i até (i + v - 1)  ) faça
     para( q := (p + 1) até (i + l) ) faça
          Remova as arestas < p, p+1 > e < q, q+1 > de G(s_k);
          Reconecte os vértices do grafo G(s_k) gerando G(s_j);
          se( f(s_j) é melhor que f(s_m) ) então
               s_m :=  s_j;
               g(s_k) := f(s_j);
          fim-se
     fim-para
fim-para
```

Como pode ser observado, o algoritmo testa *v* vizinhos (combinações de arestas) e seleciona a melhor delas. O tempo de execução do algoritmo é polinomial e possui ordem de complexidade $O(n^2)$. (OLIVEIRA, 2004) define o número de avaliações da função objetivo como:

$$numeroAvaliacoes = \frac{n \cdot (n-1)}{2} \tag{3.18}$$

Como cada avaliação da função objetivo no MOSP também é $O(n^2)$, o complexidade total resulta em $O(n^4)$.

O algoritmo mostra que, quando a heurística 2-Opt é aplicada a uma solução, o resultado retornado é apenas o melhor vizinho. Inicialmente a heurística seleciona um ponto aleatório no grafo e, a partir daí, gera o conjunto de pares de arestas que serão trocadas, produzindo todos os vizinhos que serão testados posteriormente.

A Figura 5 ilustra o funcionamento do algoritmo aplicado a uma solução s_k. As arestas são indicadas por apenas um número. Dessa forma, o número 3 indica a aresta <3,4>, o número 7 indica a aresta <7,8>, e assim por diante. Já um par de arestas é definido pelo par ordenado (x,y). Por exemplo, o par ordenado (2,7) indica o par de arestas <2,3> e <7,8>, e assim sucessivamente.

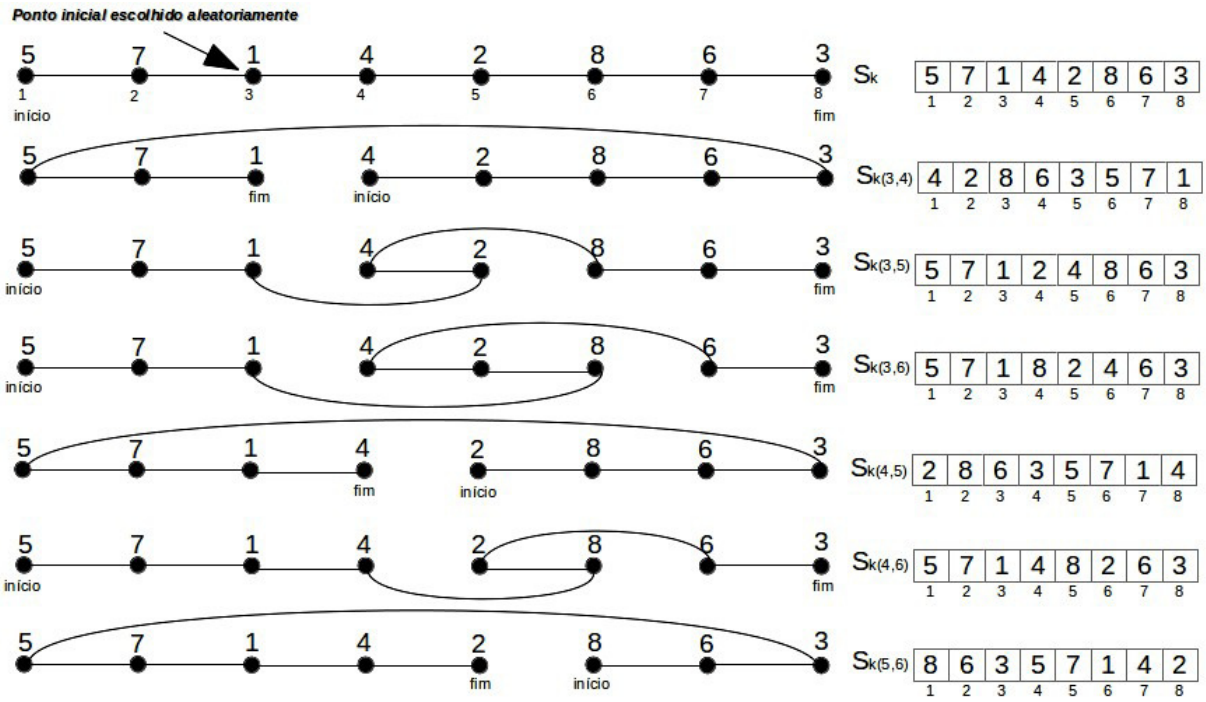

Figura 5 - Exemplo de aplicação do algoritmo 2-Opt

Na Figura 5, observa-se também que, quando duas arestas são consecutivas, a troca é feita de uma forma particular. Isso ocorre nas situações $s_{k(3,4)}$, $s_{k(4,5)}$ e $s_{k(5,6)}$. No exemplo, foram testadas todas as 6 combinações possíveis a partir do ponto inicial. No entanto, no ATPM, como o tamanho da vizinhança pode ser limitado, o custo computacional é controlado.

3.3.2.2 Heurística de *Faggioli-Bentivoglio*

A heurística de *Faggioli-Bentivoglio* consiste no rearranjo dos padrões correspondentes a uma solução através de uma forma de montagem que privilegia as semelhanças entre padrões consecutivos. Trata-se de uma heurística baseada no algoritmo proposto por (Faggioli e Bentivoglio, 1998).

Para montar a permutação final, a heurística inicia adicionando o primeiro padrão e a partir daí escolhe os novos padrões, dentre o conjunto restante, utilizando três critérios.

O primeiro critério estabelece que o padrão escolhido deve ser o que abrir menos pilhas. Uma pilha é aberta quando, na linha da tabela de padrões, houver uma sequência $0 \rightarrow 1$. Este critério parte do pressuposto de que o número de aberturas de pilhas está diretamente ligado ao número máximo de pilhas abertas na permutação completa.

Caso haja empate no número de aberturas de pilhas, utiliza-se o segundo critério, que seleciona o padrão que realizar o fechamento do maior número de pilhas. Esta configuração é identificada através da transição: $1 \rightarrow 0$.

Caso haja empate nos dois primeiros critérios, utiliza-se o terceiro, que estabelece que o padrão a ser escolhido deve ser o que mantiver o maior número de pilhas abertas. Isto ocorre nas transições $0 \rightarrow 0$ e $1 \rightarrow 1$.

A Figura 6 mostra um exemplo de busca do próximo padrão a ser anexado à permutação. À direita da tabela de padrões estão os três padrões candidatos. Pelo primeiro critério, o candidato 1 está eliminado, já que é o padrão que abre o maior número de pilhas. Pelos critérios 2 e 3 tem-se que o Candidato 2 é o padrão que deve ser selecionado.

Caso os candidatos fiquem empatados nos três critérios, um deles é selecionado aleatoriamente.

	PADRÕES		
	3	?	...
PEÇA A	1		
PEÇA B	0		
PEÇA C	0		
PEÇA D	1		
PEÇA E	1		
PEÇA F	0		
PEÇA G	0		

Candidato 1	
0	F
1	A
0	C
1	C
1	C
1	A
1	A
A = 3	
F = 1	
C = 3	

Candidato 2	
0	***F***
0	***C***
0	***C***
0	***F***
1	***C***
1	***A***
1	***A***
A = 2	
F = 2	
C = 3	

Candidato 3	
1	C
1	A
0	C
0	F
1	C
1	A
0	C
A = 2	
F = 1	
C = 4	

Figura 6 - Exemplo de busca de padrão pela heurística Faggioli-Bentivoglio

3.3.2.3 Heurística de Minimização de Danos (MD)

A Heurística de Minimização de Danos foi desenvolvida e utilizada pela primeira vez no presente trabalho. Seu funcionamento consiste em gerar vizinhos de uma solução através de trocas de padrões que possuam a menor probabilidade de causar danos à permutação, ou seja, de gerar uma solução vizinha pior do que a solução atual.

Inicialmente o algoritmo gera uma lista de *r* pares de padrões ordenada de forma decrescente, do melhor par para o pior. O melhor par de padrões é composto pelos padrões que possuírem os maiores números de pilhas abertas. Em seguida, realizam-se as trocas determinadas pelos pares, gerando o conjunto com *r* indivíduos vizinhos.

A Figura 7 exemplifica a construção da lista de pares de padrões a serem trocados. A Tabela Y não representa uma permutação, mas apenas mostra os padrões organizados em ordem decrescente de número de pilhas abertas. A partir desta tabela é montado o conjunto P_t.

O primeiro elemento do conjunto P_t é o par ordenado (6,5), que indica que o padrão 6 trocará de lugar com o padrão 5 na Tabela X, gerando assim uma nova solução: a primeira solução vizinha. Este procedimento é repetido até que se consiga os *r* vizinhos.

Percebe-se, na Figura 7, que a troca sugerida pelo primeiro par já produz um vizinho com tpa mais baixo.

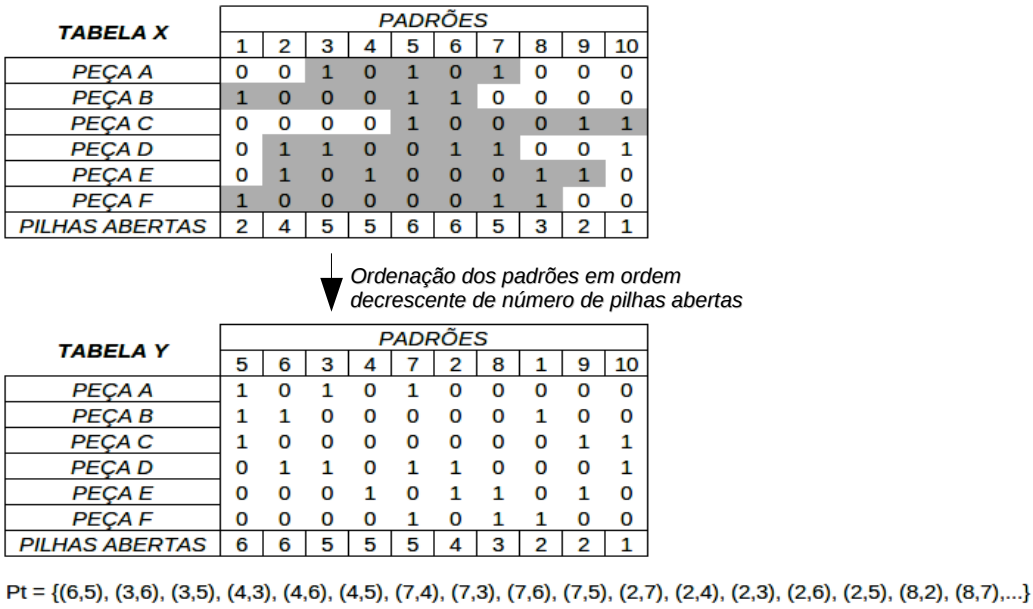

TABELA X

	PADRÕES									
	1	2	3	4	5	6	7	8	9	10
PEÇA A	0	0	1	0	1	0	1	0	0	0
PEÇA B	1	0	0	0	1	1	0	0	0	0
PEÇA C	0	0	0	0	1	0	0	0	1	1
PEÇA D	0	1	1	0	0	1	1	0	0	1
PEÇA E	0	1	0	1	0	0	0	1	1	0
PEÇA F	1	0	0	0	0	0	1	1	0	0
PILHAS ABERTAS	2	4	5	5	6	6	5	3	2	1

Ordenação dos padrões em ordem decrescente de número de pilhas abertas

TABELA Y

	PADRÕES									
	5	6	3	4	7	2	8	1	9	10
PEÇA A	1	0	1	0	1	0	0	0	0	0
PEÇA B	1	1	0	0	0	0	0	1	0	0
PEÇA C	1	0	0	0	0	0	0	0	1	1
PEÇA D	0	1	1	0	1	1	0	0	0	1
PEÇA E	0	0	0	1	0	1	1	0	1	0
PEÇA F	0	0	0	0	1	0	1	1	0	0
PILHAS ABERTAS	6	6	5	5	5	4	3	2	2	1

Pt = {(6,5), (3,6), (3,5), (4,3), (4,6), (4,5), (7,4), (7,3), (7,6), (7,5), (2,7), (2,4), (2,3), (2,6), (2,5), (8,2), (8,7),...}

Figura 7 - Exemplo de geração da lista de pares de trocas de padrões

A heurística MD dá prioridade às trocas entre padrões que possuem as seguintes características:

1. Maior número de pilhas abertas no momento do seu processamento;
2. Menor diferença entre seus números de pilhas abertas.

Os vizinhos gerados são testados e o valor de função objetivo do melhor vizinho é configurado como o *g* da solução s_k.

3.3.3 Detalhes de Implementação do ATPM

Como já mencionado, o ATPM trata-se do ATP (OLIVEIRA, 2004) estendido para o tratamento de múltiplas heurísticas. Esta seção aborda alguns detalhes de implementação importantes para o entendimento do método utilizado.

Inicialmente é construída a geração inicial. É neste momento que se define o valor da constante G_{max}, que será utilizada no restante do código. Ele é definido como aproximadamente o dobro do valor de *f* do pior indivíduo desta geração inicial:

$$G_{max} = 2 * (f_{pior\_indivíduo} + 1) \quad (3.19)$$

O alfa (α) inicial da população é definido como o delta (δ) do pior indivíduo

da população inicial.

O ciclo principal do programa pode ser resumido em quatro procedimentos principais: seleção, mutação, cruzamento e poda da população. Esses procedimentos são repetidos enquanto houver um determinado número mínimo de indivíduos na população (*POPINI*) e também enquanto não tenha sido atingido um número máximo de iterações (*MAXLOOP*).

A seleção das soluções para o cruzamento é feita sempre aos pares: uma solução do tipo base e outra do tipo guia. Apenas a solução base pode sofrer mutação, sendo que a probabilidade disto acontecer é configurável no ATPM, correspondendo à variável *PERC_MUTACAO*.

O algoritmo que realiza a mutação de uma solução s_k, definido por (OLIVEIRA, 2004), consiste na construção de uma árvore de busca com nós filhos gerados a partir de alterações genéticas dos nós pais. No entanto, a cada nível da árvore, apenas o melhor nó gerará mais nós filhos. A Figura 8 ilustra esse processo. O número de níveis da árvore é configurável no ATPM.

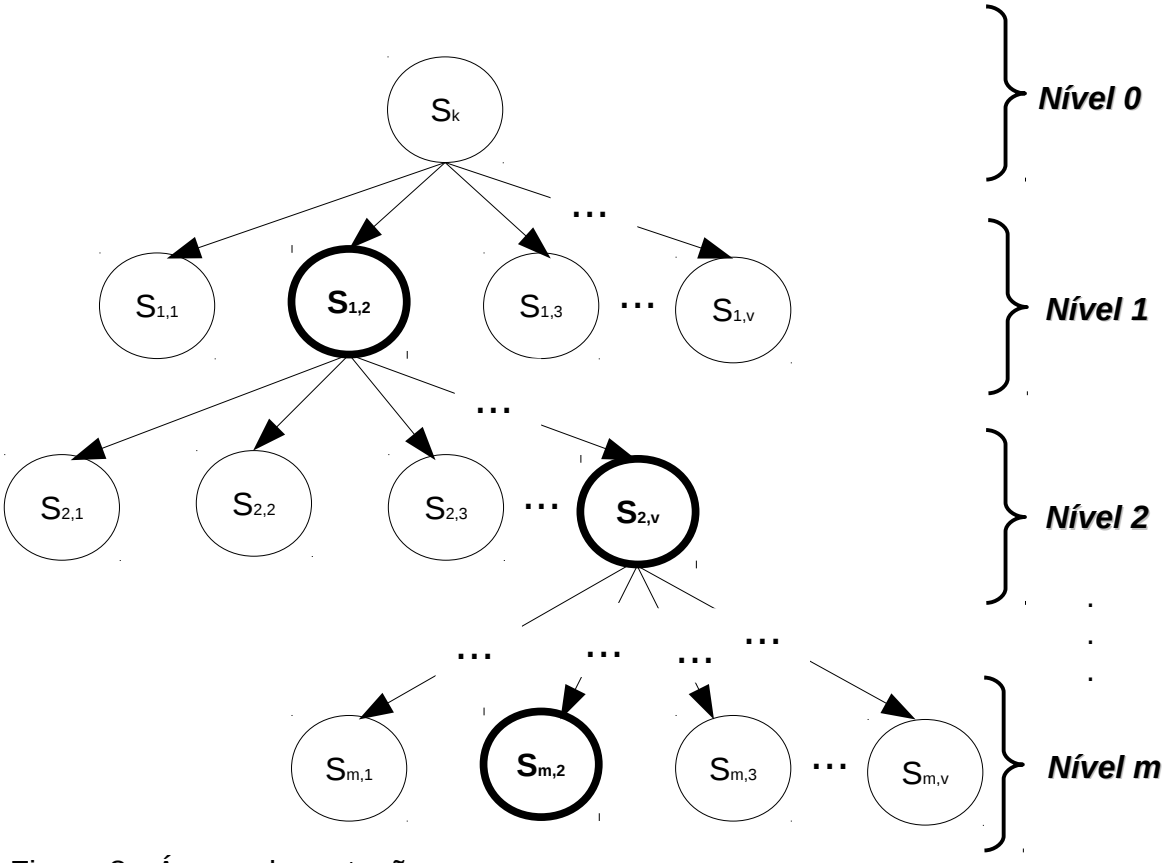

Figura 8 - Árvore de mutação

É importante ressaltar que as mutações somente são feitas em cópias de soluções. A solução original não é alterada e nem retirada da população atual. O indivíduo resultado da mutação só é inserido na população se possuir um valor de delta (δ) maior que o alfa (α) da população atual.

Após a mutação, ocorre o cruzamento entre os indivíduos base e guia, gerando um terceiro indivíduo. Os genes do indivíduo filho são copiados aleatoriamente dos pais.

4 RESULTADOS COMPUTACIONAIS

O Algoritmo de Treinamento Populacional em Múltiplas Heurísticas (ATPM) foi testado com 11 instâncias MOSP encontradas em (OLIVEIRA e LORENA, 2002). Em cada instância foram utilizadas as combinações de heurísticas:

- 2-Opt (2OPT);
- Faggioli-Bentivoglio (FAG);
- Minimização de Danos (MD);
- 2-Opt e Faggioli-Bentivoglio (2OPT + FAG);
- 2-Opt e Minimização de Danos (2OPT + MD);
- Faggioli-Bentivoglio e Minimização de Danos (FAG + MD);
- 2-Opt, Faggioli-Bentivoglio e Minimização de Danos (2OPT + FAG + MD).

O ATPM foi executado 50 vezes com cada combinação de heurísticas. Em cada execução foram realizadas 1000000 (um milhão) de chamadas a função objetivo, exceto nas instâncias mais simples. Assim é possível comparar com fidedignidade os desempenhos das combinações.

Todos os experimentos foram realizados em um computador com processador Intel Pentium i3 1.4 GHz com memória de 4 Gb. As instâncias utilizadas nos testes foram as mesmas que (OLIVEIRA, 2004) utilizou nos tetes do ATP (única heurística), mostradas na Tabela 6.

Tabela 6 - Instâncias utilizadas nos testes do ATPM.

Instância	*Número de Padrões*	*Número de Pilhas*	*Melhor Solução*
wli	10	11	4
v4000	17	10	5
v4050	16	13	4
w1	21	18	4
wsn	25	17	5
v4090	27	23	9
w2	33	48	13
v4470	47	37	9
x0	48	40	11
w3	70	84	18
w4	141	202	27

As demais configurações utilizadas no ATPM são mostradas a seguir:

- constante de equilíbrio, *d = 0.005*;
- tamanho inicial da população, *POPINI = 100*;
- percentual de indivíduos elite, *PERC_MELHORES = 20%*;
- percentual de mutação 2-Opt, *PERC_MUTACAO = 100%*;
- tamanho da vizinhança na heurística 2-Opt, *v = 20*;
- número de vizinhanças 2-Opt avaliadas a cada mutação, *m = 20*;
- número de cruzamentos em cada geração, *NUMCRUZA = 5*;

As tabelas de 7 a 13 mostram os resultados encontrados com a aplicação das combinações de heurísticas citadas. A segunda coluna mostra o mpa médio das 50 execuções. A coluna *Taxa de Acerto* indica a quantidade de vezes que o *mpa mínimo* foi encontrado nas 50 execuções. A coluna *Gap* indica a diferença entre o melhor resultado encontrado e a melhor solução conhecida para a referida instância, apresentada na Tabela 6.

Tabela 7 - Resultados dos testes realizados com a heurística 2OPT.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13	0.00	13	100.00	0.00
v4470	9.49	0.06	9	94.00	0.00
x0	11	0.00	11	100.00	0.00
w3	23	1.00	22	2.00	22.22
w4	37.78	4.77	33	4.00	22.22
Média	11.57	0.53	11	81.82	8.13

As instâncias estão dispostas em ordem crescente de tamanho de arquivo. As taxas de acerto das sete primeiras, as mais simples, quase sempre são de 100%.

A Tabela 8 mostra os testes com a heurística de Faggioli-Bentivoglio (FAG). Trata-se de uma das heurísticas mais eficientes.

Tabela 8 - Resultados dos testes realizados com a heurística FAG.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13	0.00	13	100.00	0.00
v4470	9	0.00	9	100.00	0.00
x0	11	0.00	11	100.00	0.00
w3	22.98	0.95	22	6.00	22.22
w4	39.48	3.28	34	6.00	25.93
Média	11.68	0.38	11.09	82.91	8.47

A Tabela 9 apresenta os testes com a heurística de Minimização de Danos (MD), a nova heurística desenvolvida neste trabalho.

Tabela 9 - Resultados dos testes realizados com a heurística MD.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13	0.00	13	100.00	0.00
v4470	9	0.00	9	100.00	0.00
x0	11	0.00	11	100.00	0.00
w3	23.08	0.99	22	8.00	22.22
w4	41.14	6.07	35	2.00	29.63
Média	11.84	0.64	11.18	82.73	8.80

A Tabela 10 mostra os resultados do ATPM com a combinação das heurísticas 2-Opt e Faggioli-Bentivoglio. Percebe-se que há um efeito inédito nas instâncias v4090 e v4470: uma pequena diminuição da taxa de acerto.

Tabela 10 - Resultados dos testes realizados com as heurísticas 2OPT e FAG.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13.02	0.02	13	98.00	0.00
v4470	9.04	0.04	9	96.00	0.00
x0	11	0.00	11	100.00	0.00
w3	23.04	0.97	22	8.00	22.22
w4	37.82	2.85	35	10.00	29.63
Média	11.54	0.35	11.18	82.91	8.80

A Tabela 11 mostra os resultados da combinação das heurísticas 2-Opt e Minimização de Danos. O destaque aqui é a diminuição da taxa de acerto em muitas instâncias: w2, v4470, x0, w3 e w4.

Tabela 11 - Resultados dos testes realizados com as heurísticas 2OPT e MD.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13.13	1.07	13	88.00	0.00
v4470	9.10	0.10	9	90.00	0.00
x0	11.08	0.08	11	94.00	0.00
w3	23.14	0.54	22	4.00	22.22
w4	41.66	3.79	36	2.00	33.33
Média	11.92	0.51	11.27	79.82	9.14

As Tabelas 12 e 13 mostram os resultados dos testes com as demais combinações.

Tabela 12 - Resultados dos testes realizados com as heurísticas FAG e MD.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13	1.02	13	90.00	0.00
v4470	9.02	0.02	9	98.00	0.00
x0	11.06	0.06	11	94.00	0.00
w3	23	0.98	22	4.00	22.22
w4	39.2	3.22	35	4.00	29.63
Média	11.66	0.48	11.18	80.91	8.80

Tabela 13 - Resultados dos testes realizados com as heurísticas 2OPT, FAG e MD.

Instâncias	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
wli	4	0.00	4	100.00	0.00
v4000	6	0.00	6	100.00	20.00
v4050	4	0.00	4	100.00	0.00
w1	5	0.00	5	100.00	25.00
wsn	5	0.00	5	100.00	0.00
v4090	9	0.00	9	100.00	0.00
w2	13.08	0.08	13	92.00	0.00
v4470	9.12	0.17	9	88.00	0.00
x0	11	0.00	11	100.00	0.00
w3	23.06	0.99	22	2.00	22.22
w4	38.98	4.88	33	2.00	22.22
Média	11.66	0.56	11.00	80.36	8.13

A Tabela 14 permite comparar as eficiências de todas as combinações de heurísticas. Pode-se concluir que a mais eficiente é a (2OPT+FAG), que apresentou o maior mpa médio (11.54), o menor desvio padrão (0.35) e a maior taxa de acerto (82.91%).

No entanto, também deve-se considerar a combinação das três heurísticas (2OPT+FAG+MD), que apresenta o mais baixo Gap médio (8.13) e o mais baixo mpa mínimo.

Tabela 14 - Comparativo entre os resultados das combinações de heurísticas

Heurísticas	***mpa médio***	***Desvio padrão***	***mpa mínimo***	***Taxa de acerto(%)***	***Gap (%)***
2OPT	11.57	0.53	11.00	81.82	8.13
FAG	11.68	0.38	11.09	82.91	8.47
MD	11.84	0.64	11.18	82.73	8.80
2OPT + FAG	**11.54**	**0.35**	11.18	**82.91**	8.80
2OPT + MD	11.92	0.51	11.27	79.82	9.14
FAG + MD	11.66	0.48	11.18	80.91	8.80
2OPT + FAG + MD	11.66	0.56	11.00	80.36	8.13
Média	11.66	0.49	11.13	81.64	8.61

5 CONSIDERAÇÕES FINAIS

O Treinamento Populacional em Heurísticas (TPH), ao utilizar informações sobre o problema para avaliar soluções, estabelece uma nova forma de controle do processo evolutivo. Não somente a função objetivo é usada para avaliar a qualidade de uma solução, mas também a sua adaptação a funções heurísticas. O termo treinamento consiste no processo de forçar a população a realizar essa adaptação.

Os teste realizados e apresentados no capítulo anterior permitem afirmar que o uso de múltiplas heurísticas no treinamento populacional em Algoritmos Evolutivos se mostrou competitivo com a abordagem tradicional. Portanto, a principal contribuição deste trabalho consiste na comparação entre as eficiências do AE quando sua população é treinada com uma, duas e/ou três heurísticas, especificamente as heurísticas 2-Opt, de Faggioli-Bentivoglio e de Minimização de Danos.

A heurística de Minimização de Danos (MD), inédita e desenvolvida neste trabalho, quando utilizada isolada, apresentou desempenho abaixo da média em todos os testes. No entanto, em conjunto com as heurísticas 2-Opt e/ou Faggioli-Bentivoglio, a MD se apresentou muito competitiva.

REFERÊNCIAS

AGUIAR, M. **Análise Formal da Complexidade de Algoritmos Genéticos**. 1998. 75 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1998.

FECHINE, Joseana Macêdo. **Algoritmos Genéticos e suas Aplicações**. In: Ciclo de Seminários Técnicos - CST, 2011, Campina Grande, 2011.

FERREIRA, T. L.; LEANDRO, G. V.; CAMARGO, R. F.; DILL, S. L.; PADOIN, E. L. **Execução Paralela do Algoritmo Genético Aplicado na Estimação de Parâmetros em Cluster Beowolf**. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, 2009, Caxias do Sul. Anais... São Paulo, 2009. p. 181-184.

LINDEN, R. **Algoritmos Genéticos - Uma importante ferramenta da Inteligência Computacional**. 2. ed. Rio de Janeiro: Brasport, 2008. 400 p.

CHAVES, A. A.; LORENA, L. A. N. **Clustering Search Algorithm for the Capacitated Centred Clustering Problem**. Laboratory of Computing and Applied Mathematics, INPE. São José dos Campos, 2009.

OLIVEIRA, A. C. M. **Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos**. 2004. 200 f. Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 2004. p. 35-58.

OLIVEIRA, A. C. M. **Algoritmos Evolutivos para Problemas de Otimização Numérica com Variáveis Reais** – Monografia de Exame de Qualificação – CAP/INPE, 2001. Disponível em http://www.lac.inpe.br/~lorena/monografia-alexandre.pdf.

OLIVEIRA, A. C. M.; LORENA, L. A. N. **Detecting promising areas by evolutionary clustering search**. Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series, 2004; 385-394.

ZUBEN, F. J. V. **Computação Evolutiva: Uma Abordagem Pragmática**. DCA/FEEC/Unicamp, Campinas. 2002.