Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

THIAGO DE SOUSA PEREIRA

TREINAMENTO POPULACIONAL EM MÚLTIPLAS HEURÍSTICAS APLICADO A PROBLEMAS DE SEQUENCIAMENTO DE PADRÕES

São Luís
2016

THIAGO DE SOUSA PEREIRA

TREINAMENTO POPULACIONAL EM MÚLTIPLAS HEURÍSTICAS APLICADO A PROBLEMAS DE SEQUENCIAMENTO DE PADRÕES

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: Profº Alexandre César Muniz de Oliveira

São Luís

2016

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Pereira, Thiago de Sousa.
Treinamento Populacional em Múltiplas Heurísticas aplicado a Problemas de Sequenciamento de Padrões / Thiago de Sousa Pereira. – 2016.
43 f.

Orientador(a): Alexandre César Muniz de Oliveira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2016.

1. Algoritmo Genético. 2. Sequenciamento de Padrões. 3. Treinamento Populacional em Heurísticas. I. Oliveira, Alexandre César Muniz de. II. Título.

Thiago de Sousa Pereira

Treinamento Populacional em Múltiplas Heurísticas aplicado a Problemas de Sequenciamento de Padrões

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.

Trabalho aprovado. São Luís, 15 de setembro de 2016:

Prof. Dr. Alexandre César Muniz de Oliveira
Orientador
Universidade Federal do Maranhão

Prof. Dr. Luciano Reis Coutinho
Universidade Federal do Maranhão

Prof. Dr. Geraldo Braz Júnior
Universidade Federal do Maranhão

São Luís
2016

Este trabalho é dedicado às futuras inteligências artificiais, que um dia dominarão o mundo.

Agradecimentos

Agradeço à minha família, pelo suporte que me deram ao longo dos anos. Aos meus amigos, por tornarem possível diversão em meio aos trabalhos. Aos professores que me ensinaram além do que imaginei um dia poder aprender.

"Só quem se liberta compreende a prisão."
– Eram Os Deuses Astronautas? (Um Barril de Rap)

Resumo

O Treinamento Populacional em Heurísticas é uma metaheurística de busca que utiliza uma heurística de treinamento como uma segunda avaliação dos indivíduos a fim de detectar regiões promissoras do espaço de busca mais rapidamente. Este trabalho introduz o uso de mais de uma heurística em conjunto de três formas: competitiva, cooperativa e em paralelo. Os experimentos foram realizados utilizando Problemas de Sequenciamento de Padrões. Mais especificamente, Problema de Minimização de Pilhas Abertas (MOSP) e o Problema de Leiaute de Matriz-Porta (GMLP). A proposta cooperativa se mostrou promissora, tendo atingido o platô do valor médio mais rapidamente que as outras metodologias com único processo. A metodologia em paralelo também se mostrou promissora ao encontrar o ótimo mais consistentemente em instâncias pequenas e médias e ter se aproximado mais do ótimo conhecido em instâncias grandes.

Palavras-chaves: Sequenciamento de padrões; 2-Opt; Treinamento Populacional em Heurísticas; Algoritmo Genético; Fagiolli-Bentivoglio

Abstract

Population Training Heuristics is a search metaheuristic that uses a training heuristic to provide a second evaluation of the individuals in order to detect promising regions of the search space faster. This work introduces the use of more than one heuristic together in three ways: competitive, cooperative, and in parallel. The experiments were made using Pattern Sequencing Problems. More specifically: Minimization of Open Stacks Problem and Gate Matrix Layout Problem. The cooperative methodology proved to be promising, having reached the average median value plateau faster than the others. The parallel methodology also proved to be promising, finding the optimal value more consistently in small and medium-sized instances and having the average closer to the known optimum in large ones.

Key-words: Pattern Sequencing; 2-Opt; Population Training Heuristics; Genetic Algorithm; Fagiolli-Bentivoglio

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Apesar de métodos exatos existirem, como o algoritmo Simplex (DANTZIG, 1951), grande parte dos problemas de otimização encontrados na prática são computacionalmente intratáveis: os problemas conhecidos como NP-árduos (NP-Hard) por exemplo. A tarefa de computar uma solução ótima para esses casos é reduzida a encontrar uma solução boa, apenas. Para realizar tal tarefa, existem as metaheurísticas. Metaheurísticas são algoritmos heurísticos genéricos que se aplicam a uma grande quantidade de problemas de otimização. O termo heurística vem do fato de esses algoritmos não serem exatos, mas sim voltados para encontrar ótimos locais que se aproximam do ótimo global.

Diversas metaheurísticas já foram desenvolvidas com várias características diferentes. É comum, entretanto, dividi-las em dois grandes grupos: as de solução-única e as baseadas em população.

As metaheurísticas de solução-única partem de uma solução inicial e trabalham para melhorar (refinar) esta solução. Sendo assim, diz-se que este tipo algoritmo tem grande poder de intensificação (*exploitation*). As baseadas em população são poderosas quando se trata de explorar (*exploration*) todo o espaço de busca, apesar de terem dificuldades de refinar localmente uma solução depois que encontram uma região promissora.

Uma ideia a ser estudada é a combinação de mais de uma metaheurística para resolver um mesmo problema de otimização. Vários tipos de combinação foram propostos na literatura e esses algoritmos combinados são chamados de híbridos (TALBI, 2002).

A hibridização de metaheurísticas tem se provado um excelente amplificador do poder de otimização de muitos algoritmos. Especialmente quando se utiliza otimizadores locais (refinamento de uma solução – intensificação) para compensar o baixo desempenho desse aspecto dos algoritmos baseados em população. Estes têm grande poder de estimar o ótimo global e encontrar rapidamente regiões promissoras do espaço de busca, mas costumam falhar quando a diferença de desempenho entre os indivíduos torna-se muito pequena. Quando a aptidão (*fitness*) dos indivíduos de uma população varia muito pouco de uma geração para a próxima, no caso de algoritmos genéticos, por exemplo.

Uma hibridização que foi proposta em (OLIVEIRA, 2004) é o treinamento populacional em heurísticas. Nesta abordagem, heurísticas de otimização local são utilizadas para avaliar regiões promissoras e encontrá-las mais rapidamente dentro do espaço de busca de um algoritmo génetico. Depois de encontradas, um mecanismo de intensificação é utilizado para explorar a região encontrada. Duas heurísticas de treinamento foram utilizadas separadamente: 2-Opt (CROES, 1958) e Faggioli-Bentivoglio (FAGGIOLI; BENTIVOGLIO, 1998).

1.1 Objetivos

- Investigar o emprego de múltiplas heurísticas de treinamento para o algoritmo de treinamento populacional aplicado a problemas de sequenciamento de padrões;
- Propor, implementar e avaliar tal algoritmo capaz de utilizar múltiplos treinamentos populacionais visando ampliar a capacidade de exploração tanto global quanto local do espaço de busca;
- Contribuir com novas visões e abordagens para a metaheurística genética utilizando heurísticas híbridas de treinamento populacional;
- Desenvolver e incentivar pesquisas na área de treinamento populacional em heurísticas.

1.2 Organização do trabalho

No Capítulo 2, é exposta uma explicação do tipo de problema que será atacado (Problemas de sequenciamento de padrões), que são o MOSP e GMLP, e também sua importância para a indústria.

Uma breve introdução sobre Computação Evolutiva é feita, bem como um exemplo de Algoritmo Genético simples, que é base para o algoritmo utilizado neste trabalho.

Na Seção 2.3, a introdução do conceito de treinamento populacional é feita, assim como a explicação das duas heurísticas utilizadas e do algoritmo que utilizada tais heurísticas.

O Capítulo 3 introduz o conceito de múltiplas heurísticas de treinamento e detalha as três abordagens que foram propostas e serão utilizadas.

No capítulo 4, são apresentados resultados experimentais realizados com base no que foi proposto no Capítulo 3. Estes resultados também são discutidos e avaliados aqui.

Por fim, o Capítulo 5 apresenta a conclusão e direcionamentos para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, os principais temas abordados por este trabalho serão explicados. Primeiramente, os problemas que serão estudados são definidos (Problemas de Sequenciamento de Padrões), bem como sua importância na indústria e dificuldades na sua resolução. Em seguida, uma breve explicação de Computação Evolutiva é feita, tendo como foco os Algoritmos Genéticos. Por último, o Treinamento Populacional em Heurísticas é definido, assim como as duas heurísticas que serão utilizadas.

2.1 Problemas de sequenciamento de padrões

Problemas de sequenciamento de padrões (PSP) consistem em encontrar uma permutação de linhas de uma matriz que otimize uma determinada função objetivo. Este tipo de problema tem importantes aplicações, principalmente na área de planejamento de produção (FINK; VOß, 1999).

Consideremos, por exemplo, um sistema em que a fabricação de certo produto é iniciada a partir de um pedido de um cliente. Cada produto, entretanto, é composto de diversos itens. Quer-se, então, encontrar a sequência de produção de itens que minimiza a quantidade de pedidos em aberto (YANASSE, 1997). Cada produto aqui representa uma linha da matriz do problema em questão e cada elemento destas linhas (cada coluna) representa um item que compõe o produto final. Este conjunto de itens, ou seja, uma linha completa, é um padrão.

Ter muitos pedidos não finalizados implica em maiores custos de manutenção de inventário, de transporte, atrasos nas entregas para os clientes, atraso nos pagamentos etc. Sendo assim, dada uma lista de pedidos de clientes, a ordem ou sequência em que estes pedidos (padrões) devem ser produzidos a fim de minimizar tais custos é a preocupação dos PSP.

Um exemplo específico deste tipo de problema é a indústria de vidro mencionada em YUEN (1991). Uma peça de vidro precisa ser cortada em peças menores (padrões). Quando os cortes são iniciados, pilhas são abertas para acomodar os pedaços de vidro que foram requisitados. Estas pilhas permanecem abertas até que todos os cortes daquele tipo sejam feitos. Isso pode se tornar um problema caso seja necessária movimentação provisória de pilhas ainda abertas e espaço disponível nos galpões de produção. Novamente, vê-se a necessidade de escolher a melhor sequência de padrões a serem cortados a fim de minimizar o número de pilhas abertas.

Este exemplo específico é o principal problema a ser tratado neste trabalho. Em

YANASSE (1997) trata-se de cortes em madeira, mas a ideia se mantém. O problema de minimização de pilhas abertas, chamado de MOSP (*Minimization of Open Stacks Problem*), será detalhado em seguida.

2.1.1 Minimização de pilhas abertas

O exemplo de cortes em madeira será utilizado para definir o problema. Seja I o conjunto que representa as peças inteiras de madeira (padrões). Sendo assim, $i \in I$ é um padrão deste conjunto. Cada padrão i pode ser cortado em diferentes pedaços (itens). O conjunto J contém todos os possíveis tipos de item em que os padrões podem ser cortados: $j \in J$. Um padrão i pode ou não ser cortado de forma que produza j.

A relação entre padrões e itens é representada por uma matriz $I \times J$. Esta matriz $P = \{p_{ij}\}$ define quais padrões produzem quais itens. Se i puder produzir j, $p_{ij} = 1$, e $p_{ij} = 0$ caso contrário. Não existem cortes parciais de padrões, entretanto. Cada padrão i é cortado totalmente até que se tenha todos os itens que ele pode produzir.

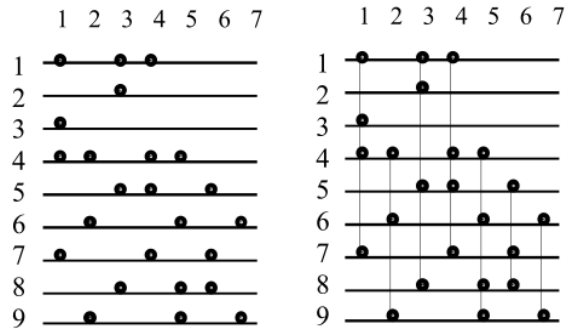
Assim que um item é produzido a partir de um corte de um padrão, uma pilha é iniciada para conter todos os itens daquele mesmo tipo. Esta pilha só é fechada quando o último padrão capaz de produzir aquele tipo de item é cortado. Fechar a pilha aqui significa removê-la do pátio de produção. Fica mais claro, agora, o motivo por trás de minimizar o número máximo de pilhas abertas (*mpa*): quanto maior este número, maior precisa ser o pátio para armazenagem, pilhas subsequentes ficam cada vez mais longe da máquina de corte, pode-se precisar mover temporariamente certas pilhas, o tempo de produção aumenta e também os riscos associados ao manuseio (LINHARES; YANASSE, 2002).

Seja π uma permutação qualquer de linhas da matriz P e $\pi(i)$ a posição da linha i em P. Ou seja, se $\pi(i) = 3$, o padrão i será o terceiro a ser cortado. Define-se, então, uma matriz auxiliar $Q^\pi = \{q^\pi_{ij}\}$ que é utilizada para calcular o *mpa* de uma determinada permutação π:

- $q^\pi_{ij} = 1$, caso $\exists x, y \mid \pi(x) \leq i \leq \pi(y)$ *e* $P_{xj} = P_{yj} = 1$
- $q^\pi_{ij} = 0$, caso contrário

Na Figura 1, a matriz à esquerda é P. Cada ponto na figura marca a presença do valor 1 na matriz. O padrão 4, por exemplo, produz os itens 1, 2, 4, 5, como pode-se ver. A matriz Q^π, à direita, inicialmente é uma cópia de P. As linhas traçadas são os 1s adicionais inseridos na matriz. Estas linhas representam o tempo de vida da pilha de cada tipo de item. A pilha do item 5, por exemplo, é aberta quando o padrão 4 é cortado, e só é fechada quando o padrão 9 é cortado.

Figura 1 – Construção de Q^{π} (à direita) a partir de P (à esquerda).



Fonte: OLIVEIRA (2004)

A linha de Q^{π} que contiver mais pilhas abertas em um dado instante dá o valor do máximo de pilhas abertas (*mpa*). Formalmente:

$$mpa(\pi) = \max_{i \in I} \sum_{j=1}^{J} q_{ij}^{\pi} \tag{2.1}$$

No exemplo, ao cortar os padrões 6 e 7, o número de pilhas abertas é máximo: $mpa = 7$. Mudando a ordem em que os padrões são cortados, diferentes valores de *mpa* podem ser encontrados. O objetivo do MOSP é encontrar a permutação π desses padrões que minimiza o valor de *mpa* (Γ é o conjunto de todas as possíveis permutações):

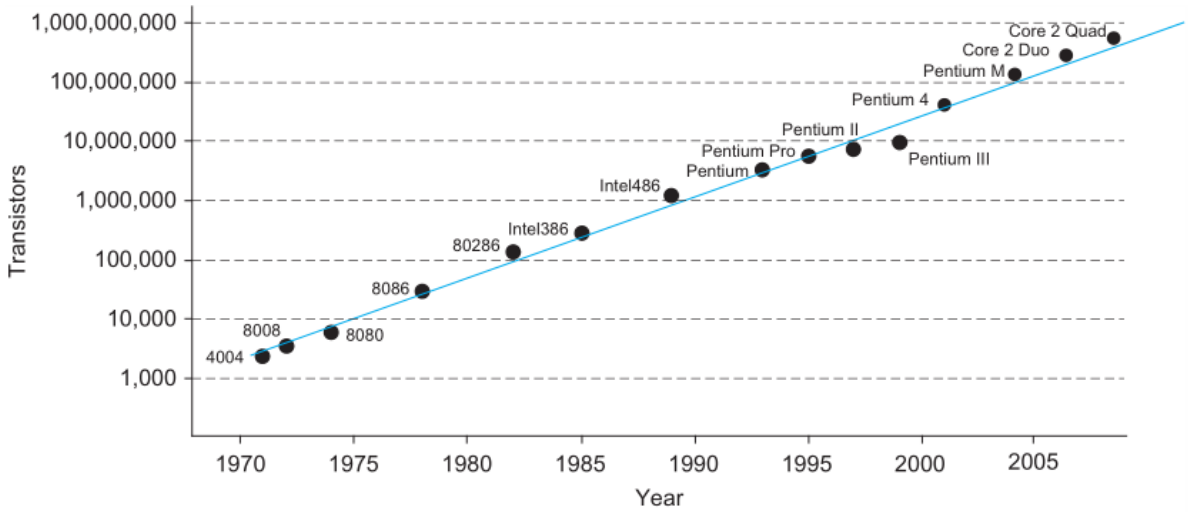
$$Z_{MOSP} = \min_{\pi \in \Gamma} (mpa(\pi)) \tag{2.2}$$

2.1.2 Problema de leiaute de Matriz-Porta

Desde meados de 1980 até os dias de hoje, o nível de miniaturização dos chips de computador encontra-se no chamado VLSI (do inglês, Very Large-Scale Integration). Esta nomenclatura está relacionada à quantidade de transístores contidas em um único chip (WESTE; HARRIS, 2011). O aumento desta quantidade ao longo dos anos foi teorizada por Moore em 1965. Ele observou que o número comercialmente viável de transístores seguia uma escala semi-logarítmica e conjecturou que este número deveria dobrar a cada 18 meses. Tal afirmação se tornou uma meta para as fabricantes (MOORE, 1965). O crescimento do número de transístores ao longo dos anos pode ser observado no gráfico da Figura 2.

Após o projeto lógico de um sistema VLSI, é necessário encontrar a melhor forma de dispor os transístores fisicamente. Esta necessidade dá origem ao problema de leiaute de Matriz-Porta, ou GMLP (do inglês, *Gate Matrix Layout Problem*). O crescimento

Figura 2 – Número de transístores em processadores Intel.

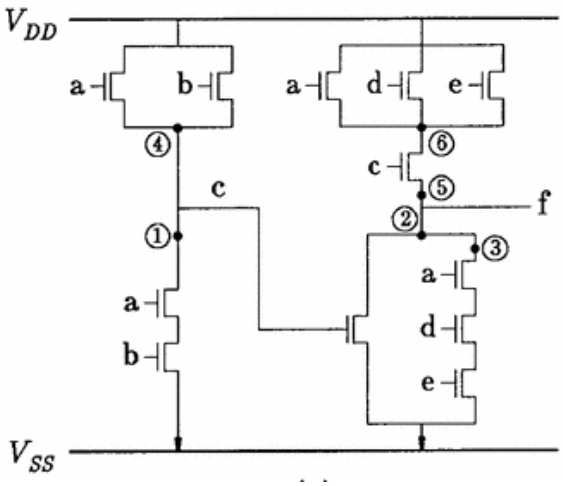


Fonte: WESTE; HARRIS (2011)

previsto por Moore tornou este problema ainda mais importante. Isto porque milhões de transístores precisam ser empacotados eficientemente.

O projeto lógico inicial é exemplificado na Figura 3 e um possível resultado de disposição do circuito pode ser visto na Figura 4. Este tipo de leiaute foi introduzido em LOPEZ; LAW (1980) e será brevemente explicado à seguir, tendo como base a definição de MOSCATO; MENDES; LINHARES (2004).

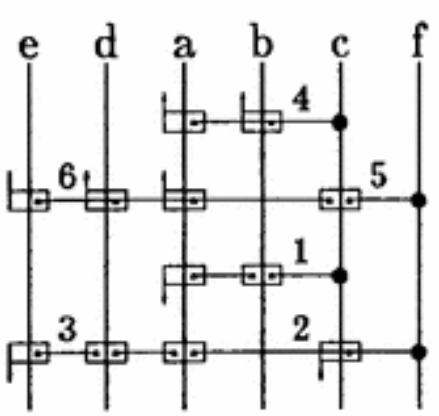
Figura 3 – Projeto lógico de um sistema VLSI.



Fonte: WONG; LEONG; LIU (1988)

Um GMLP pode ser representado como uma matriz binária com g colunas e n linhas. As colunas serão chamadas de portas (ou *gates*) e as linhas de redes (ou *nets*). Um valor 1 na posição (i, j) significa que um transístor deve ser implementado na interseção

Figura 4 – Projeto físico de um sistema VLSI no formato Matriz-Porta.



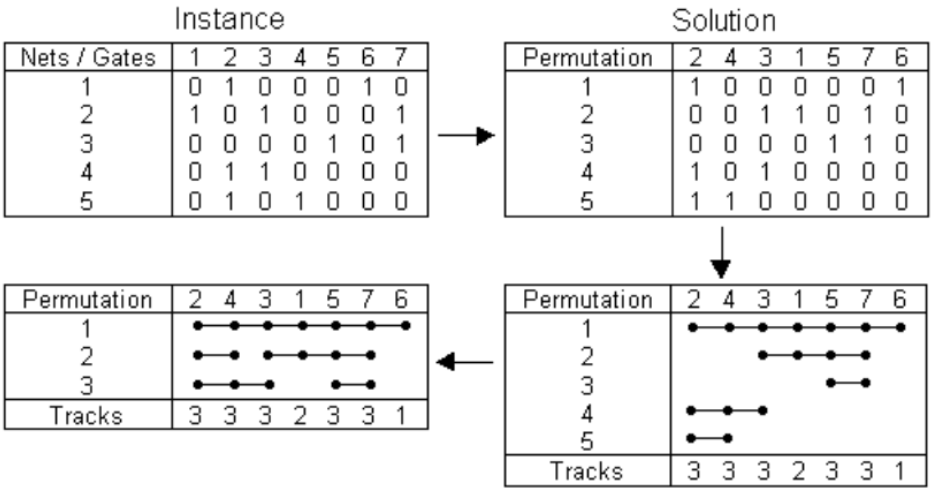
Fonte: WONG; LEONG; LIU (1988)

entre a porta i e a rede j.

O padrão VLSI pede que todos os transístores em uma mesma rede estejam interconectados e caso estas conexões se sobreponham, mais de uma rede física (trilha) precisa ser implementada. Tendo o número de superposições, sabe-se quantas trilhas serão necessárias para a construção do circuito.

O problema é, então, encontrar a ordem em que as portas devem ser dispostas (qual permutação) de forma a minimizar a quantidade necessária de trilhas. A figura 5 mostra um exemplo. Começando no canto superior esquerdo e seguindo em sentido horário temos a instância original, uma possível solução (permutação das portas), o circuito correspondente à permutação com as trilhas sobrepostas e, por último, o circuito real compactado.

Figura 5 – Instância exemplo do GMLP.



Fonte: MOSCATO; MENDES; LINHARES (2004)

Pode-se ver, então, que o GMLP é similar ao MOSP. No primeiro, quer-se minimizar a sobreposição de trilhas em um circuito integrado. No segundo, o objetivo é minimizar o número de pilhas abertas ao mesmo tempo (sobreposição das pilhas). Esta não é apenas uma similaridade, entretanto. Os dois problemas são equivalentes, como foi demonstrado em LINHARES; YANASSE (2002). Além disso, foi também demonstrado que ambos são NP-árduos (*NP-Hard*).

Por serem NP-árduos, acredita-se que não existe método resolutivo em tempo polinomial, a não ser que $P = NP$. Sendo assim, métodos aproximados são necessários, tópico que será abordado a seguir.

2.2 Computação Evolutiva

Computação evolutiva é a área da ciência que estuda a otimização e aprendizado através da evolução. A evolução das espécies, segundo a Teoria de Darwin, pode ser interpretada como um processo de aprendizado e otimização dos seres através do princípio de "sobrevivência do mais apto". Algoritmos Evolutivos (AEs) são os algoritmos contidos dentro da área de computação evolutiva. Como YU; GEN (2010) estabelece, eles têm três propriedades principais:

- São baseados em população – um grupo de soluções chamado de *população* é mantido;
- A aptidão dos indivíduos orienta a evolução – cada solução da população é chamado de *indivíduo*. Ele contém informação codificada que é avaliada pela função objetivo. (medição da aptidão);
- São dirigidos pela variação – os indivíduos sofrem diversas modificações no seu código, o que equivale a transitar pelo espaço de busca.

Vários algoritmos evolutivos foram propostos nas últimas décadas. Dentre eles, um dos mais comuns e que será usado como base para este trabalho, está o Algoritmo Genético (AG).

2.2.1 Algoritmo Genético

Este algoritmo foi inicialmente proposto em HOLLAND (1975) mas, em geral, sua implementação é diferente do algoritmo canônico lá descrito. O que se costuma fazer é aplicar uma suíte de operadores genéticos para criar um algoritmo personalizado (EIBEN; SMITH, 2003). Uma versão muito simples e similar à canônica será usada como exemplo.

Inicialmente, precisa-se definir uma codificação para os indivíduos. Uma forma muito comum é utilizar uma *string* binária. Cada indivíduo representa uma solução possível

do problema e cada solução é representada por esta *string*. Costuma-se utilizar termos da biologia como nomenclatura (daí o nome Algoritmo *Genético*). Estas *strings* ou vetores são chamadas de *cromossomos* ou *material genético* e cada *bit* é um *gene*.

O material genético do indivíduo também pode ser chamado de *genótipo*. O que o genótipo significa depende do problema. Por exemplo, se o objetivo é encontrar o máximo de uma função matemática de uma variável, pode-se arbitrar que o vetor representará um número de ponto flutuante que é a entrada da função. Usando este mesmo exemplo, o resultado da função é chamado de *aptidão*. A aptidão mede a qualidade do indivíduo (*fenótipo*).

Para a primeira geração, é necessária a criação de uma população. Esta população inicial, em geral, é criada aleatoriamente. Em seguida, precisa-se adicionar variabilidade aos indivíduos para que mais partes do espaço de soluções seja explorado. Para isso, utiliza-se os operadores genéticos: *seleção*, *crossover* e *mutação*.

A seleção de indivíduos é uma simulação da *seleção natural* de Darwin. Na seleção natural, indivíduos mais aptos (melhor adaptados ao ambiente) têm maior chance de se reproduzir e, consequentemente, passar seus genes para gerações posteriores. No algotitmo genético simples essa seleção é artificial. Um mecanismo de seleção utiliza o valor da função objetivo como uma medida da aptidão: um indivíduo tem maior chance de ser selecionado para reproduzir quanto maior for sua aptidão (*fitness*).

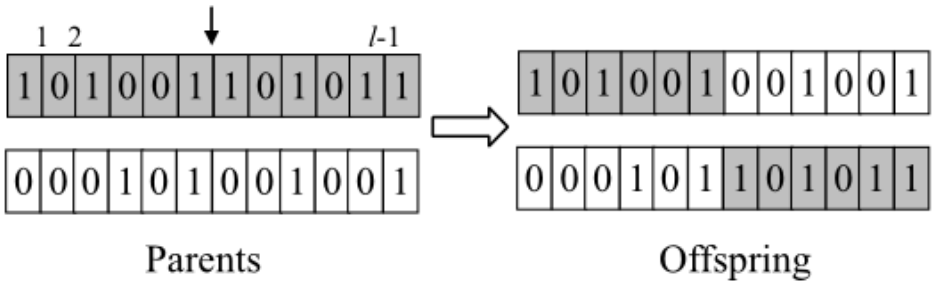
Utilizando essa metodologia, dois indivíduos são selecionados para se reproduzirem. A reprodução é simulada através de um *cruzamento* ou *crossover*. Um dos mecanismos da biologia para aumentar a variabilidade genética é o *crossover* ou *crossing-over*. Durante a formação de células reprodutivas (gametas), acontece a *meiose*. Neste processo, dois cromossomos trocam uma porção de seu material genético.

Analogamente à biologia, dois indivíduos que foram selecionados da população têm parte da sua *string* trocada, como pode ser visto na Figura 6. Os filhos são, então, introduzidos na população da geração seguinte. Este processo de reprodução é repetido até que a nova população esteja completa.

A criação de novas populações a partir de reproduções é repetida a cada geração até que a população venha a convergir para um ótimo local ou até que outro critério de parada seja atingido. Exemplos de critério de parada incluem número máximo de gerações, número máximo de chamadas à função objetivo, tamanho mínimo da população (em casos em que o tamanho da população é dinâmico), entre outros.

A reprodução, por si só, garante boa exploração do espaço de busca, entretanto não é suficiente. Costuma-se adicionar um ruído no processo evolutivo a fim de criar novas características que podem não ter sido originalmente capturadas pela população inicial. Para isso, utiliza-se o operador *mutação*. Assim como na biologia, durante a vida

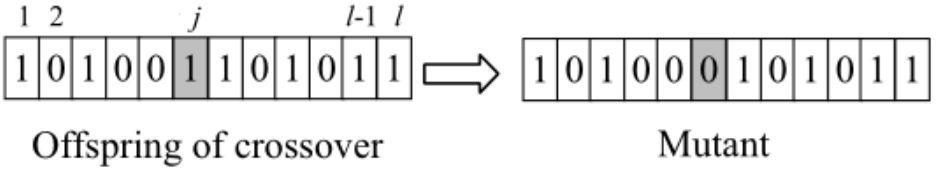
Figura 6 – Exemplo de *crossover*.



Fonte: YU; GEN (2010)

dos indivíduos, parte do material genético sofre mutações que aumentam a variabilidade genética. Neste AG simples, a mutação é feita através da mudança de 1 ou mais *bits* da string, como na Figura 7.

Figura 7 – Exemplo de mutação.



Fonte: YU; GEN (2010)

Sendo assim, um pseudocódigo que exemplifica uma implementação bem simples do algoritmo genético pode ser vista no Algoritmo 1, utilizando MAX_GEN como número máximo de gerações e 100 indivíduos por população.

No caso dos problemas estudados neste trabalho, entretanto, os indivíduos não são representados por vetores binários. Cada indivíduo é, na verdade, uma permutação possível dos padrões. Um exemplo pode ser visto na Figura 8. Acima, a ordem inicial dos padrões e abaixo, uma possível permutação do mesmo grupo de nove padrões.

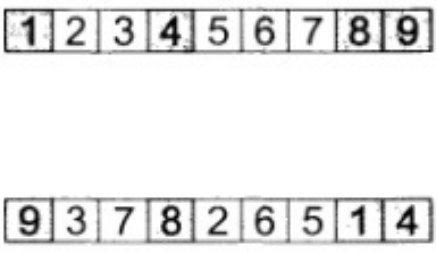
2.3 Treinamento Populacional em Heurísticas

Para muitos problemas computacionais combinatórios, encontrar sempre a solução ótima através de métodos exatos é impossível. Isto porque o tempo de execução necessário seria muito grande. Nesses casos, problemas NP-árduos, por exemplo, é comum utilizar métodos *heurísticos*. Estes métodos são capazes de gerar soluções de qualidade aceitável em tempo razoável mas não dão nenhuma garantia a respeito do grau de ótimo (*optimality*) da solução. Além disso, métodos heurísticos dependem do problema em questão (YU; GEN, 2010).

Algorithm 1 Algoritmo Genético Simples

```
1:  procedure AlgoritmoGeneticoSimples
2:      populacao ← gera_populacao()                    ▷ População inicial
3:      geracao ← 0
4:      melhor ← seleciona_melhor(populacao)            ▷ Melhor indivíduo
5:
6:      while geracao ≤ MAX_GEN do
7:          nova_populacao ← {}                         ▷ Conjunto vazio
8:          tamanho ← 0                                 ▷ Tamanho da população
9:
10:         while tamanho ≤ 100 do
11:             pai, mae ← seleciona_pais()
12:             filho ← cruzamento(pai, mae)
13:             filho ← mutacao(filho)
14:
15:             populacao[tamanho] ← filho              ▷ Inserido na população
16:             tamanho ← tamanho + 1
17:
18:             melhor_atual ← seleciona_melhor(nova_populacao)
19:             if melhor_atual.fitness ≤ melhor.fitness then
20:                 melhor ← melhor_atual               ▷ Atualiza o melhor indivíduo
21:
22:         geracao ← geracao + 1
23:     return melhor
```

Figura 8 – Exemplo de indivíduo na forma de permutação.



Fonte: EIBEN; SMITH (2003)

Alguns algoritmos, entretanto, garantem certa generalidade. Isto é, são aplicáveis a grande parte dos problemas de otimização. Este tipo de algoritmo é chamado de *metaheurística*. Dentro desta categoria, encontram-se os algoritmos evolutivos, a busca tabu (GLOVER, 1996) e o recozimento simulado (*simulated annealing*) (KIRKPATRICK C. D. GELATT, 1983). O algoritmo genético abordado na seção anterior, portanto, é uma metaheurística.

Existe uma competição entre dois aspectos no desenvolvimento de uma metaheurís-

tica: *intensificação* e *diversificação*. O balanço entre essas duas características é conhecido, no inglês, como *exploration-exploitation tradeoff*. Diversificação é necessária para garantir que toda parte do espaço de busca é inspecionado a fim de se ter uma boa estimativa do ótimo global. Intensificação é importante pois o refinamento de uma solução geralmente produz uma solução melhor (TALBI, 2002).

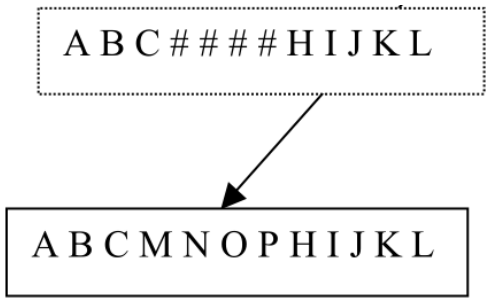
Metaheurísticas baseadas em população, como o AG, possuem bom potencial de diversificação, mas fraca intensificação. O contrário é verdade para metaheurísticas de busca local. Pode-se dizer, então, que as duas classes de algoritmos se complementam e, para se aproveitar de ambas qualidades, foram desenvolvidas as chamadas *metaheurísticas híbridas*. Enquanto o algoritmo populacional otimiza a busca globalmente, a busca local intensifica e refina as soluções encontradas. Outros tipos de hibridismo são estudados e classificados em TALBI (2002).

Um exemplo de metaheurística híbrida é o *Treinamento Populacional em Heurísticas* (TPH), introduzido em OLIVEIRA (2004). Nesta metodologia, a heurística de intensificação é utilizada para encontrar regiões promissoras do espaço de busca. O TPH será descrito em maior detalhe a seguir.

2.3.1 Regiões promissoras

A abordagem do TPH é inspirada no conceito de *regiões promissoras*. Este conceito, por sua vez, é definido a partir da definição de *esquemas* (GOLDBERG, 1989). Esquemas são soluções incompletas do problema estudado e representam hiperplanos do espaço de busca. Um exemplo de esquema para um problema com espaço de busca codificado como permutação de letras do alfabeto pode ser observado na Figura 9. A falta de informação é representada por um #, valor que pode ser substituído por letras ainda não utilizadas, neste caso.

Figura 9 – Exemplo de um esquema e uma possível estrutura instanciada.



Fonte: OLIVEIRA (2004)

Cada esquema pode instanciar um conjunto (ou uma família) de soluções completas, chamadas *estruturas*. Uma estrutura pode ser interpretada com um ponto no espaço hiperdimensional de busca, por isso diz-se que um esquema representa um hiperplano

deste espaço. Estes hiperplanos são subconjuntos do universo de busca ou, como sugerido anteriormente, *regiões* deste universo.

A decisão por utilizar métodos aproximados, como a computação evolutiva, leva em consideração a dificuldade de explorar todas as soluções de um espaço de busca grande de um problema NP-árduo. Uma forma de reduzir o número de soluções avaliadas seria, então, reconhecer quais regiões do espaço devem ser melhor exploradas sem avaliar todos os indivíduos contidos nela. Regiões bem avaliadas são chamadas de regiões promissoras e o objetivo do TPH é detectá-las.

No trabalho de OLIVEIRA (2004), duas abordagens são exploradas. Uma delas é dita *construtiva* e a outra *não-construtiva*. Ambas têm como base o algoritmo originalmente proposto em LORENA; FURTADO (2001) chamado *AGC* (Algoritmo Genético Construtivo).

O termo construtivo se refere à construção de soluções completas (estruturas) a partir de soluções incompletas (esquemas). A ideia é que, ao longo das gerações, blocos construtivos promissores, isto é, bem avaliados, tenham mais chances de se reproduzirem e de dirigir a evolução para uma região com maiores chances de conter o ótimo global. A incorporação de heurísticas de treinamento dão origem ao chamado AGC^H, que não será abordado neste trabalho.

A abordagem não-construtiva é chamada de *ATP*, Algoritmo de Treinamento Populacional, e será a base para os testes efetuados. Diferente do AGC^H, o ATP não é inicializado com uma população que contém esquemas. A população inicial é totalmente composta de estruturas, já que nem sempre a utilização de esquemas é vantajosa, especialmente quando o espaço de genótipos é idêntico ao de fenótipos, como é o caso do MOSP e GMLP.

2.3.2 Treinamento

Treinamento, em geral, refere-se a um processo em que um modelo se adapta a um determinado ambiente inicialmente desconhecido (OLIVEIRA, 2004). Pode-se dizer, então, que no processo evolutivo de um Algoritmo Genético, os indivíduos estão sendo treinados pela função objetivo, já que ela define seus graus de aptidão.

A ideia do TPH é adicionar uma segunda ligação entre os indivíduos e o problema em questão através de conhecimentos adicionais (heurísticas) e uma segunda função de avaliação. Tendo um indivíduo como ponto de partida, uma heurística é capaz de determinar uma *vizinhança* (região) e o papel dessa heurística é avaliar se o indivíduo em questão é o melhor adaptado a ela. Esta segunda avaliação aumenta a velocidade de convergência da população.

Cada indivíduo da vizinhança tem seu próprio valor de função objetivo f e sua

avaliação heurística g. Se, para um determinado indivíduo, $f = g$, dizemos que ele está *bem adaptado* à heurística empregada e é o melhor dentro da vizinhança por ela determinada. É interessante, entretanto, medir a distância entre a solução original e as soluções encontradas através da heurística. Um modo simples de calcular essa distância é utilizar a diferença $|f - g|$.

Suponhamos que H seja a heurística empregada em cima de um espaço de busca $S = \{s_k\}$. H determina sobre s_k, então, uma vizinhança $\varphi^H(s_k) = \{s_k, s_{v1}, s_{v2}, ..., s_{vl}\}$. O valor $g(s_k)$ é o resultado da busca heurística nesta vizinhança, ou seja, o *melhor vizinho* de s_k. Se o problema for de minimização e utilizarmos a própria função objetivo f para medir a qualidade dos vizinhos, podemos dizer que:

$$g(s_k) = f(s_V), s_V \in \varphi^H, f(s_V) \leq f(s_k) \tag{2.3}$$

E a distância mencionada anteriormente, pode ser definida como:

$$\wp(s_k, s_V) = |f(s_k) - g(s_k)| \tag{2.4}$$

2.3.3 Algoritmo de Treinamento Populacional

Por não trabalhar explicitamente com esquemas, pode parecer que o ATP não tem relação com detecção de regiões promissoras. Entretanto, pode-se dizer que este enfoque não-construtivo avalia sim blocos genéticos comuns à vizinhança, já que indivíduos vizinhos têm genes similares ou proximidade genotípica (OLIVEIRA, 2004).

Os indivíduos avaliados pelo ATP são ordenados através de um *ranking* δ que influencia suas probabilidades de reprodução. Este *ranking* leva em consideração a adaptação do indivíduo em relação à heurística empregada e também a sua avaliação da função objetivo.

Seja um indivíduo s_k de um problema de minimização, seu *ranking* é determinado por:

$$\delta(s_k) = d \cdot [G_{max} - f(s_k)] - |f(s_k) - g(s_k)| \tag{2.5}$$

O termo $|f(s_k) - g(s_k)|$, definido na Equação 2.4 mede a adaptação do indivíduo à heurística H empregada e o termo $d \cdot [G_{max} - f(s_k)]$ mede a avaliação da função objetivo. Pode-se ver, então, que a avaliação do indivíduo é penalizada proporcionalmente à distância $\wp(s_k, s_V)$ encontrada na vizinhança $\varphi^H(s_k)$. Os indivíduos melhor avaliados são, portanto, os que têm baixo valor de função objetivo (minimização) e que estejam bem adaptados à H.

O valor G_{max} é um limiar superior encontrado durante a criação da população inicial: o maior valor de função objetivo é atribuído a ele e qualquer indivíduo em gerações subsequentes com $f \geq G_{max}$ é rejeitado. A constante d é arbitrada experimentalmente e serve para balancear os dois componentes da Equação 2.5. Geralmente atribui-se a d um valor próximo a $1/G_{max}$.

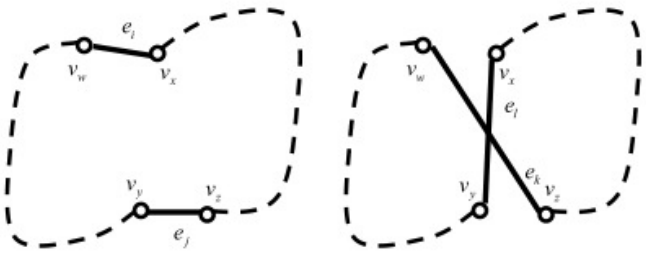
Agora que o funcionamento geral do ATP está descrito, é necessário especificar como as vizinhanças φ^H são encontradas. Mais especificamente, quais heurísticas H serão empregadas. Neste trabalho, serão utilizadas as mesmas duas heurísticas base que foram utilizadas em OLIVEIRA (2004). São elas 2-Opt (CROES, 1958) e Faggioli-Bentivoglio (FAGGIOLI; BENTIVOGLIO, 1998), que serão explicadas a seguir.

2.3.4 A heurística 2-Opt

A heurística 2-Opt inicialmente proposta por CROES (1958) é um método de busca local e foi desenvolvida com o Problema do Caixeiro Viajante (PCV) em mente. Neste problema, o caminho de menor custo quer ser encontrado dentro de um grafo. O custo entre dois vértices é representado pelo peso da aresta entre eles e geralmente é representado como uma distância entre cidades.

A partir de uma sequência inicial, chama-se de *movimento 2-Opt* a troca de duas arestas que não compartilham um mesmo vértice. Um exemplo de movimento pode ser observado na Figura 10. Todas as trocas possíveis são realizadas para um indivíduo (sequência original de cidades, por exemplo) até que se encontre a melhor combinação de arestas possíveis, ou seja, o ótimo local desta vizinhança.

Figura 10 – Exemplo de movimento 2-Opt. Troca de arestas não consecutivas.



Fonte: YU; GEN (2010)

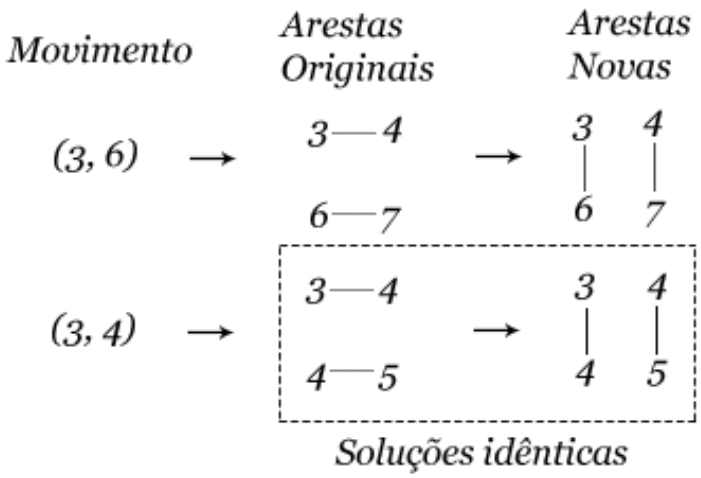
No caso do PCV, a diferença de custos entre a solução original e uma solução derivada é apenas a diferença entre a soma das arestas originais e a soma das arestas resultantes. Para problemas de sequenciamento de padrões, entretanto, se há uma mudança na ordem dos padrões, é necessário avaliar toda a nova sequência gerada para então saber se houve melhoria no custo.

Uma variação do 2-Opt, portanto, é utilizada em OLIVEIRA (2004) e também neste trabalho. Apenas parte de todas as trocas de aresta possíveis são avaliadas. Mais especificamente, l trocas são feitas, ou seja, a heurística encontra apenas uma estimativa do melhor indivíduo na vizinhança φ^{2-Opt}. O tamanho l da vizinhança é arbitrado experimentalmente e define o esforço computacional necessário para avaliar φ^{2-Opt} já que o número de avaliações necessárias é dado por $L_{2-Opt} = (l \cdot (l-1))/2$.

Supondo um indivíduo com 10 padrões e $l = 4$, 6 permutações serão avaliadas. Um número p é escolhido ao acaso, 2, por exemplo, e a seguinte vizinhança é gerada: $\varphi^{2-Opt} = \{(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)\}$. Nesta notação, um par (p,q) representa as arestas $[p, p+1]$ e $[q, q+1]$ que serão trocadas.

Usando esta abordagem, arestas consecutivas podem ser selecionadas para troca. A Figura 11 mostra como isto é redundante já que soluções idênticas seriam geradas.

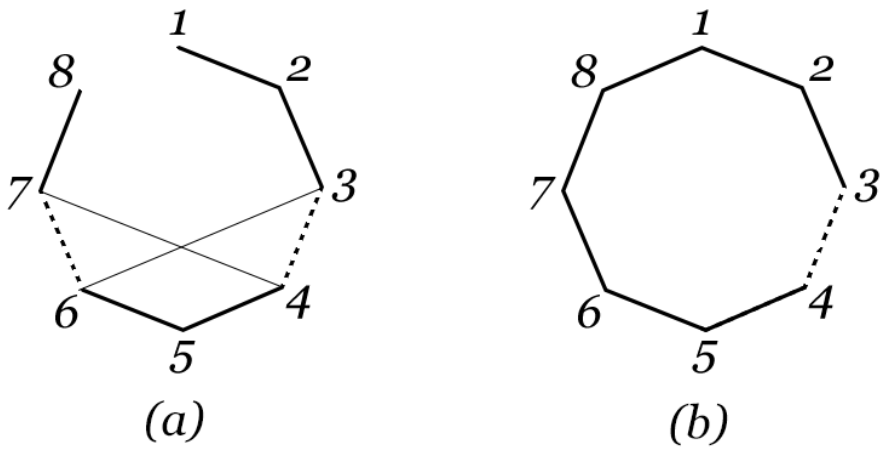
Figura 11 – Redundância na troca de arestas consecutivas



Para solucionar este problema, um *movimento 2-Opt alternativo* é utilizado. O movimento $(p, p+1)$, que tem arestas consecutivas, por exemplo, seria realizado removendo a aresta $[p, p+1]$ e adicionando a aresta $[n-1, 1]$, considerando que o indivíduo é uma sequência de n padrões. Além disso, o indivíduo agora irá iniciar a sequência a partir do vértice $p+1$.

A Figura 12 mostra graficamente um exemplo de troca de arestas não-consecutivas (a) utilizando o movimento comum e consecutivas (b), utilizando o movimento alternativo. A partir da sequência original 12345678, o movimento (3, 6) em (a) resulta na sequência 12365478. Já em (b), a aresta **8-1** é adicionada e a sequência 45678123 é formada.

Figura 12 – (a) Movimento 2-Opt comum (b) Movimento 2-Opt alternativo



2.3.5 A heurística de Faggioli-Bentivoglio

A segunda heurística de treinamento utilizada é baseada no trabalho de FAGGIOLI; BENTIVOGLIO (1998). Esta heurística é dita construtiva no sentido de que a vizinhança é construída a partir de regras de construção de soluções.

Seja uma solução inicial s_k de tamanho n, parte do começo dela é fixada. Essa parte fixada é chamada de π'. O tamanho a ser fixado é arbitrado a partir de um parâmetro l, a parte fixada deve ter $n - l$ padrões. Diz-se que a vizinhança φ^{fagg} é definida por todas as soluções que começam com π'.

A partir daí, os padrões restantes devem ser anexados ao final da solução sendo construída a fim de gerar uma solução completa que será avaliada pela função objetivo e dará o valor de $g(s_k)$. Essa construção é dirigida por uma hierarquia de 3 regras:

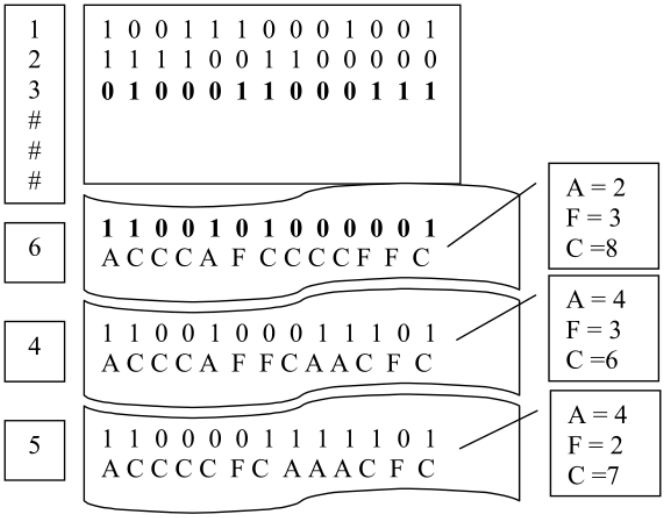
1. Dentre os padrões disponíveis (não contidos em π', é escolhido aquele que abrir o menor número de pilhas novas;
2. Dentre os selecionados no item anterior, é escolhido o que fechar o maior número de pilhas;
3. Caso ainda seja necessário desempate, é escolhido o padrão que mantiver o maior número de pilhas já abertas.

Há ainda um quarto item, caso ainda assim haja empate entre os padrões. Neste caso, um padrão é escolhido ao acaso entre os candidatos.

Um exemplo é mostrado na Figura 13. O padrão inicialmente contem a sequência **123456**. Para um valor de $l = 3$, temos que $\pi' = (1, 2, 3)$ e $\varpi = \{4, 5, 6\}$ é o conjunto de padrões restantes candidatos a serem anexados à quarta posição da solução que está sendo construída. O padrão 6 é o mais apto a ser anexado, já que tem o menor número de

novas pilhas que serão abertas ($A = 2$). Em seguida, como os padrões restantes 4 e 5 têm o mesmo valor de A, o número de pilhas que serão fechadas (F) serve como desempate, dando prioridade ao 4. A solução vizinha encontrada a partir dessa construção e que será usada para o cálculo de $g(s_k)$ é, então, **123645**.

Figura 13 – Exemplo da aplicação dos critérios da heurística Faggioli-Bentivoglio.



Fonte: OLIVEIRA (2004)

2.3.6 Outras considerações sobre a implementação do ATP

Por ter um tamanho de população dinâmico, o ATP utiliza um mecanismo de poda. Indivíduos com *ranking* $\delta \leq \alpha$ são eliminados. Este α é um limiar que é atualizado a cada geração a fim de se tornar mais restritivo ao longo do tempo. A Equação 2.6 mostra seu comportamento. Aqui, T é o número máximo de gerações e t é a geração atual. δ_P e δ_1 são, respectivamente, o melhor e o pior ranking da população atual. $|P|$ é o tamanho da população e k é um parâmetro de ajuste que serve para controlar o quão rápido o valor de α deve crescer.

$$\alpha = \alpha + k \cdot |P| \cdot \frac{\delta_P - \delta_1}{T - t} \tag{2.6}$$

Para realizar os cruzamentos, um dos pais é selecionado de um percentual elite da população enquanto o outro é selecionado da população total. A chance de um indivíduo ser selecionado é proporcional ao seu *ranking*. Antes de efetuar o cruzamento e avaliar a prole, uma mutação 2-Opt agressiva é realizada no indivíduo elite selecionado. Esta mutação funciona como o mecanismo de intensificação de busca dentro da região promissora (elite) detectada a cada geração.

Diferente da heurística de treinamento 2-Opt utilizada, em que apenas uma vizinhança de largura l é inspecionada, a mutação 2-Opt analisa sucessivas vizinhanças de largura l em forma de árvore: a melhor estrutura de cada nível é usada como raiz da próxima vizinhança a ser inspecionada. A mutação para quando a solução não puder mais ser melhorada ou se um limite de profundidade arbitrado for atingido.

3 Treinamento Populacional em Múltiplas Heurísticas

A metodologia utilizada em OLIVEIRA (2004) utiliza uma única heurística para treinar a população de um algoritmo genético. A ideia deste trabalho é ampliar o algoritmo de forma a permitir mais de uma heurística de treinamento. No caso, serão utilizadas as duas heurísticas (2-Opt e Faggioli-Bentivoglio) ao mesmo tempo mas de formas diferentes.

Três ideias de multiplicidade foram propostas: competição, cooperação e paralelismo com migração.

No treinamento *competitivo* (COMP), em uma mesma população existem indivíduos que são treinados por heurísticas diferentes na hora de sua criação. Isto é, quando um indivíduo é criado e vai ser avaliado pela heurística de treinamento, há 50% de chance de ele ser classificado como treinado por 2-Opt (OPT) e 50% de chance de ser treinado por Faggioli-Bentivoglio (FAG).

Não existe predeterminação a respeito de qual heurística será utilizada, por isso existe um pequeno desequilíbrio em relação ao número de indivíduos de cada heurística. Mas, estatisticamente, se houver n heurísticas de treinamento sendo utilizadas, cada indivíduo terá $1/n$ de chance de ser treinado por cada heurística.

Nesta heurística competitiva, o cálculo do *ranking* (δ) de cada indivíduo é feito utilizando a mesma Equação 2.5 que é utilizada para uma única heurística. Isto porque apesar de haver múltiplas heurísticas, apenas uma heurística é aplicada de cada vez.

O termo competitivo vem do fato de que as heurísticas de treinamento estão competindo em uma mesma população para dirigir a evolução para as suas regiões promissoras específicas. Cada heurística dirigirá seus indivíduos para uma região diferente do espaço de busca e consequentemente isso resultará em indivíduos com *rankings* diferentes que estão competindo para serem selecionados para o cruzamento e intensificação.

A abordagem *cooperativa* (COOP), por sua vez, tenta proporcionar uma ajuda mútua entre as heurísticas de treinamento. Cada indivíduo da população é avaliado pelas duas heurísticas e aquela que obtiver melhor avaliação, é selecionada. O cálculo do *delta*, entretanto, precisa levar em conta as duas avaliações. Para isso, a Equação 3.1 foi utilizada.

$$\delta = \alpha \cdot (G_{max} - f) - \beta \cdot ((f - g_{fagg}) + (f - g_{opt})) \tag{3.1}$$

Nesta equação, f é o valor da função objetivo e g_{fagg} e g_{opt} são os valores das heurísticas de treinamento FAG e OPT, respectivamente. Os parâmetros α e β são ajustados

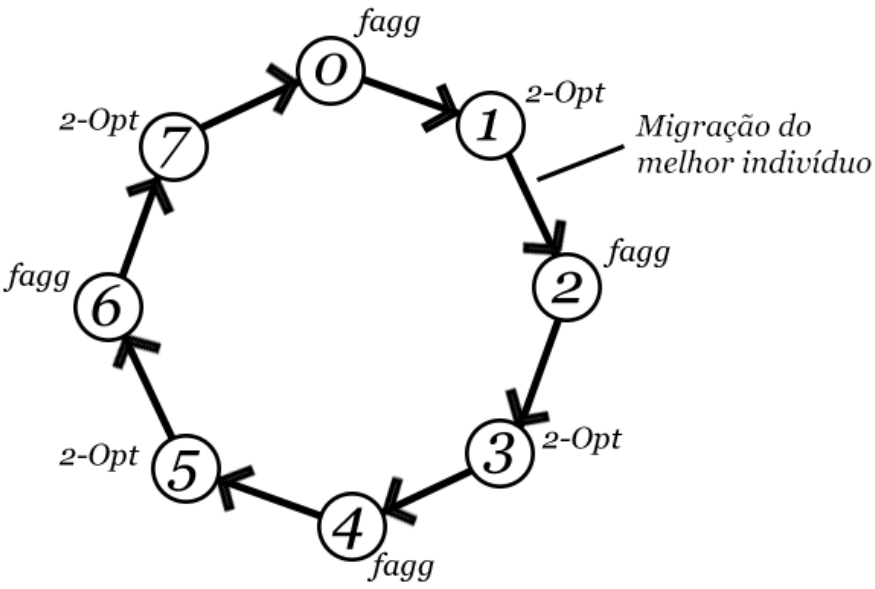
para balancear a importância da função objetivo e das heurísticas para o ranking. Os valores utilizados foram $\alpha = 0.6$ e $\beta = 0.4$.

Uma terceira abordagem de multiplicidade foi proposta: populações em paralelo. Nesta implementação, feita utilizando MPI, n processos são iniciados, cada um utilizando uma heurística de treinamento. No caso de existirem apenas duas heurísticas, se o processo tiver uma ID par, ele usará FAG, caso seja ímpar, OPT. Esta metodologia de distribuição é facilmente estentida para mais de duas heurísticas utilizando o operador matemático de resto.

Os processos guardam, separadamente, a melhor solução que foi encontrada até o momento. Ao final de toda geração, cada processo verifica se um indivíduo superior ao atual melhor que está guardado foi encontrado. Caso isto seja verdade para o processo p, uma mensagem é passada ao processo $(p + 1)\%n$, onde n é o total de processos e % é o operador de resto. Nesta mensagem está contida a permutação (material genético) daquele indivíduo que foi encontrado.

Ao receber a mensagem, o processo $p+1$ avalia o indivíduo segundo a *sua* heurística e o adiciona na população. A passagem de mensagens funciona como um mecanismo de migração entre as populações deste modelo circular (Figura 14) e garante que sempre que um indivíduo migrar, ele será avaliado e classificado (*ranking*) por uma heurística diferente da da sua população original.

Figura 14 – Migração de indivíduos promissores entre as populações



4 Testes computacionais e resultados

Quatro instâncias de cada problema (MOSP e GMLP) foram selecionada para realização dos testes computacionais. Cada instância foi submetida à cinco variações do Algoritmo de Treinamento Populacional: ATP com Faggioli-Bentivoglio (FAG), ATP com 2-Opt (OPT), ATP com FAG e OPT competitivo (COMP), ATP com FAG e OPT cooperativo (COOP) e, por último, ATP com populações com FAG e OPT em paralelo. Neste último caso, testes foram feitos com 2, 4 e 8 processos (populações). No caso da metodologia em paralelo, os nomes utilizados nas tabelas e gráficos são PARA2, PARA4 e PARA8.

Todas as instâncias escolhidas para o MOSP contêm 40 padrões, mas têm quantidades variadas de tipos de peças (20, 30, 40, 50). São elas: p4020n9, p4030n9, p4040n9, p4050n9. As instâncias o GMLP (W1, W2, W3, W4) são relativamente mais trabalhosas computacionalmente do que as MOSP, especialmente a instância a instância W4, que é a maior instância encontrada na literatura, com 141 portas e 202 redes. Os ótimos conhecidos para a instâncias GMLP são, respectivamente, 4, 14, 18 e 27 (OLIVEIRA, 2004). Não foi encontrado de forma consistente, entretanto, os valores ótimos para as instâncias do MOSP na literatura. Sendo assim, os valores 13, 14, 16 e 15 foram utilizados para p4020n9, p4030n9, p4040n9 e p4050n9, respectivamente.

Cada metodologia foi executada 30 vezes, para que uma média dos resultados pudesse ser encontrada. Todos os testes foram executados em um processador AMD FX-8120 com 8 núcleos e 8GB de memória RAM. Os seguintes parâmetros foram utilizados para estas execuções:

- Máximo de gerações: 100;
- População inicial: 50;
- k_α (Constante de ajuste da Equação 2.6): 0.02;
- População Mínima: 10;
- Tamanho da vizinhança: 20;
- Número máximo de vizinhanças: 20;
- Desvio (Constante da Equação 2.5): 0.005;

Na Tabela 1 podemos ver a média do resultado encontrado para as 30 execuções de cada instância, bem como um intervalo de confiança dado pelo valor do desvio padrão.

A ideia é comparar as duas colunas da esquerda (OPT e FAG) com as duas da direita (COMP e COOP).

Tabela 1 – Resultados dos testes das 4 heurísticas de treinamento para um único processo

	Instância	OPT	FAG	COMP	COOP
MOSP	p4020n9	13 ± 0	13 ± 0	13 ± 0	13 ± 0
	p4030n9	14.17 ± 0.38	14.27 ± 0.45	14.27 ± 0.45	14.2 ± 0.41
	p4040n9	16.2 ± 0.41	16.07 ± 0.25	16.07 ± 0.25	16.27 ± 0.45
	p4050n9	15.03 ± 0.18	15 ± 0	15 ± 0	15.07 ± 0.25
GMLP	W1	4.1 ± 0.31	4.07 ± 0.25	4.03 ± 0.18	4.07 ± 0.25
	W2	14.2 ± 0.48	14.27 ± 0.64	14.2 ± 0.41	14.27 ± 0.64
	W3	18.73 ± 0.74	19.07 ± 1.31	18.93 ± 0.83	18.83 ± 0.75
	W4	29.8 ± 1.37	29.83 ± 1.58	30.07 ± 1.76	30.07 ± 1.36

A Tabela 2 mostra o comparativo de resultados para as mesmas instâncias utilizando a heurística de treinamento em paralelo, com 2, 4 e 8 processos (PARA2, PARA4 E PARA8). Pode-se ver, por exemplo, que, em comparação com as heurísticas não-paralelas, esta atingiu o ótimo com maior consistência (Instâncias W1 e W2). E para as instâncias maiores, tanto a média quando a variância atingiram resultados menores e, portanto, melhores. A instância W4, utilizando 8 processos, por exemplo, atingiu um resultado médio cerca de 7% melhor do que seus competidores não-paralelos.

Tabela 2 – Resultados dos testes da heurística em paralelo para 2, 4, e 8 processos (populações)

		Número de processos (populações)		
	Instância	2	4	8
MOSP	p4020n9	13 ± 0	13 ± 0	13 ± 0
	p4030n9	14.1 ± 0.31	14.07 ± 0.25	14 ± 0
	p4040n9	16.03 ± 0.18	16 ± 0	16 ± 0
	p4050n9	15 ± 0	15 ± 0	15 ± 0
GMLP	W1	4 ± 0	4 ± 0	4 ± 0
	W2	14 ± 0	14 ± 0	14 ± 0
	W3	18.57 ± 0.77	18.1 ± 0.31	18.03 ± 0.18
	W4	29.67 ± 1.63	28.53 ± 0.94	28.03 ± 0.72

Para corroborar a observação de que as metodologias paralelas tiveram melhor resultado que as não-paralelas, uma análise de variância (Teste ANOVA) foi feita para as instâncias do GMLP. A Figura 15 mostra graficamente o comparativo das médias encontradas por cada uma das 7 metodologias em W4 e a Tabela 3 mostra o valor de $F(6, 203)$ e o valor de p (p-value) para as 4 instâncias.

Pela Tabela 3, apenas para a instância W1 não houve diferença estatística significativa entre as médias ($p > 0.05$). Para comparar as metodologias duas a duas, então, foi feito um Teste de Tukey da diferença honestamente significativa (*Tukey's Honestly Significant Difference*). Os valores p ajustados podem ser vistos na Tabela 4.

Figura 15 – Boxplot comparativo das 7 metodologias

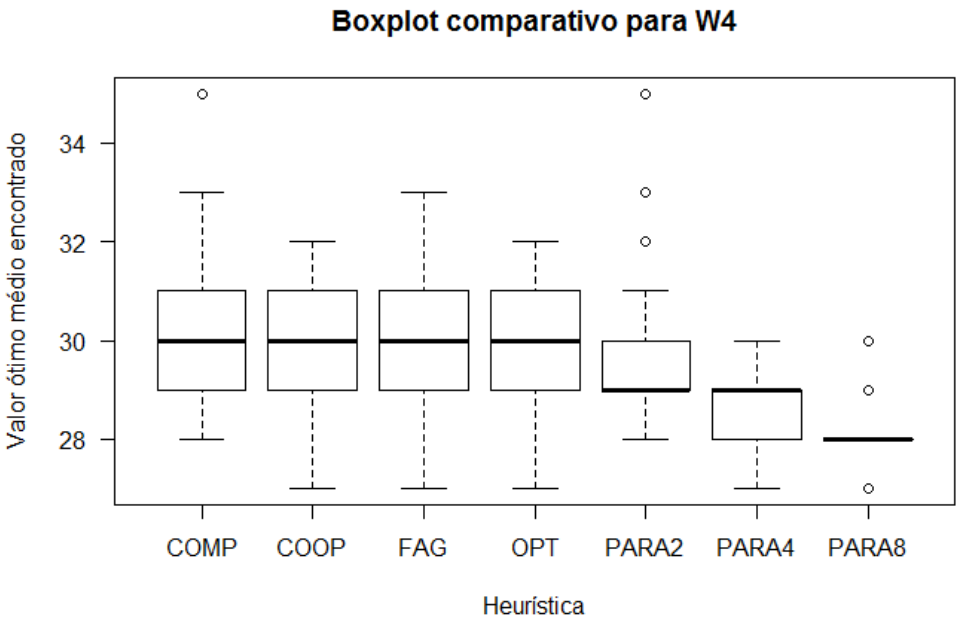


Tabela 3 – Comparativo do teste ANOVA para as 4 instâncias do GMLP

Instância	$F(6, 203)$	p-value
W1	1.35	0.237
W2	2.81	0.012
W3	8.04	8.49e-8
W4	10.97	1.42e-10

Apesar de o teste ANOVA indicar que $p = 0.012$ para instância W2, nenhum dos valores de p ajustados foram inferiores a 0.17. Ou seja, não houve diferença estatística significativa entre as médias para esta instância. Para as instâncias W3 e W4, entretanto, houve diferença ($p < 0.05$) ao comparar PARA4 e PARA8 com todas as metodologias não-paralelas (FAG, OPT, COMP, COOP). PARA8 encontrou a melhor média absoluta dentre todas as metodologias (28.03)

As metodologias não-paralelas, quando comparadas entre si, também não exibiram diferença estatística significativa para suas médias encontradas.

Apesar de a diferença entre a média final das metodologias não-paralelas não ter sido significativa, houve um destaque na evolução do valor médio, como pode ser visto na Figura 16. Pode-se ver que COOP teve um comportamento ligeiramente mais estável que seus competidores e atingiu o plateau de convergência média cerca de 250 mil chamadas à função objetivo antes.

O gráfico que pode ser visto na Figura 17 mostra um trecho da evolução da melhor solução para 8 populações em paralelo agrupadas por heurística. As populações que

Tabela 4 – Valores de p ajustados para a metodologias duas a duas

	p-value ajustado			
	W1	W2	W3	W4
COOP-COMP	0.9937623	0.9961541	0.9988613	1.0000000
FAG-COMP	0.9937623	0.9961541	0.9943502	0.9947689
OPT-COMP	0.8260996	1.0000000	0.9544288	1.0000000
PARA2-COMP	0.9937623	0.5115450	0.5322078	0.9206912
PARA4-COMP	0.9937623	0.5115450	**0.0009517**	**0.0005187**
PARA8-COMP	0.9937623	0.5115450	**0.0002470**	**0.0000008**
FAG-COOP	1.0000000	1.0000000	0.9074531	0.9947689
OPT-COOP	0.9937623	0.9961541	0.9988613	1.0000000
PARA2-COOP	0.8260996	0.1738185	0.8381469	0.9206912
PARA4-COOP	0.8260996	0.1738185	**0.0060316**	**0.0005187**
PARA8-COOP	0.8260996	0.1738185	**0.0018050**	**0.0000008**
OPT-FAG	0.9937623	0.9961541	0.6435032	0.9947689
PARA2-FAG	0.8260996	0.1738185	0.1685107	0.9992044
PARA4-FAG	0.8260996	0.1738185	**0.0000588**	**0.0061275**
PARA8-FAG	0.8260996	0.1738185	**0.0000129**	**0.0000201**
PARA2-OPT	0.4000009	0.5115450	0.9815819	0.9206912
PARA4-OPT	0.4000009	0.5115450	**0.0301683**	**0.0005187**
PARA8-OPT	0.4000009	0.5115450	**0.0106059**	**0.0000008**
PARA4-PARA2	1.0000000	1.0000000	0.2377970	0.0279266
PARA8-PARA2	1.0000000	1.0000000	0.1152193	**0.0001612**
PARA8-PARA4	1.0000000	1.0000000	0.9998906	0.7998829

utilizaram OPT estão em vermelho, enquanto as FAG estão em preto. Pode-se inferir que o compartilhamento (migração) de indivíduos torna o processo evolutivo como um todo mais homogêneo. Ou seja, há realmente uma cooperação entre as populações que estão evoluindo.

A Tabela 5 mostra os tempos médios de execução em paralelo para as instâncias do GMLP, que são mais trabalhosas computacionalmente. Para fins de comparação, os tempos para a heurística OPT também foram marcados, como pode-se ver na Tabela 6. Apesar de usar apenas uma heurística, serviu para o cálculo do Speedup (Equação 4.1) e da Eficiência (Equação 4.2), já que não existe versão serial da heurística com múltiplas populações e migração de indivíduos.

$$S = \frac{T_{serial}}{T_{paralelo}} \tag{4.1}$$

$$E = \frac{S}{N_{processos}} \tag{4.2}$$

Como era de se esperar, para instâncias pequenas (W1 e W2), o Speedup não foi significativo. Para as instâncias maiores, especialmente a W3, o resultado foi superior a 2

Figura 16 – Comparativo da evolução do ótimo médio para a instância W4 (GMLP)

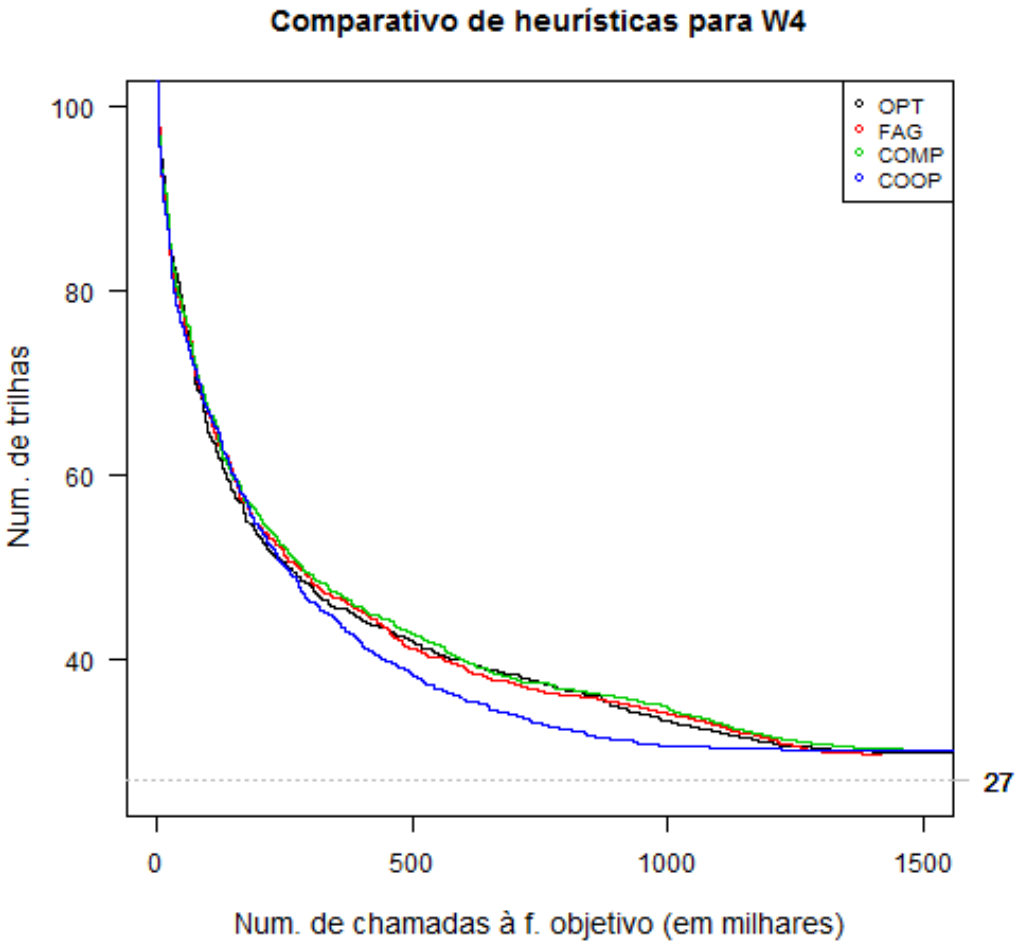


Tabela 5 – Tempo médio de execução em paralelo para instâncias GMLP

		Número de processos (populações)		
	Instância	2	4	8
GMLP	W1	$0.12 \pm 0.05s$	$0.12 \pm 0.04s$	$0.17 \pm 0.07s$
	W2	$0.81 \pm 0.31s$	$0.87 \pm 0.32s$	$1.08 \pm 0.41s$
	W3	$23.44 \pm 8.44s$	$19.8 \pm 8.42s$	$20.42 \pm 6.6s$
	W4	$409.02 \pm 29.86s$	$398.57 \pm 35.34s$	$451.59 \pm 24.8s$

nos três casos. Para a instância W4, o resultado foi um pouco menor, mas ainda assim superior a 1, embora tenha eficiência bem baixa para o caso de 8 processos (0.16).

É interessante ressaltar, entretanto, que estes valores são calculados apesar de o algoritmo não encontrar a solução ótima sempre. Ou seja, na maioria dos casos, a execução foi interrompida pela geração máxima ou pelo tamanho mínimo de população. Isto é diferente de avaliar o Speedup da paralelização de um algoritmo de ordenação, por exemplo, em que o resultado final é sempre o mesmo quando se parte do mesmo vetor inicial. Ou seja, não existe aleatoriedade. Além disso, os testes em paralelo foram

Figura 17 – Comparativo da evolução de 8 populações em paralelo para uma iteração da instância W4 (GMLP)

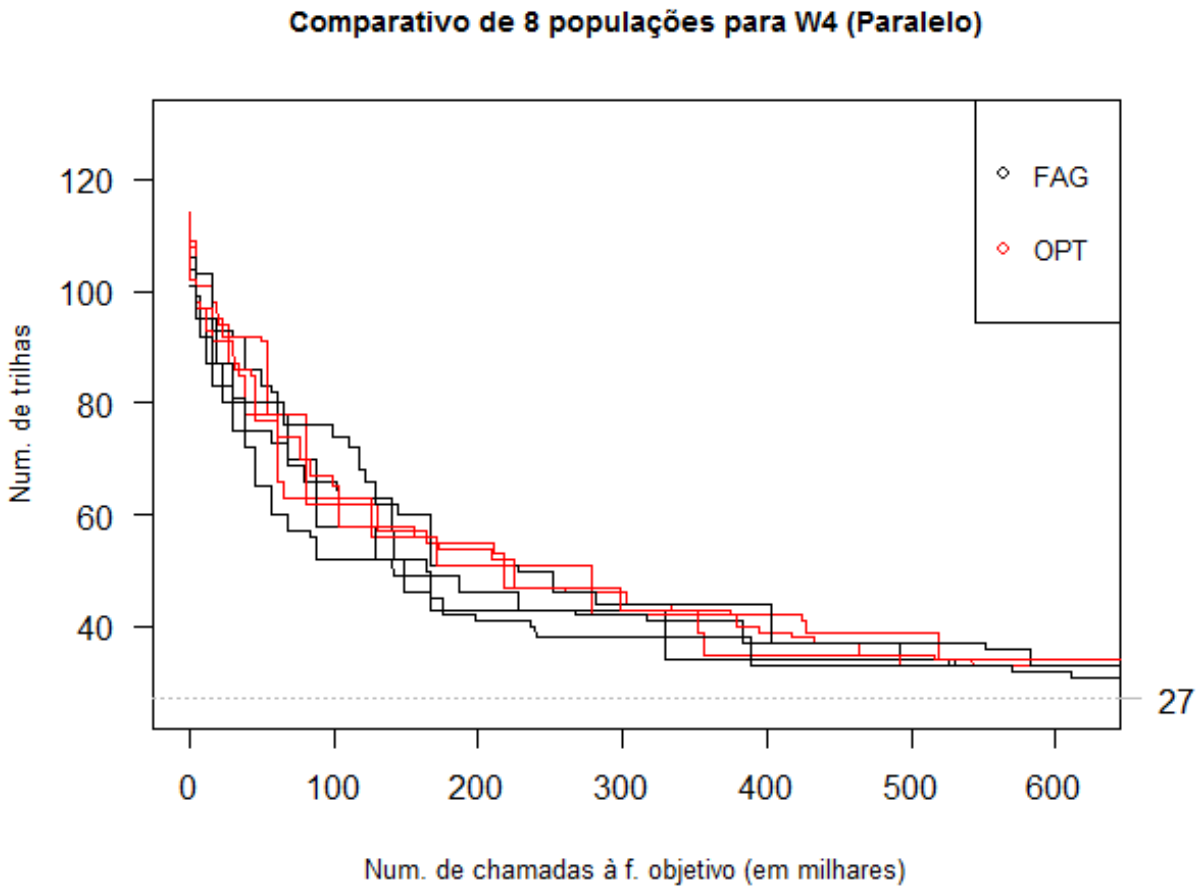


Tabela 6 – Tempo de execução das instâncias GMLP utilizando 2-Opt

	Instância	Tempo (OPT)
GMLP	W1	$0.11 \pm 0.02s$
	W2	$0.88 \pm 0.37s$
	W3	$57.49 \pm 37.64s$
	W4	$592.08 \pm 192.12s$

realizados em apenas uma máquina, então não há garantia de que todos os processos foram executados estritamente em paralelo a todo momento.

Tabela 7 – Relação dos valores de Speedup para as 4 instâncias do GMLP

		Número de processos (populações)		
	Instância	2	4	8
GMLP	W1	0.92	0.92	0.65
	W2	1.09	1.01	0.81
	W3	2.45	2.90	2.82
	W4	1.45	1.49	1.31

Tabela 8 – Relação dos valores de Eficiência para as 4 instâncias do GMLP

		Número de processos (populações)		
	Instância	2	4	8
GMLP	W1	0.46	0.23	0.12
	W2	0.55	0.25	0.10
	W3	1.23	0.73	0.35
	W4	0.75	0.37	0.16

5 Conclusão

O trabalho realizado serviu para explorar a utilização de múltiplas heurísticas de treinamento em conjunto para a detecção de regiões promissoras do espaço de busca. As heurísticas individuais utilizadas foram Fagiolli-Bentivoglio e 2-Opt. As heurísticas múltiplas propostas incluíram três variações: competitiva, cooperativa e em paralelo.

Oito instâncias de problemas de sequenciamento de padrões (MOSP e GMLP) foram utilizadas. Cada metodologia foi executada 30 vezes para cada instância a fim de encontrar um valor médio, bem como um tempo médio, no caso do GMLP, que são mais árduas.

As metodologias com heurísticas múltiplas de treinamento COOP e COMP não obtiveram resultados superiores à seus competidores OPT e FAG, que utilizam apenas uma heurística. Apesar disso, a heurística *cooperativa* COOP teve comportamento divergente das outras, mostrando-se mais estável na descida hiperbólica até o platô médio de convergência.

A heurística em paralelo, mais especificamente com 4 e 8 processos, se mostrou promissora tendo diferença estatística significativa ao ser comparada com as metodologias não-paralelas. Isto significa que ela encontrou o ótimo mais vezes e com maior consistência.

Algumas ideias de trabalhos futuros se apresentam aqui. Uma delas é expandir a aplicação deste tipo de algoritmo (ATP) para outros problemas como o Problema do Caixeiro viajante. Uma outra ideia, mais direta, consistiria em aumentar o número de heurísticas de treinamento que trabalharão em conjunto a fim de explorar regiões diferentes do espaço de busca. Ou ainda aplicar o treinamento populacional à problemas que já dispõem de várias heurísticas desenvolvidas.

Referências

CROES, G. A. A method for solving traveling-salesman problems. *Operations Research*, INFORMS, v. 6, n. 6, p. 791–812, 1958. ISSN 0030364X, 15265463. Disponível em: <http://www.jstor.org/stable/167074>. Citado 2 vezes nas páginas 12 e 26.

DANTZIG, G. B. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*. New York: Wiley, 1951. Citado na página 12.

EIBEN, A. E.; SMITH, J. E. *Genetic Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. 37–69 p. ISBN 978-3-662-05094-1. Disponível em: <http://dx.doi.org/10.1007/978-3-662-05094-1_3>. Citado 2 vezes nas páginas 19 e 22.

FAGGIOLI, E.; BENTIVOGLIO, C. A. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, v. 110, n. 3, p. 564 – 575, 1998. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221797002683>. Citado 3 vezes nas páginas 12, 26 e 28.

FINK, A.; VOß, S. Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, v. 26, n. 1, p. 17 – 34, 1999. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054898800014>. Citado na página 14.

GLOVER, F. Tabu search and adaptive memory programing – advances, applications and challenges. In: *Interfaces in Computer Science and Operations Research*. Berlin: Kluwer, 1996. p. 1–75. Citado na página 22.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675. Citado na página 23.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: U Michigan Press, 1975. Citado na página 19.

KIRKPATRICK C. D. GELATT, M. P. V. S. Optimization by simulated annealing. *Science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 00368075, 10959203. Disponível em: <http://www.jstor.org/stable/1690046>. Citado na página 22.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, vlsi design, and flexible machines. *Computers & Operations Research*, v. 29, n. 12, p. 1759 – 1772, 2002. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054801000545>. Citado 2 vezes nas páginas 15 e 19.

LOPEZ, A. D.; LAW, H. F. S. A dense gate matrix layout method for mos vlsi. *IEEE Transactions on Electron Devices*, v. 27, n. 8, p. 1671–1675, Aug 1980. ISSN 0018-9383. Citado na página 17.

LORENA, L. A. N.; FURTADO, J. a. C. Constructive genetic algorithm for clustering problems. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 9, n. 3, p. 309–327, set. 2001. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162/106365601750406019>. Citado na página 24.

MOORE, G. E. Cramming more components onto integrated circuits. *Electronics*, v. 38, n. 8, April 1965. Citado na página 16.

MOSCATO, P.; MENDES, A.; LINHARES, A. *VLSI design: gate matrix layout problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. 455–478 p. ISBN 978-3-540-39930-8. Disponível em: <http://dx.doi.org/10.1007/978-3-540-39930-8_18>. Citado 2 vezes nas páginas 17 e 18.

OLIVEIRA, A. C. M. de. *Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, 2004. Citado 10 vezes nas páginas 12, 16, 23, 24, 25, 26, 27, 29, 31 e 33.

TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, v. 8, n. 5, p. 541–564, 2002. ISSN 1572-9397. Disponível em: <http://dx.doi.org/10.1023/A:1016540724870>. Citado 2 vezes nas páginas 12 e 23.

WESTE, N.; HARRIS, D. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, 2011. ISBN 9780321547743. Disponível em: <https://books.google.com.br/books?id=sv8OQgAACAAJ>. Citado 2 vezes nas páginas 16 e 17.

WONG, D. F.; LEONG, H. W.; LIU, C. L. *Gate Matrix Layout*. Boston, MA: Springer US, 1988. 145–164 p. ISBN 978-1-4613-1677-0. Disponível em: <http://dx.doi.org/10.1007/978-1-4613-1677-0_7>. Citado 2 vezes nas páginas 17 e 18.

YANASSE, H. H. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, v. 100, n. 3, p. 454 – 463, 1997. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221797841070>. Citado 2 vezes nas páginas 14 e 15.

YU, X.; GEN, M. *Introduction*. London: Springer London, 2010. 3–10 p. ISBN 978-1-84996-129-5. Disponível em: <http://dx.doi.org/10.1007/978-1-84996-129-5_1>. Citado 3 vezes nas páginas 19, 21 e 26.

YUEN, B. J. Heuristics for sequencing cutting patterns. *European Journal of Operational Research*, v. 55, n. 2, p. 183 – 190, 1991. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179190222H>. Citado na página 14.