UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FREDERICO GOMES PIRES AZZOLINI

OTIMIZAÇÃO DO LAYOUT DE ESTALEIROS COM ALGORITMOS GENÉTICOS: UM ESTUDO COMPARATIVO

BAURU
Novembro/2019

FREDERICO GOMES PIRES AZZOLINI

OTIMIZAÇÃO DO LAYOUT DE ESTALEIROS COM ALGORITMOS GENÉTICOS: UM ESTUDO COMPARATIVO

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Profa. Dra. Simone das Graças Domingues Prado
Coorientador: Prof. Dr. Walther Azzolini Júnior

BAURU
Novembro/2019

Frederico Gomes Pires Azzolini Otimização do Layout de Estaleiros com Algoritmos Genéticos: Um Estudo Comparativo/ Frederico Gomes Pires Azzolini. – Bauru, Novembro/2019- 45 p. : il. (algumas color.) ; 30 cm.
Orientador: Profa. Dra. Simone das Graças Domingues Prado
Coorientador: Prof. Dr. Walther Azzolini Júnior
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Bacharelado em Ciência da Computação, Novembro/2019.
1. AG 2. BRKGA 3. Algoritmo Genético 4. Otimização 5. Estaleiros 6. Navio

Frederico Gomes Pires Azzolini

Otimização do Layout de Estaleiros com Algoritmos Genéticos: Um Estudo Comparativo

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Profa. Dra. Simone das Graças Domingues Prado
Orientadora
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profa. Dra. Andrea Carla Gonçalves Vianna
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Aparecido Nilceu Marana
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, _____ de _____ de _____.

Dedico estas páginas aos meus pais, aos quais eu não agradeço tanto quanto deveria e sem os quais eu não seria metade do que me tornei. À minha família, os que estavam desde o início e os que passaram a estar nos últimos 2 anos, pelo apoio e pelos exemplos que me fizeram crescer e aprender tanto. À minha namorada, por fazer parte dos meus sonhos e me incentivar a ser melhor diariamente. Aos meus amigos, de infância e de faculdade, pessoas que me ensinaram sobre respeito, leadade e amor e quem pretendo levar para o resto da vida. Por fim, dedico também ao time Bauru Badgers e todos os atletas, por me ensinar a confiar em mim e nas pessoas ao meu redor e me mostrar o verdadeiro significado de ser um vencedor.

Agradecimentos

Agradeço à Prof. Simone pela orientação, paciência e todo o auxílio durante o desenvolvimento deste trabalho.

Agradeço ao meu pai, Prof. Walther, pela oportunidade de me envolver neste trabalho.

Agradeço à UNESP e aos professores do Departamento de Computação pelos últimos 5 anos da minha vida, por toda a dedicação, experiências e conhecimento que me foram passados.

Eu posso não ter ido para onde eu pretendia ir, mas eu acho que acabei terminando onde eu pretendia estar.

Douglas Adams

Resumo

A presença de métodos computacionais de otimização na indústria naval é surpreendentemente limitada. Poucas pesquisas são conduzidas na área, gerando dificuldades de aplicação de novas tecnologias. Consequentemente se torna muito difícil a redução de custos, visto que em um estaleiro o produto produzido possui grandes dimensões, dificultando e aumentando o custo de transporte. Nos poucos trabalhos encontrados é comum a utilização de Algoritmos Genéticos para otimização de layout. Neste trabalho foram desenvolvidos, testados e comparados dois algoritmos: um algoritmo genético tradicional e um algoritmo genético de chaves viciadas. Os resultados se mostraram satisfatórios e trazem novas possibilidades de pesquisa.
Palavras-chave: GA, BRKGA, Algoritmo Genético, Otimização, Estaleiros, Navio.

Abstract

The presence of computing optimization methods in naval industry is astonishingly scarce. Very few researches are being conducted in the area which creates an obstacle to the application of new technology. Consequently it becomes very hard to reduce costs, since the products made in shipyards are very big, complicating transportation and scaling costs. In some of the few researches on the theme the application of Genetic Algorithms in layout optimization. For this work two algorithms were developed, tested and compared: a traditional genetic algorithm and a biased random-key genetic algorithm. The results were very positive and allow new research possibilities.

Keywords: GA, BRKGA, Genetic Algorithm, Optimization, Shipyard, Ship.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

FLP	*Facility Layout Planning*
AG	Algoritmo Genético
PLI	Planejamento do *Layout* de Instalações
AE	Algoritmos Evolutivos
BRKGA	Biased Random-Key Genetic Algorithm

Lista de códigos

Sumário

1 Introdução

Produtividade é considerada a relação entre o que é produzido e os fatores necessários para essa produção. Esses fatores podem ser definidos como pessoas, máquinas, materiais e outros. Aumento de produtividade consiste em produzir mais consumindo menos, o que implica em diminuição de despesas e consequentemente aumento de lucro.

Todo processo industrial, portanto, tem como objetivo alcançar o estágio máximo de produtividade. *Facility Layout Planning* (FLP) ou Planejamento do *Layout* de Instalações (PLI) compõe os estudos e estratégias envolvendo modos de organizar otimamente os fatores da produção de modo a alcançar esse objetivo. O PLI abrange fatores como o custo de manejo de determinados materiais, a distância percorrida por produtos intermediários ou finais, e o tempo de produção de produtos. O custo de manejo de materiais, por exemplo, pode representar de 30% a 40% dos custos de produção industrial, podendo aumentar para 70% em alguns casos específicos (MATSON; MELLICHAMPH; SWAMINATHAN, 1992). O tipo de produto manufaturado influencia diretamente essa percentagem. Indústrias como a do setor naval, devido ao peso e volume elevado dos materiais utilizados, tem uma quantidade considerável de seu custo voltada apenas ao seu manejo o que tende a aumentar conforme novos e maiores modelos de embarcação são produzidos.

Portanto a otimização do *layout* de estaleiros é imperativo para que seja possível reduzir ao máximo o custo do manejo e consequentemente do custo total de produção. Entretanto, os estaleiros tendem a ser construídos sem nenhum processo sistemático, baseando-se somente no conhecimento e experiência prática de *experts* na área (YOUNG et al., 2008). Uma realidade incompatível com o constante crescimento da participação da computação na indústria.

Devido a essa deficiência surgiram trabalhos cujos objetivos seriam utilizar poder computacional para tomar essas decisões de modo mais rápido, fácil e correto, poupando tempo, mão de obra especializada e recursos. Algoritmos Genéticos (AG), por exemplo, vêm sendo frequentemente escolhidos para lidar com problemas de PLI. (AIELLO; ENEA; GALANTE, 2006) buscaram resolver um problema de multi-objetivo, no qual era necessário maximizar o alcance dos requerimentos de distância, adjacência e proporção de área entre os departamentos enquanto minimizaria os custos de manejo de material. Azadivar e Wang (2000) tinham como objetivo minimizar o tempo do ciclo de produção e utilizaram os cromossomos do AG para representar as diferentes organizações possíveis dos departamentos.

Nick (2008) desenvolveu um sistema de duas etapas para encontrar o *layout* ótimo de estaleiros, a primeira etapa sendo constituída de um AG para determinar a melhor posição topológica dos departamentos e a segunda sendo um algoritmo de crescimento estocástico que busca encontrar a melhor geometria para cada prédio. Seu sistema utiliza penalidades

presentes na função objetivo referentes às dimensões dos departamentos e às suas relações de aspecto (lado maior dividido pelo lado menor). Baseados neste trabalho Choi, Kim e Chung (2017) aplicaram o mesmo sistema de duas etapas, por diminuir com sucesso a complexidade do problema ao reduzi-lo em dois mais simples, e pela sua eficiência em entregar um *layout* realista (PARSONS et al., 2008). Entretanto, buscaram adaptá-lo, reduzindo as influências das dimensões dos departamentos e adicionando fatores como restrições de posição entre eles.

Este trabalho tem como objetivo replicar o resultado de um dos artigos mais recentes na área com um novo algoritmo desenvolvido e aplicado pelo autor, de modo a demonstrar como a utilização de diferentes técnicas geram resultados diversos e concluir qual das utilizadas representa a melhor alternativa para a resolução do problema. Para isso é comparada a primeira etapa do sistema utilizado por Choi, Kim e Chung (2017) com outro similar, utilizando um algoritmo genético de características diferentes.

No Capítulo 2 são explicadas alguns aspectos teóricos importantes para o entendimento do texto, no Capítulo 3 são apresentadas as ferramentas utilizadas no trabalho. O Capítulo 4 explicita detalhes sobre os algoritmos e os testes executados. Por fim os Capítulos 5 e 6 listam os resultados provenientes dos testes feitos e a conclusão do autor.

2 Fundamentação Teórica

Através dos avanços tecnológicos e computacionais, diversas técnicas foram desenvolvidas para otimizar valores e processos. Alguns dos algoritmos de otimização mais utilizados atualmente provém da chamada Computação Natural, que busca estudar padrões encontrados na natureza e replicá-los computacionalmente. Este trabalho utiliza dois algoritmos provenientes diretamente dessa linha de estudos e esse capítulo tem como objetivo clarificar alguns aspectos teóricos importantes para o completo entendimento do texto.

2.1 Biologia Evolutiva

A Teoria da Evolução de Charles Darwin é a alternativa mais aceita pela academia para explicar a criação e continuidade das espécies como é aceita hoje e teve seu nascimento com a publicação de Darwin (1859). Também conhecida como Darwinismo, a teoria explica evolução como uma série de pequenas mudanças aleatórias sofridas pelos indivíduos durante o ato de reprodução, podendo apresentar consequências positivas ou negativas quando expostas ao ambiente. Se positivas permitem que o indivíduo prospere e passe essa mudanças para seus descendentes, enquanto mudanças negativas dificultam sua sobrevivência e reprodução.

2.1.1 Gene

São as unidades funcionais da hereditariedade, constituídos de segmentos de DNA. Os tipos de gene e suas diferentes organizações são os responsáveis pelas características variadas presentes nos seres vivos (SNUSTAD; SIMMONS, 2013).

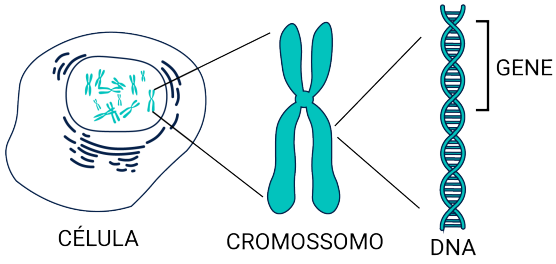
2.1.2 Cromossomo

Um conjunto de genes, organizados em cadeia (Figura 1). Cromossomos podem ser diferentes entre si e possuírem conjuntos diferentes de genes. Cromossomos que tenham genes iguais são chamados homólogos, enquanto cromossomos com genes diferentes são chamados de heterólogos (SNUSTAD; SIMMONS, 2013).

2.1.3 *Crossover*

Para que haja reprodução sexuada, é necessário que os organismos envolvidos possuam gametas, que irão se unir e dar origem a um novo organismo (MICHOD; LEVIN; EDS, 1987). O processo de de divisão celular responsável pela criação de gametas é chamado meiose: células que originalmente possuíam 2N cromossomos (organizados em pares) se dividem em 4 células de N cromossomos (RIBEIRO, 2013).

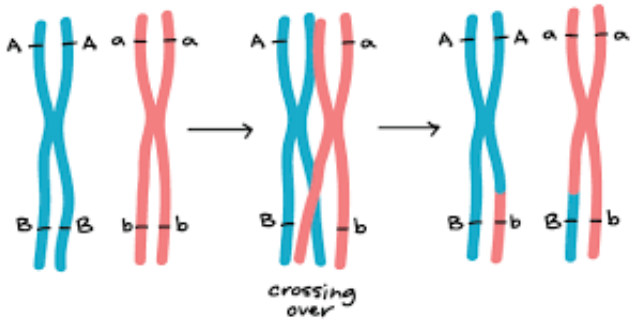
Figura 1 – Representação do Material Genético em uma Célula



Fonte: https://kintalk.org/genetics-101/

Durante uma das etapas da meiose ocorre o cruzamento de dois segmentos dos cromossomos envolvidos, causando troca de material genético entre ambos (Figura 2). O fenômeno foi descrito inicialmente por Creighton e McClintok (1931).

Figura 2 – *Crossing-Over* em um Cromossomo Durante a Meiose



Fonte: http://ecologia.ib.usp.br/evosite/evo101/IIIC3Causes.shtml Traduzido

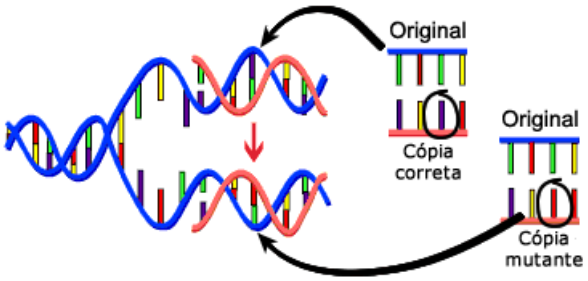
2.1.4 Mutação

Mutações são mudanças na cadeia genética de um organismo, ocorrendo mais precisamente nos nucleotídeos (estruturas químicas que formam o DNA, Figura 3) (MCNAUGHT; WILKINSON, 1987).

Essas mudanças podem ser causadas por exposições a alguns tipos de radiação, erros durante a divisão celular, contaminação viral ou mutagênicos químicos (LEROI, 2005).

O resultado de uma mutação pode gerar consideráveis alterações nas características de um indivíduo, assim como ser tão insignificantes a ponto de nem serem notadas. Portanto é impossível obter uma conclusão sobre uma mutação até que o indivíduo se relacione com o ambiente.

Figura 3 – Exemplo Didático de Mutação



Fonte: https://kintalk.org/genetics-101/

2.2 Algoritmos Evolutivos

Algoritmos Evolutivos (AE) são algoritmos baseados na biologia evolutiva (BECK, 1996) cuja principal função é otimização de problemas mono-objetivos e multiobjetivos. Para isso as possíveis soluções para o problema a ser minimizado são modelados como cromossomos, sendo cada variável um gene, e o grupo de soluções viáveis a serem analisadas como população.

A população inicial é gerada por meio de um algoritmo desenvolvido especificamente para o problema e as populações posteriores por meio de processos virtualmente aleatórios de seleção, mutação e recombinação (BECK, 1996). Essa aleatoriedade perante uma seleção constante permite que o algoritmo, a partir de uma primeira geração suficientemente ampla, passe por uma convergência de resultados até que encontre a solução ótima.

Diversos ramos de pesquisa desenvolveram as técnicas computacionais classificadas como AE, mas este trabalho pretende focar em uma técnica específica: os AG.

2.2.1 Algoritmos Genéticos

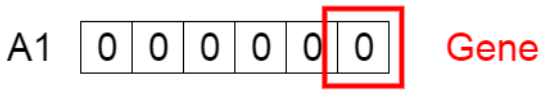
Foram inicialmente propostos por Holland (1992) que considerava a evolução natural um processo simples mas robusto, pois seria capaz de gerar resultados satisfatórios independente dos parâmetros iniciais utilizados (GABRIEL; DELBEM, 2008).

De acordo com Michalewicz (1996) e Zuben (2000) a presença dos operadores de mutação e recombinação nesses algoritmos faz com que equilibrem dois objetivos conflitantes: a exploração do espaço de busca das soluções e o aproveitamento das melhores soluções encontradas (GABRIEL; DELBEM, 2008).

2.2.1.1 Gene

Os genes em um AG são representados pelas variáveis do problema, sendo cada variável presente na solução um gene diferente (Figura 4). A quantidade de genes varia dependendo do problema, assim como a gama de valores que cada um pode assumir.

Figura 4 – Representação de Gene

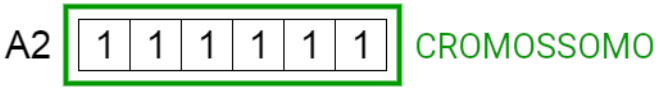


Fonte: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

2.2.1.2 Cromossomo

O conjunto de variáveis utilizadas pelo problema, ao serem agrupadas em um vetor, geram um cromossomo (Figura 5). Cada cromossomo representa uma resposta possível ao problema e podem ser vetores de uma, duas ou múltiplas dimensões.

Figura 5 – Representação de Cromossomo

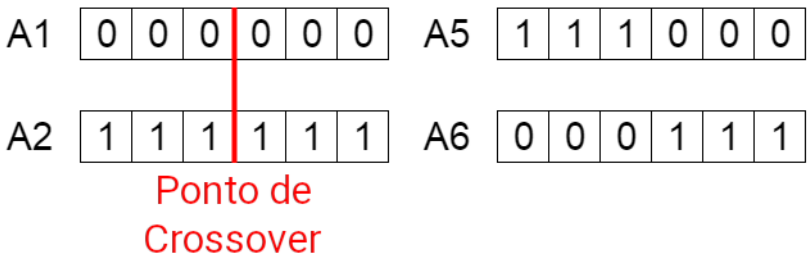


Fonte: baseado em https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

2.2.1.3 *Crossover*

O *crossover*, ou recombinação, é a forma utilizada para gerar um ou mais novos cromossomos a partir de dois escolhidos da população. O objetivo é que a nova geração mescle características da anterior, possibilitando encontrar melhores resultados (Figura 6).

Figura 6 – *Crossover* Genérico



Fonte: baseado em https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

2.2.1.4 Mutação

O processo de mutação consiste em modificar aleatoriamente um indivíduo existente, gerando mudanças inesperadas e facilitando a fuga de ótimos locais (BECK, 1996) (Figura 7).

Figura 7 – Mutação Genérica

Fonte: baseado em https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

2.2.1.5 Pseudo-Código Simplificado de um AG

Algoritmo 1 – Algoritmo Genético - Elaborado pelo Autor

Entrada: Equação Objetivo Ob, Limite Máximo de Gerações L, Número de Individuos por Geração N, Probabilidade de Crossover C Probabilidade de Mutação M

1. ***Criar*** *um vetor de cromossomos* P *vazio*
2. ***Criar*** *um vetor de float* R
3. ***Criar*** N *cromossomos aleatóriamente e* ***inserí-los*** *em* P
4. ***Para cada*** *cromossomo* p *em* P, ***faça***
5. ***Calcular*** *o valor de* Ob *referente a* p *e* ***armazenar*** *no vetor* R
6. ***Criar*** *a variável* $best = 0$
7. ***Encontrar*** *o menor valor de* R *e* ***armazenar*** *em* $best$
8. ***Criar*** *a variável* $temp = 0$
9. ***Criar*** *a variável* $finish = FALSE$
10. ***Criar*** *a variável* $it = 0$
11. ***Enquanto*** $finish == FALSE$ *e* $it < L$
12. ***Para cada*** *par de cromossomos* $p1$ *e* $p2$ *em* p, ***faça***
13. ***Gerar*** *o descendente* $p3$ *(crossover) e* ***armazenar*** *em* P, *com probabilidade* C
14. ***Se*** *não ocorrer crossover*
15. ***Criar*** *o descendente como cópia exata de um dos pais*
16. *Com probabilidade* M *aplicar mutação em* $p3$
17. ***Para cada*** *cromossomo* p *em* P, ***faça***
18. ***Calcular*** *o valor de* Ob *referente a* p *e* ***armazenar*** *no vetor* R
19. ***Descartar*** *os cromossomos* p *em* P *que possuírem os piores valores de* Ob *até restarem* N
20. ***Encontrar*** *o menor valor de* R *e* ***armazenar*** *em* $temp$
21. ***Se*** $temp < best$, ***então***
22. $best = temp$, $it = it + 1$
23. ***Se não***
24. $finish = TRUE$
25. ***Retorna*** $best$

2.2.2 Biased Random-Key Genetic Algorithm

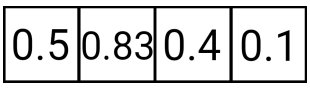
Uma variante do Algoritmo Genético clássico, o Biased Random-Key Genetic Algorithm (BRKGA) foi citado inicialmente na literatura por Gonçalves e Resende (2011). A principal diferença apresentada por esse método é a obtenção de soluções viáveis por meio da decodificação de uma solução codificada. Para isso é necessaria a presença de um decodificador: um método computacional de baixo custo lógico que receba um vetor de valores aleatórios (geralmente valores dentro do intervalo [0, 1[ ) e retorne uma solução viável.

2.2.2.1 Cromossomo

Nesse tipo de algoritmo, o cromossomo não possui os valores a serem associados à cada gene. Ao invés disso, contém os valores entre 0 e 1 associados para cada variável (Figura 8).

Antes de serem submetidos à função *decoder* esses cromossomos não são soluções viáveis.

Figura 8 – Representação de Cromossomo em um BRKGA [1: 0.5, 2: 0.83, 3: 0.4, 4: 0.1]



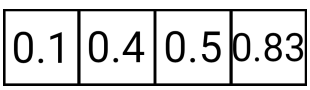
Fonte: Elaborada Pelo Autor

2.2.2.2 *Decoder*

O objetivo dessa função é transformar os cromososomos criados aleatoriamente em soluções viáveis e inúmeras estratégias podem ser utilizadas, desde mais simples como ordenação de vetores até mais complexas.

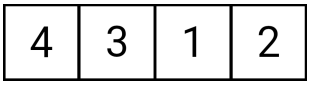
Utilizando a Figura 8 como exemplo e aplicando uma função decoder de ordenação, seria obtida uma nova ordem (Figura 9) e então a função retornaria o vetor da Figura 10.

Figura 9 – Representação de Cromossomo em um BRKGA [4: 0.1, 3: 0.4, 1: 0.5, 2: 0.83]



Fonte: Elaborada pelo Autor

Figura 10 – Nova ordem de genes referente a Figura 9



Fonte: Elaborada pelo Autor

2.2.2.3 Mutação

Ao invés de utilizar alguma estratégia para modificar um cromossomo existente a mutação no BRKGA consiste na criação de um novo cromossomo, que substituirá algum previamente existente.

2.2.3 *Layout* de Estaleiros

A otimização do *layout* de estaleiros é pouco explorada pela academia, tendo uma quantidade limitada de artigos publicados. Choi, Kim e Chung (2017) em seu artigo apresentam o modelo de duas etapas para a solução desse tipo de problema, pois a divisão gera dois problemas menos complexos e por consequência de mais fácil resolução.

O modelo de dois estágios em questão foi baseado no trabalho de Nick (2008), tendo como inovações a simplificação da função objetivo e a adição das restrições de alinhamento.

O primeiro estágio consiste em uma simplificação do terreno disponível, de modo a representá-lo como uma matriz m x n sendo cada espaço da matriz a posição de um departamento (é necessário que a matriz contenha exatamente o número de departamentos do problema, sem espaços vazios). O AG será responsável por encontrar a melhor ordem possível de organização dos departamentos para que o custo de produção seja o menor possível.

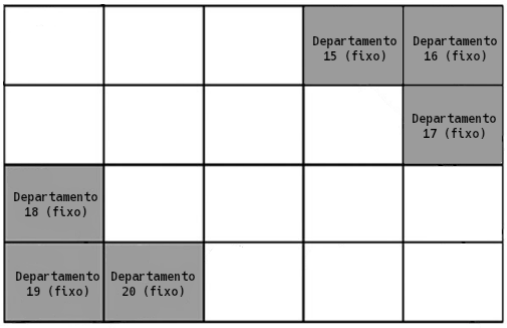
No segundo estágio, que não será objetivo de estudo deste trabalho, o indivíduo ótimo obtido no estágio anterior é aplicado em um grid M x N com uma escala muito mais detalhada do terreno real. Iniciando com áreas unitárias, um algoritmo de crescimento estocástico faz com que os departamentos cresçam a cada iteração até atingirem a área necessária.

Os departamentos neste trabalho, como nos anteriores (NICK, 2008) (CHOI; KIM; CHUNG, 2017), estão sujeitos a um tipo de restrição em relação à topologia (posição fixa), um em relação à geometria (dimensões fixas) e dois em relação à proximidade (adjacência e alinhamento).

As restrições de dimensão fixa e adjacência são aplicados apenas na segunda etapa do sistema, na qual são calculadas as dimensões dos departamentos, portanto não são importantes neste estudo. No primeiro estágio, todas as restrições de adjacência são consideradas como restrições de alinhamento.

Departamentos com posição fixa possuem suas coordenadas pré-definidas e não podem ser alocados em outros locais (Figura 11).

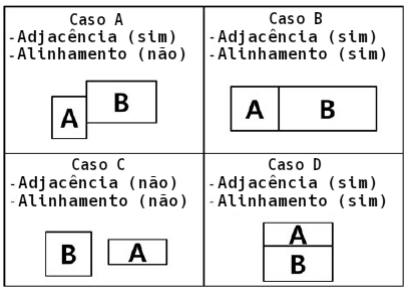
Figura 11 – Departamentos Fixos



Fonte: Choi, Kim e Chung (2017) (adaptado e traduzido)

Departamentos com restrições de adjacência são sempre associados em duplas e precisam estar diretamente em contato um com o outro: uma de suas faces deve ter a mesma dimensão e estar em contato direto com uma das faces da sua dupla (Figura 12).

Figura 12 – Alinhamento x Adjacência



Fonte: Choi, Kim e Chung (2017) (traduzido)

3 Ferramentas e Tecnologias

3.1 MATLAB

O software MATLAB, cuja sigla significa Matrix Laboratory, teve sua criação no fim dos anos 70 pelo presidente do departamento de Ciências da Computação da Universidade do Novo México da época, Cleve Moler e foi rapidamente disseminado entre universidades.

Em 1983, ao visitar a Universidade de Stanford, Moler conheceu o engenheiro Jack Little, que viu grande potencial na linguagem. Os dois então se juntaram a Steve Bangert reescreveram MATLAB em C, e logo em seguida fundou a empresa MathWorks.

A linguagem utilizada pelo programa, também chamada de MATLAB, M-code ou apenas M, é própria e foi o ponto de partida para o desenvolvimento do *software*. É uma linguagem de multiplos paradigmas.

Todas as variáveis são criadas utilizando o operador $=$, possuindo tipos implícitos e facilmente modificados. Uma variável pode receber novos valores a partir de constantes, de outras variáveis ou de funções.

Outra característica que torna o manuseio das variáveis mais prático é que todas são consideradas vetores, podendo ter um valor vazio ou unitário (vetor 1x1) ou múltiplos valores, como de um vetor tradicional.

As funções na linguagem M podem retornar e receber inúmeras variáveis e quando são chamadas não precisam apresentar todos os argumentos presentes na sua definição, desde que esta variável não seja utilizada durante sua execução.

Seus códigos podem ser escritos utilizando o Paradigma Procedural ou o Paradigma Orientado a Objetos (POO), permitindo a criação de Classes, Eventos e *listeners*. O programa também aceita arquivos em C++, C#, e Java.

O *software* não se destaca pela sua velocidade de execução, sendo consideravelmente mais lento que linguagens como C++, Python ou C. Entretanto, é um sistema robusto com inúmeras funções matemáticas imbutidas. Isso faz com que se torne muito utilizados em setores de engenharia e pesquisa pois facilita o processo de teste e criação de sistemas mais complexos.

Por essas características o MATLAB foi escolhido para o desenvolvimento dos algoritmos deste trabalho.

3.2 GA_framework

Para facilitar e acelerar o desenvolvimento do código foi utilizado um *framework* que entrega uma estrutura genérica para a execução de algoritmos genéticos, exigindo do usuário a criação dos códigos específicos para cada modalidade do problema.

O *framework* de Freitas (2012) utilizado está disponível da plataforma de usuários da MathWorks, podendo ser utilizado gratuitamente.

4 Método

O trabalho foi dividido em três estágios: o primeiro constituído pelo desenvolvimento de um código baseado no trabalho de Choi, Kim e Chung (2017), o segundo pelo desenvolvimento de um novo algoritmo e o último pela comparação e estudo dos resultados.

Ambos os algoritmos tem como objetivo minimizar a Equação 4.1, desenvolvida por Choi, Kim e Chung (2017).

$$\sum_{i \epsilon D} \sum_{j \epsilon D} f_{i,j}^{tp} c_{i,j}^{tp} d_{i,j}^{tv} \tag{4.1}$$

Sendo:

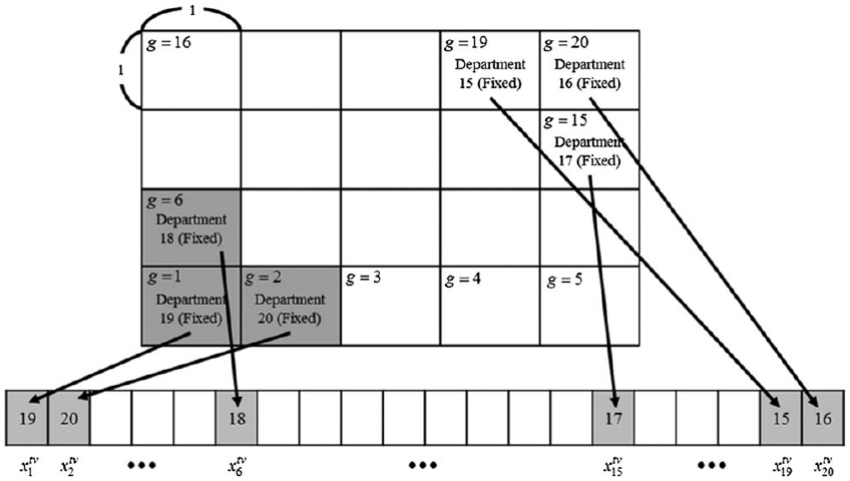
$$f_{i,j}^{tp} \quad \text{Fluxo de Materiais Entre Os Departamentos (Tabela 1)} \tag{4.2}$$

$$c_{i,j}^{tp} \quad \text{Custo De Transporte Entre Dois Departamentos (Tabela 2)} \tag{4.3}$$

$$d_{i,j}^{tv} \quad \text{Distância Euclidiana Entre Dois Departamentos} \tag{4.4}$$

Todo este estudo, por ter como objetivo comparar os resultados obtidos por Choi, Kim e Chung (2017), utiliza como os dados fornecido pelos autores, presentes nas Tabelas 1, 2, 3 e na Imagem 13.

Figura 13 – Posições Fixas Dos Departamentos 17, 18, 19 e 20



Fonte: Choi, Kim e Chung (2017)

Tabela 1 – Fluxo Entre Os Departamentos - Fonte: Choi, Kim e Chung (2017)

Departamento De Partida	Departamento Destino	Quantidade (ton.)
3	4	1860
4	5	1200
4	7	660
5	8	800
5	9	95
5	10	48
7	9	660
8	7	4
8	9	670
8	10	120
9	10	1300
10	11	700
10	13	300
11	12	600
12	14	600
13	11	300
14	15	600

Tabela 2 – Custo De Transporte Entre Os Departamentos - Fonte: Choi, Kim e Chung (2017)

Número Do Departamento	1	2	3	4	5	...	20
1	0	1	1	1	1	...	1
2	1	0	1	1	1	...	1
3	1	1	0	1	1	...	1
4	1	1	1	0	1	...	1
5	1	1	1	1	0	...	1
...	...	...	...	...	...	...	...
20	1	1	1	1	1	...	0

4.1 Primeiro Estágio

No estágio inicial do projeto, o trabalho publicado por (CHOI; KIM; CHUNG, 2017) foi estudado e analisado para que fosse possível a replicação de seu algoritmo com a maior fidelidade possível. Apesar do ótimo resultado exposto pelo trabalho, os autores não forneceram algoritmos ou explicações detalhadas sobre o processo de desenvolvimento do seu algoritmo, dificultando a sua replicação.

Portanto a estratégia utilizada por este estudo foi o desenvolvimento de um algoritmo genético mais genérico possível, fazendo apenas as adaptações realmente necessárias para seu funcionamento.

Tabela 3 – Restrições De Proximidade Dos Departamentos - Fonte: Choi, Kim e Chung (2017)

Departamento	Departamento
3	4
4	5
4	7
5	9
7	9
9	10
11	12

4.1.1 Criação de Indivíduos

A função de criação de indivíduos deriva de uma lógica completamente aleatória mas precisou passar por ajustes para se adaptar aos requerimentos do problema, como restrições de posição e proximidade (Algoritmo 2).

Algoritmo 2 – Criação de Indivíduos - Elaborado Pelo Autor

Entrada: Lista de Restrições r

1. ***Criar*** *um vetor vazio* ind
2. ***Inserir*** *em* ind *os departamentos com restrições de centroid fixo, se houver algum, em suas respectivas posições*
3. ***Inserir*** *em* ind *todos os departamentos com restrições de agrupamento, em posições aleatórias que respeitem essas restrições, se houver algum*
4. ***Inserir*** *os departamentos restantes, se houver algum, nas posições que ainda não foram ocupadas, de maneira aleatória*

4.1.2 *Crossover*

Por se tratar de um vetor de números inteiros, sem repetições e entre um intervalo específico (1 - 20) métodos convencionais de *crossover* não são aplicáveis.

Um *crossover* de um ponto só, por exemplo, poderia selecionar um departamento duas vezes no mesmo filho, bastando que os pais possuíssem uma cópia do mesmo em cada parte selecionada.

Foi utilizada portanto a técnica de *Crossover* em Ciclo (Algoritmo 3), designada para atuar em problemas com cromossomos ordenados.

Algoritmo 3 – *Crossover* em Ciclo - Adaptado de RubiciteInteractive (2008)

Entrada: Indivíduo $p1$ e Indivíduo $p2$

1. ***Selecionar*** *randomicamente um gene de* $p1$ *e copiar para o novo indivíduo* c *na mesma posição.*
2. ***Armazenar*** *o valor* $v1$ *do gene e seu índice* i
3. ***Procurar*** *o gene de* $p2$ *com o valor* $v1$*. Armazenar sua posição* i
4. ***Procurar*** *em* $p1$ *o gene de valor* $v2$ *na posição* i
5. ***Se*** $v2$ *for igual a* $v1$
6. ⌊ *Sai do Loop*
7. ***Se*** *não*
8. ⌈ ***Armazenar*** $v2$ *na posição* i *em* c
9. ⌈ $v2$ *se torna* $v1$
10. ⌊ ***Voltar*** *ao Passo 3*
11. ***Copiar*** *de* $p2$ *os genes restantes para completar* c
12. ***Retorna*** c

4.1.3 Mutação

De modo similar ao *crossover*, e devido aos mesmos fatores, uma técnica simples não funcionaria no cromossomo do problema, por se tratarem de genes com um espaço amostral de valores amplo.

Foi utilizada então a técnica *Swap*, que consiste em selecionar dois genes aleatórios do indivíduo e invertê-los. Entretanto é importante notar que os genes selecionados não devem possuir restrições de posição.

Algoritmo 4 – Mutação *Swap* - Elaborado pelo Autor

Entrada: Indivíduo p

1. ***Selecionar*** *randomicamente um gene* $g1$ *de* p
2. ***Enquanto*** $g1$ *possuir restrição*
3. ⌊ ***Selecionar*** *randomicamente um gene* $g1$ *de* p
4. ***Selecionar*** *randomicamente um gene* $g2$ *de* p
5. ***Enquanto*** $g2$ *possuir restrição*
6. ⌊ ***Selecionar*** *randomicamente um gene* $g2$ *de* p
7. ***Inverter*** *os valores entre* $g1$ *e* $g2$
8. ***Retorna*** p

O código foi completamente desenvolvido na plataforma MATLAB e, após finalizado, submetido uma bateria de testes utilizando as mesmas entradas utilizadas por (CHOI; KIM; CHUNG, 2017) de modo a confirmar sua fidelidade e permitir comparações.

Os testes são constituídos pela simples execução do algoritmo, garantindo que seu resultado ótimo se mantenha coerente com o fornecido pelos seus autores. Com as informações obtidas (listadas na Tabela 4) o primeiro estágio é encerrado, dando início ao segundo

Tabela 4 – Informações armazenadas - Fonte: Elaborada pelo Autor

1	Probabilidade de *crossover*
2	Probabilidade de Mutação
3	Quantidade de Indivíduos
4	Tempo de Execução
5	Valor Ótimo

4.2 Segundo Estágio

Devido ao teor de inovação da proposta e consequente falta de trabalhos anteriores ligando a técnica BRKGA ao problema utilizado neste estudo, as técnicas utilizadas nos algoritmos desta etapa foram desenvolvidas pelo autor especificamente para o problema abordado.

4.2.1 Criação de Indivíduos

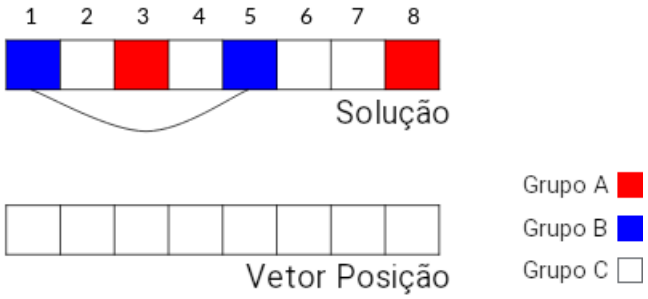
Assim como no primeiro estágio, foi necessário que houvesse uma adaptação na lógica da criação de indivíduos para que se tornasse viável ao problema.

Nessa versão, os departamentos são divididos em três grupos: os que possuem posição fixa (grupo A), os que possuem restrições de proximidade com outros departamentos (grupo B) e os restantes, que não possuem nenhum tipo de restrição (grupo C). São criados também dois vetores: o vetor *solution* representará o novo indivíduo, e possuirá apenas índices entre 0 e 1, e o vetor *position* representará a posição dos departamentos no *layout* (Figura 14).

Inicialmente cada departamento do grupo C recebe um índice aleatório, armazenado em seu respectivo gene no vetor *solution* (Figura 15). Logo após são inseridos em *position* todos os departamentos do grupo A em suas respectivas posições obrigatórias e seus genes no vetor *solution* recebem o índice -1 (Figura 16). Por fim, são inseridos os departamentos do grupo B no mesmo vetor, em posições aleatórias que respeitem as restrições de proximidade (Figura 17).

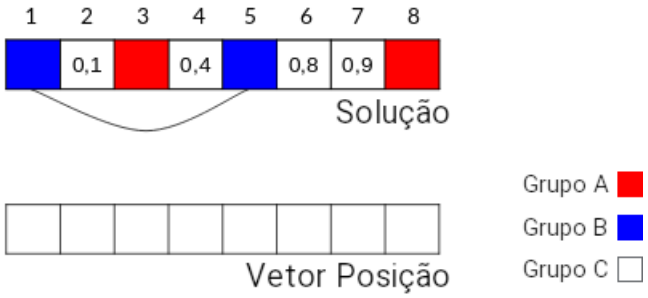
Nesse momento, são adicionados aleatóriamente os departamentos do grupo C nos espaços vazios de *position* (Figura 18). A partir dessa configuração, o índice dos departamentos

Figura 14 – Ínicio do Algoritmo de Criação de Indivíduos



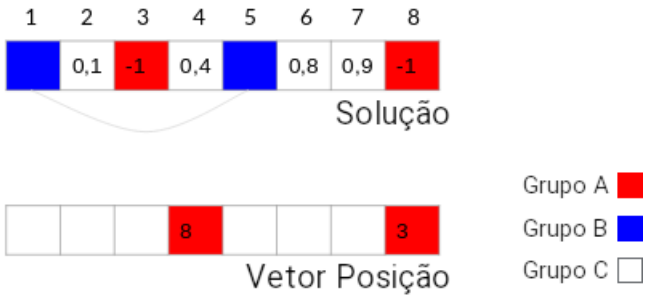
Fonte: Elaborada pelo Autor

Figura 15 – Algoritmo de Criação de Indivíduos - Passo 2



Fonte: Elaborada pelo Autor

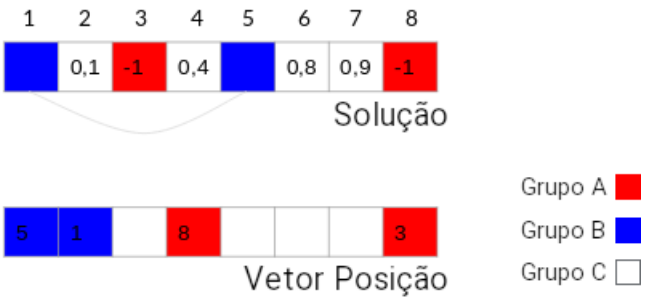
Figura 16 – Algoritmo de Criação de Indivíduos - Passo 3



Fonte: Elaborada pelo Autor

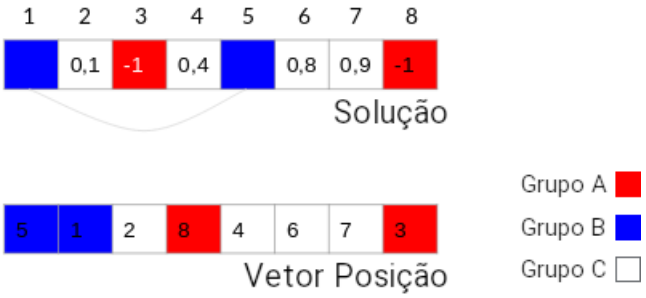
restantes é calculado, de modo que ao passarem por ordenação em ordem crescente gerem a mesma ordem presente em *position* (Figura 19).

Figura 17 – Algoritmo de Criação de Indivíduos - Passo 4



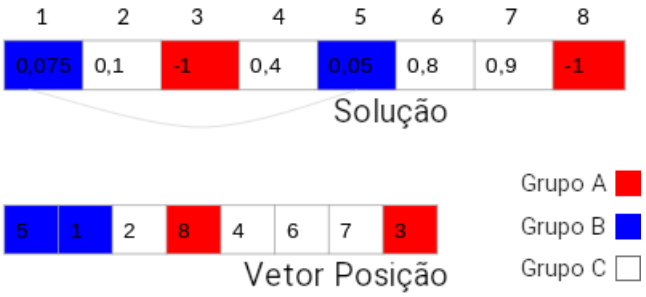
Fonte: Elaborada pelo Autor

Figura 18 – Algoritmo de Criação de Indivíduos - Passo 5



Fonte: Elaborada pelo Autor

Figura 19 – Algoritmo de Criação de Indivíduos - Passo 6



Fonte: Elaborada pelo Autor

4.2.2 *Crossover*

O algoritmo de *Crossover* utilizado nessa versão do algoritmo foi baseado no *Crossover* Uniforme Viciado (CUV). Essa técnica consiste em analisar individualmente cada gene do novo cromossomo e rolar uma probabilidade para decidir de qual pai virá o valor do mesmo. Essa probabilidade sendo maior em favor do cromossomo com melhor valor objetivo.

Para os departamentos sem nenhuma restrição, a função funciona da mesma maneira que um CUV padrão e para os departamentos com posição fixa basta copiá-los na mesma posição,

pois estarão sempre no mesmo gene. A mudança no algoritmo acontece quando referente aos departamentos com restrições de proximidade. Esses departamentos são escolhidos como um grupo e não individualmente, ou seja, se o cromossomo novo receber um gene deste tipo do primeiro pai, todos os outros genes do mesmo tipo virão deste mesmo progenitor.

4.2.3 Decoder

O decoder utilizado no algoritmo é uma função de ordenação crescente, com a simples adaptação de manter os genes com índices -1 em suas posições originais, movendo apenas os genes restantes.

Após a finalização do desenvolvimento desse novo código, a bateria de testes previamente mencionada será repetida de modo a adquirir todas as informações necessárias para comparação (Tabela 4).

4.3 Terceiro Estágio

O terceiro estágio é constituído pelo estudo e comparação desses dois agrupamentos de resultado. Foram produzidos gráficos e tabelas que permitam a visualização do avanço alcançado, seja na otimização do resultado ou na diminuição de tempo de execução.

5 Resultados

Neste Capítulo serão listados os dados adquiridos nas etapas de teste dos dois estágios do trabalho, que serão analisados no Capítulo 6.

Os algoritmos foram executados em 4 grupos de teste, referentes ao tamanho da população utilizada (500, 1.000, 5.000 e 10.000 indivíduos).

Para cada grupo de testes foram feitos 400 experimentos,de modo a criar um espaço amostral que possibilitaria uma análise completa dos algoritmos.

5.1 Resultado de Choi, Kim e Chung (2017)

No artigo utilizado como base não são fornecidos dados como o tempo de execução médio do algoritmo ou o valor ótimo do melhor indivíduo para serem usados como comparativo. Entretanto, é possível encontrar o segundo item utilizando o indivíduo indicado como ótimo pelos autores (Figura 20).

Figura 20 – Melhor Indivíduo Encontrado por Choi, Kim e Chung (2017)

Fonte: Choi, Kim e Chung (2017)

Aplicando a função objetivo no indivíduo em questão constata-se que seu valor ótimo é de 11.819 unidades. Com esse valor é possível comparar a qualidade dos algoritmos testados.

5.2 Resultados do Primeiro Estágio

Tabela 5 – Dados Obtidos Perante Execução Do Algoritmo AG 400 Vezes - Fonte: Elaborada pelo Autor

Tamanho da População	**Probabilidade de *Crossover***	**Probabilidade de Mutação**	**Tempo Médio de Execução (s)**	**Melhor Valor (u.m.)**
500	0,9	0,05	1.922,8	11.077
1000	0,9	0,05	3.723,2	10.947
5000	0,9	0,05	16.364	10.947
10000	0,9	0,05	37.808	10.947

5.2.1 Valores Ótimos

Ao ser executado com a menor população estipulada pelos testes (500 indivíduos) o primeiro algoritmo se apresentou capaz de alcançar uma solução viável e inferior a apresentada por Choi, Kim e Chung (2017) (Figura 21).

Com o primeiro aumento da população houve uma melhora na resposta obtida, reduzindo de 11.077 unidades para 10.947 unidades (Figura 22).

Nos aumentos posteriores, entretanto, nota-se que o valor dos indivíduos ótimos se estabilizou e deixou de alcançar melhora (Figuras 23 e 24).

Figura 21 – Melhor Indivíduo Encontrado Pelo AG (500 Indivíduos, Valor Ótimo: 11.077 Unidades)

1	12	14	15	16
13	11	8	2	17
18	10	9	5	6
19	20	7	4	3

Fonte: Elaborada pelo Autor

Figura 22 – Melhor Indivíduo Encontrado Pelo AG (1000 Indivíduos, Valor Ótimo: 10.947 Unidades)

1	12	14	15	16
2	11	10	8	17
18	13	9	5	6
19	20	7	4	3

Fonte: Elaborada pelo Autor

Figura 23 – Melhor Indivíduo Encontrado Pelo AG (5000 Indivíduos, Valor Ótimo: 10.947 Unidades)

2	12	14	15	16
6	11	10	8	17
18	13	9	5	1
19	20	7	4	3

Fonte: Elaborada pelo Autor

Figura 24 – Melhor Indivíduo Encontrado Pelo AG (10000 Indivíduos, Valor Ótimo: 10.947 Unidades)

1	12	14	15	16
6	11	10	8	17
18	13	9	5	2
19	20	7	4	3

Fonte: Elaborada pelo Autor

5.2.2 Tempo de Execução

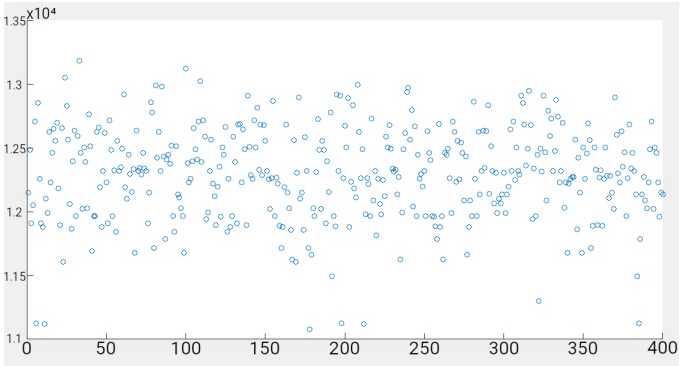
Como pode ser observado na Tabela 5 a progressão do Tempo de Execução do algoritmo acontece em um crescimento linear, crescendo com o aumento do tamanho da população do grupo de teste.

5.2.3 Estabilidade

As Figuras 25, 26, 27 e 28 exibem os valores ótimos (Eixo Y) adquiridos ao final de cada experimento dos agrupamentos de teste (Eixo X), permitindo a visualização da estabilidade do algoritmo e o cálculo do seu desvio padrão.

Este valor nos permite inferir quão frequentemente o algoritmo é capaz de alcançar um bom resultado, sendo menor a medida que a estabilidade dos valores aumenta.

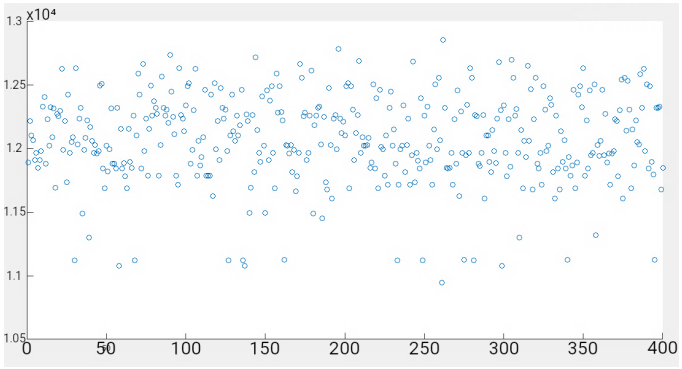
Figura 25 – Valor Dos Indivíduos Resultantes (500 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 395,5)



Fonte: Elaborada pelo Autor

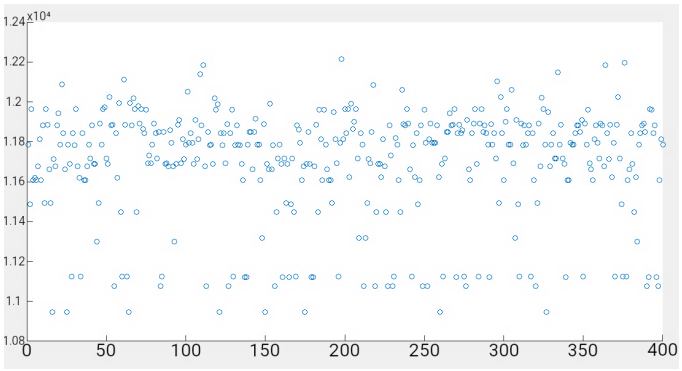
É possível concluir, portanto, que a estabilidade do AG é proporcional ao aumento da população utilizada.

Figura 26 – Valor Dos Indivíduos Resultantes (1000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 342,6)



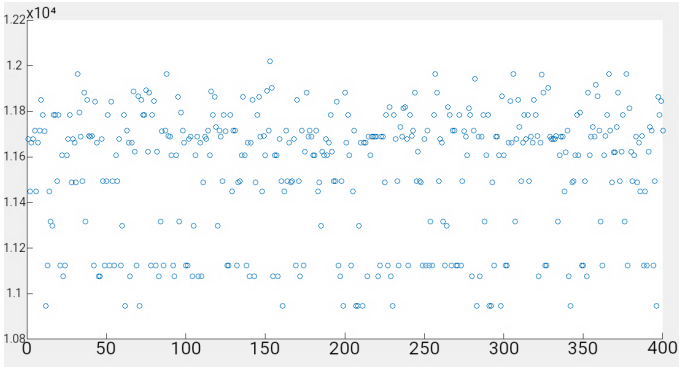
Fonte: Elaborada pelo Autor

Figura 27 – Valor Dos Indivíduos Resultantes (5000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 282,1)



Fonte: Elaborada pelo Autor

Figura 28 – Valor Dos Indivíduos Resultantes (10000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 278)



Fonte: Elaborada pelo Autor

Tabela 6 – Dados Obtidos Perante Execução Do Algoritmo BRKGA 400 Vezes

Tamanho da População	**Probabilidade de *Crossover***	**Probabilidade de Mutação**	**Tempo Médio de Execução (s)**	**Melhor Valor (u.m.)**
500	0,9	0,05	1.989,4	11.882
1000	0,9	0,05	2.030	10.947
5000	0,9	0,05	24.607	10.947
10000	0,9	0,05	29.066	10.947

5.3 Resultados do Segundo Estágio

5.3.1 Valores Ótimos

Em contrapartida ao algoritmo anterior, o BRKGA não foi capaz de encontrar um indivíduo com valor menor ou igual do apresentado por Choi, Kim e Chung (2017) (Figura 29).

Figura 29 – Melhor Indivíduo Encontrado Pelo BRKGA (500 Indivíduos) - Fonte: Elaborada pelo Autor

4	3	1	15	16
7	5	8	14	17
18	9	10	12	2
19	20	13	11	6

Fonte: Elaborada pelo Autor

No testes posteriores, por outro lado, o algoritmo se mostrou eficiente e alcançou os mesmos resultados exibidos pelo anterior (Figuras 30, 31, 32).

Figura 30 – Melhor Indivíduo Encontrado Pelo BRKGA (1000 Indivíduos)

1	12	14	15	16
6	11	10	8	17
18	13	9	5	2
19	20	7	4	3

Fonte: Elaborada pelo Autor

5.3.2 Tempo de Execução

Observando a Tabela 6 percebe-se que o tempo médio de execução do BRKGA apresentou um comportamento menos linear se comparado ao primeiro algoritmo. Apesar de

Figura 31 – Melhor Indivíduo Encontrado Pelo BRKGA (5000 Indivíduos)

1	12	14	15	16
2	11	10	8	17
18	13	9	5	6
19	20	7	4	3

Fonte: Elaborada pelo Autor

Figura 32 – Melhor Indivíduo Encontrado Pelo BRKGA (10000 Indivíduos)

2	12	14	15	16
6	11	10	8	17
18	13	9	5	1
19	20	7	4	3

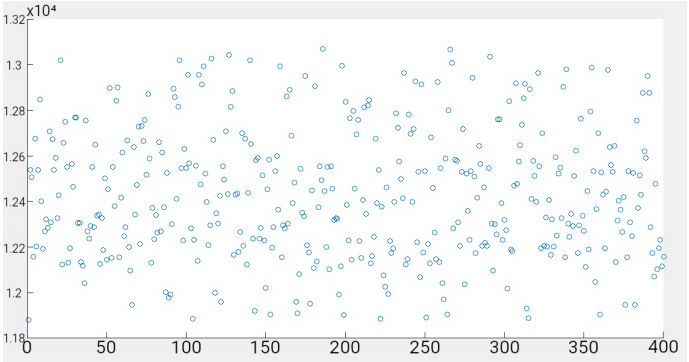
Fonte: Elaborada pelo Autor

ainda crescente, o aumento foi ligeiramente menos acentuado a partir dos 5.000 indivíduos, indicando uma escalabilidade superior ao AG.

5.3.3 Estabilidade

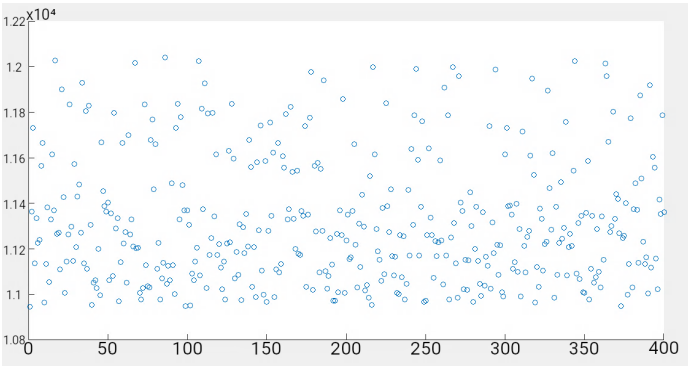
Assim como no AG, o desvio padrão do segundo algoritmo sofre uma diminuição proporcional ao aumento da população utilizada. Entretanto o BRKGA apresenta valores consideravelmente menores ao do algoritmo anterior (Figura 33, 34, 35, 36), se mostrando mais estável.

Figura 33 – Valor Dos Indivíduos Resultantes (500 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 284,2)



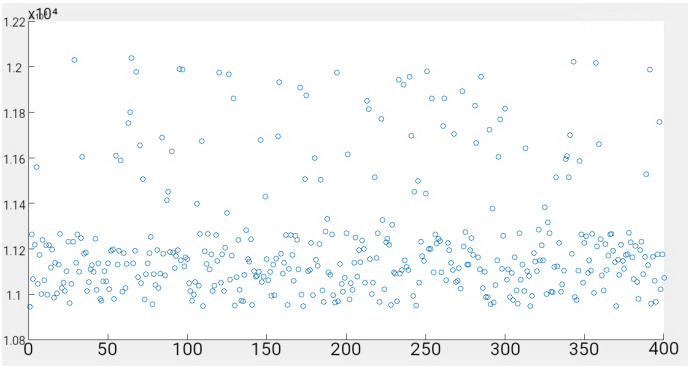
Fonte: Elaborada pelo Autor

Figura 34 – Valor Dos Indivíduos Resultantes (1000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 280)



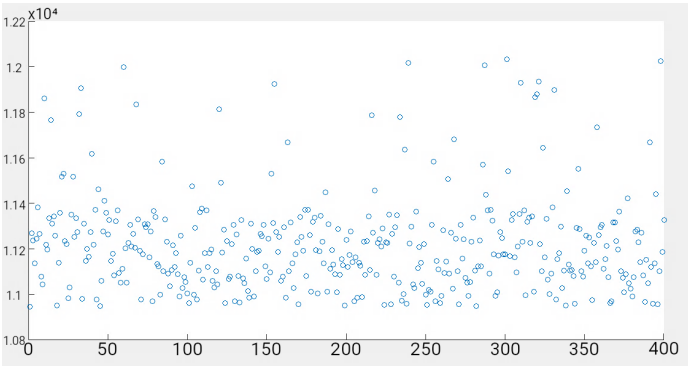
Fonte: Elaborada pelo Autor

Figura 35 – Valor Dos Indivíduos Resultantes (5000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 261,2)



Fonte: Elaborada pelo Autor

Figura 36 – Valor Dos Indivíduos Resultantes (10000 indivíduos, Valor Ótimo (y) x Experimento (x), Desvio padrão: 214,5)



Fonte: Elaborada pelo Autor

6 Conclusão

Ambos os algoritmos conseguiram atingir resultados com valor inferior ao exposto pelo artigo utilizado como base para esse trabalho (CHOI; KIM; CHUNG, 2017). Esse fato por si só é o suficiente para comprovar a capacidade de ambos os algoritmos de resolver o problema em questão e garantir a importância do estudo feito.

O trabalho, todavia, se propõe a concluir qual deles o faz com melhor desempenho e para isso é necessário estipular critérios para a seleção.

Os critérios escolhidos pelo autor são:

- Encontrar o ponto ótimo global,
- Utilizar o menor número de indivíduos possível,
- Gastar o menor tempo possível.

Os dois algoritmos se mostram limitados quando utilizados com populações de tamanho reduzido, apresentando valores facilmente superados em testes posteriores. O AG, entretanto, aparenta sofrer menos nestas situações, conseguindo já nesta etapa um resultado inferior ao de Choi, Kim e Chung (2017) (11.077 unidades), demonstrando uma evolução em relação à sua versão original.

Mas a partir de uma população de 1000 indivíduos, ambos os algoritmos se igualam, alcançando o valor mínimo de 10.947 unidades.

Portanto ambos os algoritmos se encontram equivalentes nos dois primeiros critérios, pois ambos alcançaram o ótimo global utilizando a mesma dimensão de indivíduos.

Resta a análise, portanto, do terceiro critério: velocidade. O tempo médio de execução do AG, com 1000 indivíduos, é de 3.723,2 segundos (pouco mais de 1 hora) enquanto o tempo médio de execução do BRKGA, também com 1000 indivíduos, é de 2.030 segundos (aproximadamente 34 minutos).

Mediante os fatos, é clara a superioridade do algoritmo BRKGA em relação ao AG quando aplicado ao problema apresentado por Choi, Kim e Chung (2017), uma aplicação inédita para o problema.

Este trabalho e sua conclusão demonstram com sucesso a necessidade atual de exploração dessa área de pesquisa, com a utilização de novos métodos de otimização e com o avanço de métodos já utilizados.

Referências

AIELLO, G.; ENEA, M.; GALANTE, G. M. A multi-objective approach to facility layout problem by genetic research algorithm and electre method. *Robotics and Computer-Integrated Manufacturing*, v. 22, n. 5, p. 447–455, 2006.

AZADIVAR, F.; WANG, J. Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, v. 38, n. 18, p. 4369–4383, 2000.

BECK, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. [S.l.]: Oxford University Press, Inc, 1996. ISBN 0195099710.

CHOI, M.; KIM, S. H.; CHUNG, H. Optimal shipyard facility layout planning based on a genetic algorithm and stochastic growth algorithm. *Ships and Offshore Structures*, 2017.

CREIGHTON, H. B.; MCCLINTOK, B. A correlation of cytological and genetical crossing-over in zea mays. *Proceedings of the National Academy of Science of the United States of America*, v. 17, n. 8, p. 492–497, 1931.

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

FREITAS, A. de. *Open Genetic Algorithm Toolbox*. 2012. Disponível em: <<https://www.mathworks.com/matlabcentral/fileexchange/37998-open-genetic-algorithm-toolbox>>.

GABRIEL, P. H. R.; DELBEM, A. C. B. *Fundamentos de algoritmos evolutivos*. [S.l.]: ICMC-USP, 2008.

GONçALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, p. 487–525, 2011.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. [S.l.]: MIT Press Cambridge, MA, USA, 1992. ISBN 0262082136.

LEROI, A. M. *Mutants: on genetic variety and the human body*. [S.l.]: Penguin, 2005.

MATSON, J. O.; MELLICHAMPH, J. M.; SWAMINATHAN, S. R. Excite: expert consultant for in- plant transportation equipment. *International Journal of Production Research*, v. 30, n. 8, p. 1969–1983, 1992.

MCNAUGHT, A. D.; WILKINSON, A. *Compendium of Chemical Terminology*. [S.l.]: International Union of Pure and Applied Chemistry, 1987.

MICHALEWICZ, Z. *Genetic algorithms + data structures = evolution programs*. 3. ed. [S.l.]: Springer-Verlag Berlin, Heidelberg, 1996. ISBN 3540606769.

MICHOD, R. E.; LEVIN, B. R.; EDS. The evolution of sex. an examination of current ideas. *Science*, v. 240, n. 4855, p. 1072 – 1073, 1987.

NICK, E. K. Fuzzy optimal allocation and arrangement of spaces in naval surface ship design. *Tese (PhD)*, University of Michigan, 2008.

PARSONS, M.; CHUNG, H.; NICK, E. K.; DANIELS, A.; LIU, S. U.; PATEL, J. Intelligent ship arrangements: a new approach to general arrangement. *Naval Engineers Journal*, 2008.

RIBEIRO, K. D. K. da F. *Meiose*. [S.l.]: Brasil Escola, 2013. Disponível em: https://brasilescola.uol.com.br/biologia/meiose.htm. Acesso em 04 de outubro de 2019.

RUBICITEINTERACTIVE. *Cycle Crossover Operator Tutorial*. 2008. Disponível em: <<http://www.rubicite.com/Tutorials/GeneticAlgorithms/CrossoverOperators/CycleCrossoverOperator.aspx>>.

SNUSTAD, D. P.; SIMMONS, M. J. *Fundamentos de Genética*. [S.l.]: Editora Guanabara Koogan, 2013.

YOUNG, J. S.; KWANG, K. L.; DONG, K. L.; IN, H. H.; JONG, H. W.; JONG, G. S. Development of a design framework for simulation based shipyard layout. *Journal of the Society of Naval Architects of Korea*, v. 45, p. 202–212, 2008.

ZUBEN, F. J. V. Computação evolutiva: Uma abordagem pragmática. 2000.