UNIVERSIDADE DE SÃO PAULO
FFCLRP-DEPARTAMENTO DE COMPUTAÇÃO E MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

“Operadores de Recombinação por Decomposição para
Otimização Pseudo-Booleana”

Diogenes Laertius Silva de Oliveira Filho

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Ribeirão Preto-SP

2019

Diogenes Laertius Silva de Oliveira Filho

“Operadores de Recombinação por Decomposição para
Otimização Pseudo-Booleana”

Versão Corrigida

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Orientador: Renato Tinós

Ribeirão Preto–SP

2019

Diogenes Laertius Silva de Oliveira Filho

Operadores de Recombinação por Decomposição para Otimização Pseudo-Booleana. Ribeirão Preto–SP, 2019.

77p. : il.; 30 cm.

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências,
Área: Computação Aplicada.

Orientador: Renato Tinós

1. Otimização Combinatória 2. Inteligência Artificial. 3. Algoritmos Genéticos.

Diogenes Laertius Silva de Oliveira Filho

“Operadores de Recombinação por Decomposição para
Otimização Pseudo-Booleana”

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Trabalho aprovado. Ribeirão Preto–SP, 24 de Janeiro de 2019:

Professor Orientador

Renato Tinós

Professor Convidado

André Ponche de Leon

Professor Convidado

Alexandre Delbem

Professor Convidado

Danilo Sanches

Ribeirão Preto–SP

2019

*Transmita o que aprendeu. Força, mestria.
Mas fraqueza, insensatez, fracasso também.
Sim, fracasso acima de tudo. O maior professor, o fracasso é.
Luke, nós somos o que eles crescem além.
Esse é o verdadeiro fardo de todos os mestres.
(Yoda, Star Wars: Os Últimos Jedi)*

Agradecimentos

Aqui jaz meus sinceros agradecimentos a todos que de alguma forma contribuíram para essa pesquisa de mestrado, fruto de muito estudo, dedicação, abdicção, noites de sono em claro e muito aprendizado com os erros no caminho.

A minha família, Diogenes Laertius, Janaina Bernardino, Jayne Karla e Donnie Oliveira, pelo apoio incondicional em horas de alegria e de aflição, sem o qual não haveria conseguido superar as adversidades encontradas nos caminhos que trilhei.

Ao meu orientador, Prof. Dr. Renato Tinós, pela paciência, orientação, amizade e enorme apoio dedicado à este trabalho sem o qual não seria possível sua realização, sempre disposto a discutir novas abordagens e pontos que certamente contribuíram para meu aprendizado profissional e pessoal.

A Maria José, Agenor, Carla Caldas e Osman, que na ausência de meus pais, foram meus guias e tutores, saiba que os tenho com grande apreço em meu coração.

Aos meus grandes amigos, João Vitor, George Narita, Raniery Júnior, Ariane Priscilla, Felipe Anderson, Kim Morise, Arnóbio Júnior, Patrícia Fukushima, Diogo Turra, Raquel Cabral, Raissa Campagnaro, Rafael Delalibera, Filipe Verri, Angelica Ribeiro, Welighton Zhao, Artur Acelino pelo apoio e pelas horas disponíveis para ouvir lamentos e boas novas, saibam que sempre levarei as lembranças de todos os momentos que passamos juntos.

Aos meus amigos irmãos de apartamento, Luiz Vinícius e Lucas Tadeu com que pude dividir horas de alegria e diversão, obrigado por me receberem tão bem.

A todos os meus amigos do CGR, pelos conselhos e recomendações que proporcionaram experiências memoráveis durante todas as aventuras vividas nesse período.

Um agradecimento especial à Fundação de Amparo à Pesquisa de São Paulo (FAPESP), pelo suporte financeiro investido no projeto N. 2016/16769-0 que possibilitou o desenvolvimento deste trabalho e sem o qual não teria sido possível alcançar os resultados obtidos.

*“Words are, in my not-so-humble opinion,
our most inexhaustible source of magic.
Capable of both inflicting injury, and remedying it.”
(Albus Dumbledore, Harry Potter and the Deathly Hallows)*

Resumo

Utiliza-se recombinação de soluções em diversas estratégias de otimização, principalmente aquelas relacionadas a meta-heurísticas populacionais. Operadores de recombinação por decomposição particionam as variáveis de decisão do problema de modo a permitir a decomposição da função de avaliação. Assim, encontra-se, com custo computacional proporcional ao custo de se avaliar uma solução do problema, a melhor solução entre um número de soluções descendentes que cresce exponencialmente com o número de partições encontradas. Recombinação por decomposição foi até aqui utilizada apenas em problemas em que as informações sobre o relacionamento entre as variáveis de decisão são conhecidas a priori. O objetivo principal desta pesquisa de mestrado foi o desenvolvimento de um novo operador de recombinação por decomposição para todos os problemas de otimização pseudo-Booleana. Para isso, foi necessário estimar as ligações entre as variáveis de decisão por meio de procedimentos utilizados em algoritmos de estimação de distribuição e avaliar as partições encontradas pelo novo operador de recombinação. Os resultados encontrados demonstram que o novo operador desenvolvido obteve resultados relevantes para os problemas abordados em relação a geração de novas soluções candidatas por recombinação, em comparação aos demais operadores de recombinação utilizados.

Palavras-chave: otimização combinatória. algoritmos genéticos. *genetic linkage*.

Abstract

The recombination of solutions is important for most of the population meta-heuristics. Recombination by decomposition partitions the decision variables of the problem in order to allow the decomposition of the evaluation function. In this way, it allows to find, with computational cost proportional to the cost of evaluating one solution of the problem, the best solution among a number of offspring solutions that grows exponentially with the number of partitions found by the recombination operator. Recombination by decomposition has been so far used only in problems where the information about the linkage between the decision variables is known. The main objective of this project was the development of new operators of recombination by decomposition for all pseudo-Boolean optimization problems. For this purpose, was necessary to estimate the linkage between the decision variables by using procedures generally employed in estimation of distribution algorithms. Our results show that the new recombination operator obtained significant results for the problems chosen relate to the generation of new solutions by recombination, in comparison to the other recombination operators used.

Keywords: combinatorial optimization. genetic algorithm. genetic linkage.

Lista de figuras

Figura 1 – Exemplo de relação de adjacência para $K = 2$ variáveis do modelo Kauffman e $n = 3$ loci.	31
Figura 2 – Diagrama de estados de um algoritmo genético.	37
Figura 3 – Ilustração do funcionamento do <i>crossover</i> de um ponto (A) e do operador de mutação denominado <i>bit-flip</i> (B), onde x_i representa um gene na posição i do indivíduo.	38
Figura 4 – Aplicação do operador proposto para o modelo NK com $N = 6$ e $K = 2$. a) Grafo de interações. Neste exemplo, as soluções pais são $\mathbf{x} = [1, 1, 0, 0, 1, 1]^T$ e $\mathbf{y} = [0, 1, 0, 1, 1, 1]^T$. b) De modo a identificar as partições recombinantes, um grafo não-direcionado G é criado a partir do grafo de interações removendo-se todos os elos dos vértices i nos quais $x_i = y_i$. As partições recombinantes são os componentes conectados com mais de um vértice ou nos quais $x_i \neq y_i$. Aqui duas partições recombinantes são identificadas: a primeira com os vértices 4,5 e 6; a segunda com os vértices 1,2 e 3. c) A melhor solução descendente é gerada pela recombinação das melhores soluções parciais. Neste exemplo, o descendente herda os elementos da primeira partição de $\mathbf{y}$ e os elementos da segunda partição de $\mathbf{x}$	41
Figura 5 – Diagrama dos estados de um algoritmo de estimação de distribuição.	42
Figura 6 – Funcionamento do BOA.	45
Figura 7 – Grafo direcionado das variáveis de decisão do problema esperado para o modelo NK com $n = 10$, $K = 3$ e vizinhança por adjacência.	58
Figura 8 – Grafo direcionado das variáveis de decisão do problema, obtido através das configurações da BN com $n = 10$, $K = 3$ e vizinhança por adjacência.	59
Figura 9 – Grafo direcionado das variáveis de decisão do problema para o modelo NK com $n = 10$, $K = 3$ e vizinhança aleatória. O grafo gerado de forma aleatória no início da execução do AG permanece o mesmo até o fim em uma execução padrão com outros operadores.	59
Figura 10 – Grafo direcionado das variáveis de decisão do problema, obtido através das configurações da BN com $n = 10$, $K = 3$ e vizinhança aleatória.	60

Lista de tabelas

Tabela 1	– Exemplo de uma tabela representando 2^{K+1} combinações possíveis e sua distribuição de contribuição ao <i>fitness</i> para $N=3$	32
Tabela 2	– Termos usados comumente no contexto biológico e seus equivalentes no contexto matemático.	35
Tabela 3	– Soluções que constituem a população inicial, onde v_i representa o valor para a i -ésima posição da solução.	43
Tabela 4	– Distribuição de probabilidades para variáveis com valores recorrentes entre as soluções mais bem avaliadas. Note que, a probabilidade para uma opção alternativa de valor de variável é complemento da corrente e deve somar 1.	43
Tabela 5	– Tabela com valor médio de <i>fitness</i> para 50 execuções do GA com Modelo NK, vizinhança por adjacência e condição de parada 20 mil gerações. O símbolo s indica que o valor encontrado é estatisticamente significativo, seguido de $+$ se o BPX obteve melhor resultado ou $-$ caso contrário. .	63
Tabela 6	– Tabela com valor médio de <i>fitness</i> para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.	63
Tabela 7	– Tabela com porcentagem de soluções geradas melhores que os pais para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.	64
Tabela 8	– Tabela com porcentagem de soluções geradas melhores que os pais para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.	64
Tabela 9	– Tabela com porcentagem de soluções geradas melhores o melhor da execução para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.	65
Tabela 10	– Tabela com porcentagem de soluções geradas melhores o melhor da execução para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.	65
Tabela 11	– Tabela com médio de tempo em segundos por execução do operador para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.	66
Tabela 12	– Tabela com médio de tempo em segundos por execução do operador para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.. . . .	66

Tabela 13 – Tabela com valor médio de <i>fitness</i> para o problema da mochila 0-1 para configuração com busca local primeiro melhoramento e sem. O símbolo <i>s</i> indica que o valor encontrado é estatisticamente significativo, seguido de + se o BPX obteve melhor resultado ou – caso contrário.	67
Tabela 14 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema da mochila 0-1 com configuração junto com busca local primeiro melhoramento e sem.	68
Tabela 15 – Tabela com valor médio de soluções melhores que o melhor da população para o problema da mochila 0-1 com configuração junto com busca local primeiro melhoramento e sem.	68
Tabela 16 – Tabela com valor médio dos melhores <i>fitness</i> para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$. O símbolo <i>s</i> indica que o valor encontrado é estatisticamente significativo, seguido de + se o BPX obteve melhor resultado ou – caso contrário. .	70
Tabela 17 – Tabela com valor médio dos melhores <i>fitness</i> para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$.	70
Tabela 18 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$	70
Tabela 19 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$	70
Tabela 20 – Tabela com valor médio de soluções melhores que o melhor por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$	70
Tabela 21 – Tabela com valor médio de soluções melhores que o melhor por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$	70

Lista de abreviaturas e siglas

AE	Algoritmos Evolutivos
AG	Algoritmos Genéticos
BN	<i>Bayesian Networks</i>
BOA	<i>Bayesian Optimization Algorithm</i>
BPX	<i>Bayesian Partition Crossover</i>
EDA	<i>Estimation Distribution Algorithms</i>
GAPX	<i>Generalized Partition Crossover</i>
GPX	<i>Generalized Asymmetric Partition Crossover</i>
LS	<i>Local Search</i>
LSFI	<i>Local Search First-Improvement</i>
PB	Pseudo-Booleanos
PX	<i>Partition Crossover</i>
RTS	<i>Restricted Tournament Selection</i>
RWS	<i>Roulette Wheel Selection</i>
TSP	<i>Traveling Salesman Problem</i>
VIG	<i>Variable Interaction Graph</i>

Sumário

	Introdução	25
1	PROBLEMAS DE OTIMIZAÇÃO	28
1.1	Problemas Pseudo-Booleanos	29
1.1.1	Problema da Mochila 0-1	29
1.1.2	Funções Unárias e Funções Enganosas	30
1.2	Problemas Pseudo-Booleanos k-restritos	30
1.2.1	NK Landscape	31
1.3	Otimização Caixa-Preta, Cinza e Branca	32
2	COMPUTAÇÃO EVOLUTIVA	34
2.1	Esquemas e Blocos Construtivos	35
2.2	Algoritmos Genéticos	36
2.2.1	Operadores Genéticos	38
2.2.2	Partition Crossover (PX)	39
2.2.3	PX para Problemas de Otimização Pseudo- Booleana k -restritos	40
2.3	Algoritmos de Estimação de Distribuição	41
2.3.1	Algoritmo de Otimização Bayesiano	44
2.3.1.1	Redes Bayesianas	45
2.3.1.2	Construindo Redes Bayesianas	45
3	BAYESIAN PARTITION CROSSOVER	47
3.1	BPX	47
3.2	Experimentos	52
4	RESULTADOS	55
4.1	Critérios e Recursos Utilizados	55
4.2	Grafos de Interação Real e Obtido com Modelo NK	57
4.3	Resultados: Eficiência dos Operadores	61
4.3.1	Modelo NK	61
4.3.1.1	Uniforme e 2-pontos	61
4.3.1.2	PX, BPX e BPEX	62
4.3.2	Mochila 0-1	67
4.3.3	Funções Enganosas	69
5	CONCLUSÕES	71

5.1	Contribuições da Pesquisa	71
5.2	Vantagens e Desvantagens	72
5.3	Trabalhos Futuros	73
	Referências	74

Introdução

Contextualização

A recombinação de soluções assume papel de extrema importância em meta-heurísticas populacionais, particularmente nos algoritmos genéticos (EIBEN; SMITH et al., 2003). Vale ressaltar que a recombinação de soluções não é exclusividade de meta-heurísticas populacionais, sendo utilizada em diversos paradigmas de otimização. Por exemplo, *multi-trial* LKH (Lin-Kernigan-Helsgaun) é reconhecidamente a heurística de maior sucesso para o problema do caixeiro viajante (TSP – *traveling salesman problem*) (COOK, 2012). *Multi-trial* LKH obteve sucesso em encontrar o ótimo global em todos os problemas pesquisados por K. Helsgaun e detém recordes para diversos problemas nos quais as soluções ótimas não são conhecidas (HELSGAUN, 2009). *Multi-trial* LKH faz uso de um operador de recombinação conhecido como *iterative partial transcription* (MÖBIUS et al., 1999), que é fundamental para o sucesso da estratégia ao recombinar soluções obtidas em diferentes reinicializações e execuções do algoritmo. O processo de recombinar soluções localmente ótimas encontradas em diversas execuções de um algoritmo, ou por diferentes algoritmos, faz com que a pesquisa por operadores eficientes de recombinação seja de grande importância para a área de otimização de um modo geral.

Em Algoritmos Evolutivos (AEs), operadores de recombinação, também chamados de operadores de *crossover*, são aleatórios em sua maioria. Whitley et al. (WHITLEY; HAINS; HOWE, 2009; HAINS; WHITLEY; HOWE, 2011) propuseram um operador de recombinação determinístico chamado *generalized partition crossover* (GPX) para o TSP simétrico. Posteriormente GPX foi estendido para o TSP assimétrico no *generalized asymmetric partition crossover* (GAPX) (TINÓS; WHITLEY; OCHOA, 2014), e mais recentemente no GPX2 (TINÓS; OCHOA; WHITLEY, 2018). Estes acrescentam uma série de melhorias ao GPX, permitindo explorar um número maior de soluções descendentes potenciais. Quando aplicados para recombinar as soluções produzidas por *multi-trial* LKH, GPX2 encontra com frequência soluções com maior *fitness* (TINÓS; OCHOA; WHITLEY, 2018).

Recentemente, mostrou-se que é possível aplicar operadores similares para outros

problemas além do TSP. Em (TINÓS; WHITLEY; CHICANO, 2015), os autores apresentam o *Partition Crossover* (PX) para classe de problemas pseudo-Booleanos k -restritos, que contém os problemas MAX- k SAT (WHITLEY; CHEN, 2012), NK *landscapes* e algumas versões do *spin glass* (WHITLEY; CHICANO; GOLDMAN, 2016). Em (TINÓS et al., 2018), o PX é estendido para a versão do problema de agrupamento (*clustering*) com codificação discreta.

Basicamente, tais operadores particionam as soluções em p sub-conjuntos de variáveis de decisão através da identificação de componentes conectados no grafo resultante da remoção de características comuns no grafo de interações. O grafo de interações contém as informações das ligações entre as variáveis de decisão, informação esta extraída da análise da função de custo.

O particionamento do grafo de interações permite encontrar a melhor solução entre 2^p possíveis soluções descendentes a um custo computacional similar ao de se avaliar duas soluções do problema ¹, sendo p o número de partições encontradas. Isto é possível porque os operadores exploram a decomposição linear da função de custo (ou função de *fitness*, na terminologia utilizada em computação evolutiva). Esta é formada por subfunções componentes que podem ser computadas usando-se somente as soluções parciais de uma ou outra solução pai dentro das partições encontradas pelo operador. Desta forma, escolhendo-se as melhores soluções parciais, encontra-se a melhor dentre 2^p possíveis soluções descendentes a um custo de avaliar apenas 2 soluções ². Operadores de recombinação com tais características são chamados aqui de operadores de recombinação por decomposição ³.

É possível demonstrar que, quando dois ótimos locais são recombinados por PX, frequentemente as soluções descendentes são também ótimas (TINÓS; WHITLEY; CHICANO, 2015). Também, é possível constatar a eficiência de PX quando redes de ótimos locais (OCHOA et al., 2015) são analisadas. Em uma rede de ótimos locais, a superfície de *fitness* de um problema de otimização combinatória é comprimida em um grafo no qual os vértices são os ótimos locais. Já as arestas indicam possíveis transições entre os ótimos locais. Quando o PX é utilizado, as redes de ótimos locais geralmente apresentam grande conectividade, permitindo alcançar novos ótimos locais por meio de um número menor de operações de transformação das soluções.

¹ Por exemplo, no TSP o custo de avaliar uma solução é $O(n)$, sendo n o número de vértices no grafo (por exemplo, cidades no TSP clássico).

² GAPX foi capaz de identificar $p = 42$ partições em uma única aplicação de recombinação. Tal número de partições resultou na identificação da melhor solução descendente dentre 2^{42} possíveis soluções a um custo de avaliar apenas 2 soluções (TINÓS; WHITLEY; OCHOA, 2014), que no caso do TSP é $O(n)$.

³ Por simplicidade, iremos utilizar a partir daqui o termo PX para todos os operadores de recombinação por decomposição. Os operadores de recombinação por decomposição conhecidos são: PX, GPX, GPX2 e GAPX

Objetivos

O objetivo principal deste trabalho é o desenvolvimento de um novo operador de recombinação por decomposição que possa ser aplicado em qualquer problema pseudo-Booleano. Todos os operadores de recombinação por decomposição desenvolvidos na literatura requerem como condição essencial que se conheça a relação entre as variáveis de decisão do problema *a priori*. Em algoritmos genéticos, se utiliza o termo *genetic linkage* para designar a relação entre as variáveis de decisão. Nos problemas em que recombinação por decomposição foi até aqui aplicada, as informações sobre as relações entre as variáveis de decisão são capturadas pelo grafo de interações, que é obtido pela análise da função de custo.

Para que seja possível a aplicação desse novo operador, é necessário estimar as informações sobre a ligação entre as variáveis de decisão por meio de procedimentos utilizados em Algoritmos Evolutivos (AEs) baseados em modelo (também conhecidos como algoritmos de estimação de distribuição) e avaliar as partições encontradas pelo operador de recombinação (LARRAÑAGA; LOZANO, 2001).

Organização

Este trabalho está organizado da seguinte maneira: O Capítulo 1 apresenta os problemas de otimização do tipo pseudo-Booleano (PB) abordados na etapa de experimentação deste trabalho. Já no Capítulo 2 é realizada uma revisão bibliográfica na área da computação evolutiva, descrita através de seus algoritmos e principais conceitos, além de seus operadores genéticos que elencam a importância do estudo e desenvolvimento de novos algoritmos. No Capítulo 3 é descrito passo-a-passo o novo operador de recombinação proposto nesse trabalho, seguido no Capítulo 4 pela descrição dos experimentos e análise dos resultados do operador nos problemas abordados. Por fim, no Capítulo 5 são apresentadas as contribuições da pesquisa na investigação por métodos de recombinação eficientes para a área de otimização de maneira geral, assim como limitações do trabalho e passos futuros do mesmo.

Problemas de Otimização

Problemas de otimização surgiram através da preocupação em encontrar as melhores variáveis de decisão de acordo com um ou vários objetivos, i.e. a melhor solução para uma instância de um determinado problema (PAPADIMITRIOU; STEIGLITZ, 1982). Esses problemas podem ser separados em duas categorias principais de acordo com as variáveis que operam: contínuos e discretos.

Problemas de otimização discretos ou combinatórios, são aqueles restritos a variáveis discretas, ou seja as variáveis de decisão só assumem um conjunto de valores discretos, tais como um conjunto de cores, sim ou não, números inteiros, entre outros. Em contraste a esse tipo de problema, os de otimização contínuos, são aqueles os quais as variáveis de decisão podem assumir um valor real de acordo com suas restrições.

Visto isso, um problema de otimização é dito como um conjunto I de instâncias (PAPADIMITRIOU; STEIGLITZ, 1982). Uma instância de um problema de otimização por sua vez, como descrito por Papadimitriou e Steiglitz (1982), é um par $(\mathbf{X}, f)$ onde, $\mathbf{X}$ é qualquer conjunto de soluções possíveis e f é a função objetivo que atua mapeando as soluções para números reais, i.e.:

$$f : \mathbf{X} \rightarrow \mathbb{R} \quad (1.1)$$

Dessa forma, o problema de otimização pode ser matematicamente representado como o problema de se encontrar a melhor solução $\mathbf{x}_1$ conhecida como ótimo global que satisfaça:

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_n), \forall n = 1 \dots |\mathbf{X}| \quad (1.2)$$

1.1 Problemas Pseudo-Booleanos

Funções objetivo que trabalham com um conjunto finito de variáveis de decisão e que realizam a conversão para um número real estão presentes na literatura há bastante tempo (BOROS; HAMMER, 2002). A classe de problemas pseudo-Booleanos são representados por uma função, que mapeia um vetor de n variáveis do alfabeto binário para um valor real e pode ser descrita da seguinte forma $f : \mathbb{B}^n \mapsto \mathbb{R}$. O termo pseudo-Booleano (PB) é atribuído apesar dessas funções não serem estritamente Booleanas. Um exemplo de problema dessa classe que será abordado nesse trabalho é o problema da Mochila 0-1.

1.1.1 Problema da Mochila 0-1

O problema da mochila é um popular problema de otimização combinatória da literatura. Apesar de se tratar de um problema NP-completo, alguns algoritmos da literatura como o Algoritmo Genético (AG) (HOLLAND, 1975), que será descrito na Seção 2.2, combinados com outros algoritmos i.g. busca gulosa, tem exibido resultados interessantes (ZHAO et al., 2009).

A variedade de aplicações deste problema no mundo real abrange desde seleção de projetos, planejamento econômico até carregamento de estoque, tornando-se um problema intensivamente estudado (SALKIN; KLUYVER, 1975; PISINGER, 2007). O problema da mochila 0-1 envolve a maximização do benefício (P) sem porém, exceder a capacidade da mochila (V). Matematicamente, o objetivo é maximizar,

$$\sum_{i=1}^n P_i x_i \quad (1.3)$$

dados que P_i corresponde ao benefício do item i , W_i ao peso também do item i , e $\mathbf{x}$ obedece a uma codificação binária de soluções candidatas, representadas por meio de um vetor binário ($i = 1, 2, 3, \dots, n$) onde,

$$x_i = \begin{cases} 1 & , \text{ se o objeto for selecionado} \\ 0 & , \text{ caso contrário} \end{cases} \quad (1.4)$$

sujeito à restrição,

$$\sum_{i=1}^n W_i x_i \leq V \quad (1.5)$$

1.1.2 Funções Unárias e Funções Enganosas

Funções unárias (*unation functions*) são funções cujas quais dependem apenas do número de 1s inclusos dentro da *string* (vetor de variáveis de decisão), ou no caso em questão, da solução candidata, independentemente da posição de ocorrência na mesma (DEB; GOLDBERG, 1994). O estudo desse tipo de função é interessante uma vez que seu entendimento e manipulação são fáceis, dado que possuem apenas $n + 1$ valores de função diferentes e um espaço de busca de 2^n , onde n é o tamanho da *string*.

Através do uso de funções armadilhas (*trap functions*), é possível criar funções enganosas (*deceptive*), onde generalizando, uma solução candidata com todos os alelos iguais a 1 é o ótimo global e outra solução candidata com todos os alelos iguais a 0, é um ótimo local que possui uma grande bacia de atração (TINÓS; YANG, 2007). A equação 1.6 descreve as condições para se criar uma função armadilha:

$$f(\mathbf{x}) = \begin{cases} \frac{a}{b}(z - u(\mathbf{x})), & \text{se } u(\mathbf{x}) \leq z; \\ \frac{b}{n-z}(u(\mathbf{x}) - z), & \text{senão} \end{cases} \quad (1.6)$$

sendo, $u(\mathbf{x})$ é a função unária do vetor binário $\mathbf{x}$, n o tamanho da solução candidata, b o ótimo global e a e z representam os valores dos parâmetros da função, o ótimo possivelmente enganoso e a mudança de inclinação que separa os ótimos respectivamente.

1.2 Problemas Pseudo-Booleanos k -restritos

Os problemas da classe dos Pseudo-Booleanos denominados k -restritos desempenham um papel importante na computação evolutiva, otimização combinatória, biofísica e aprendizado de máquina (SUTTON; WHITLEY; HOWE, 2012). O diferencial dessa classe é que as subfunções que compõem a função de avaliação são definidas pela interação epistática de k variáveis da solução binária, sendo k uma constante maior ou igual a 0 (OCHOA et al., 2015; SUTTON; WHITLEY; HOWE, 2012).

O problema de otimização pseudo-Booleano pode ser descrito como uma busca pela atribuição que melhor satisfaça a função objetivo, de maneira a maximizar ou minimizar, a depender do problema a função de avaliação, dado um conjunto finito de variáveis binárias como entrada (EÉN; SORENSSON, 2006). NK-Landscapes, MAX-kSAT, MAX-CUT e modelos *Spin Glass* são representantes dessa classe de problemas PB k -restritos (SUTTON; WHITLEY; HOWE, 2012; WHITLEY; CHEN, 2012).

1.2.1 NK Landscape

O modelo NK, também conhecido como NK *landscape*, foi proposto inicialmente por Stuart Kauffman (KAUFFMAN, 1993) e é uma das classes mais populares de problemas para realizar testes de técnicas de otimização em instâncias aleatórias de problemas (PELIKAN et al., 2009). O sistema robusto ajustável de Kauffman foi projetado para simular a adaptação de organismos e o processo de evolução em um ambiente competitivo (WELCH; WAXMAN, 2005), onde N representa uma constante fixa de genes do cromossomo e K a quantidade de genes os quais contribuem de forma relacionada em cada subfunção da função de *fitness*.

Em um cromossomo de N genes, cada *locus* interage com até $N - 1$ loci da solução, valor máximo possível de atribuir a K . Quando K é maior que 0, pode-se dizer que existe epistasia (WELCH; WAXMAN, 2005). Dessa forma, cada subfunção f_i depende do alelo do i -ésimo gene e de K outros alelos atribuídos aos genes a qual se estabelece uma relação (GEARD et al., 2002). Um terceiro parâmetro para o modelo é a distribuição de interação entre os genes, se acontece entre o locus corrente e seus K -vizinhos adjacentes ou de forma aleatória entre os loci.

A Figura 1, assim como o exemplo presente em (KAUFFMAN, 1993), descreve a interação entre as N variáveis ou genes do modelo para $K = 2$ com distribuição adjacente. O *fitness* de cada solução candidata é calculado através da função $f(\mathbf{x})$, pela média das contribuições ao *fitness* (f_i) de cada N locus da solução e seus K loci adjacentes, explícita na Equação 1.7. Onde $\mathbf{x} \in \mathbb{B}^N$ é a solução candidata e $\mathbf{m}_i \in \mathbb{B}^N$ é uma máscara que indica os elementos de $\mathbf{x}$ que influenciam a subfunção f_i .

Figura 1 – Exemplo de relação de adjacência para $K = 2$ variáveis do modelo Kauffman e $n = 3$ loci.

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x}, \mathbf{m}_i) \quad (1.7)$$

Deste modo, o número de combinações possíveis para cada $k = K + 1$ genes com alelos $\in \{0, 1\}$ é igual a 2^{K+1} , sendo que f_i depende de x_i e de outras K variáveis de decisão. Os valores para cada combinação das k variáveis de decisão que influenciam a subfunção f_i são usualmente geradas aleatoriamente seguindo uma distribuição uniforme no intervalo entre 0.0 e 1.0 (KAUFFMAN, 1993). Um exemplo de atribuição dos valores para as subfunções é dado na Tabela 1.

Tabela 1 – Exemplo de uma tabela representando 2^{K+1} combinações possíveis e sua distribuição de contribuição ao *fitness* para $N=3$.

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	f_1	f_2	f_3	$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x})$
0	0	0	.1	.2	.3	.2
1	0	0	.2	.1	.6	.3
1	1	0	.3	.4	.8	.5
1	1	1	.2	.9	.1	.4
1	0	1	.3	.6	.7	.6
0	1	0	.3	.1	.9	.43
0	1	1	.3	.8	.6	.56
0	0	1	.5	.8	.9	.7

1.3 Otimização Caixa-Preta, Cinza e Branca

Através do conhecimento da estrutura do problema abordado é possível direcionar os esforços dos algoritmos, como forma de otimizar a busca pela melhor solução possível de ser encontrada. O conceito de estrutura do problema pode ser entendido como a forma em que as variáveis de decisão do problema interagem entre si, através de padrões, influenciando dessa forma na função objetivo.

Os problemas de otimização caixa-preta, assim como seus otimizadores, são aqueles os quais o conhecimento sobre a estrutura do problema e a semântica das soluções candidatas em potencial são desconhecidos ou não são utilizados explicitamente para melhorar a busca pelos ótimos. Assim sendo, é necessário o uso de técnicas de aprendizado da estrutura para amostragem, avaliação e processamento de novas soluções candidatas (PELIKAN et al., 2003).

Já os problemas de otimização caixa-cinza, descritos em Santana (SANTANA, 2017), são vistos como uma classe de problemas onde se tem algum conhecimento *a priori*

sobre a estrutura do problema. Assim, através do uso dessa informação, não estimada por meio de algoritmos, é possível tornar a otimização do problema uma tarefa mais eficiente. Os algoritmos caixa-branca por sua vez, são aqueles os quais toda a estrutura do problema e as subfunções de avaliação são conhecidas.

Alguns problemas do mundo real e outros abordados anteriormente como o NK Landscape fazem parte da classe de problemas de otimização caixa-cinza. Esses problemas por sua vez podem ser otimizados utilizando Algoritmos Genéticos com operadores de transformação eficientes como o PX, que será descrito na Seção 2.2.2. Já nos problemas caixa-preta como o da Mochila 0-1 e o problema de otimização derivado de funções enganosas aqui descrito, não se conhece a priori as relações entre as variáveis de decisão (estrutura do problema).

Computação Evolutiva

Na vasta área da ciência da computação existe uma ramificação da inteligência artificial denominada computação evolutiva. Com base na teoria da evolução das espécies de Charles Darwin ¹, surgiram nos anos 1960 as primeiras publicações relacionadas aos AEs. AEs são metaheurísticas populacionais inspiradas em princípios evolutivos de seleção, recombinação e mutação de indivíduos, como exemplo os Algoritmos Genéticos (AGs) (HOLLAND, 1975). Mais tarde a comunidade científica uniu os conceitos de herança estudados por Gregor Mendel ao Darwinismo, criando assim a síntese evolutiva moderna ou neo-Darwinismo (COLIN; JONATHAN, 2002).

A primeira etapa de um AE é a definição de uma população inicial de indivíduos, geralmente denominados soluções candidatas ou cromossomos. A cada geração mecanismos de seleção e transformação dos indivíduos que são utilizados para gerar uma nova população. Esses indivíduos são avaliados de acordo com o resultado real de uma função de aptidão ou *fitness*, que representa o quão eficiente o indivíduo é em relação ao problema (EIBEN; SMITH, 2008).

A seleção dos indivíduos pode acontecer de diferentes formas variando de acordo com a abordagem utilizada e o método evolutivo, porém selecionando sempre os indivíduos em função do *fitness* destes (MITCHELL, 1995). A nova população é então gerada através da transformação (reprodução) das características presentes nos pais. Os operadores mais comuns de reprodução são a mutação e a recombinação. Esta última, tema desta pesquisa, pode ser realizada de diferentes formas, como a descrita na Seção 2.2.2.

Com a recombinação, os indivíduos descendentes compartilham características presentes nos pais. Para garantir a diversidade na população, existe a probabilidade de ocorrer uma perturbação nos genes do indivíduo, denominado mutação, fazendo com que o mesmo desenvolva uma característica singular que não necessariamente estava presente nos pais.

¹ *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, 1859.*

Assim, tem-se uma nova geração de indivíduos, disponibilizando novas potenciais soluções para o problema. Caso algum critério pré-estabelecido de parada tenha sido alcançado, como um número finito de gerações, o algoritmo finaliza, do contrário, realiza outra vez as etapas de seleção, recombinação, mutação e assim por diante (FOGEL, 2006).

Para elucidar a diversidade dos termos empregados neste trabalho e na literatura, é possível observar a Tabela 2. Devido a diversidade dos problemas e a capacidade de resolução dos algoritmos, múltiplas técnicas evolutivas foram desenvolvidas, tais como os Algoritmos Genéticos (HOLLAND, 1975), Estratégias Evolutivas (BEYER; SCHWEFEL, 2002), Programação Evolutiva (BACK, 1996) e Algoritmos de Estimação de Distribuição (MÜHLENBEIN; PAASS, 1996).

Tabela 2 – Termos usados comumente no contexto biológico e seus equivalentes no contexto matemático.

Contexto Genético	Contexto Computacional
Alelo	Conjunto de valores possíveis das variáveis de decisão
Locus	Variável de decisão
Indivíduo, Cromossomo, Genótipo	Solução, Vetor de variáveis de decisão (<i>string</i>)
População	Conjunto de soluções candidatas

2.1 Esquemas e Blocos Construtivos

Um esquema é descrito como um template que representa um subconjunto de soluções candidatas que possuem valores semelhantes em determinadas posições (SIVANANDAM; DEEPA, 2007). Tal template é representado por *strings* contendo os símbolos 1, 0 e *, um meta-caractere usado para descrever um valor sem importância, podendo assumir qualquer valor do alfabeto utilizado. Por exemplo, o subconjunto de soluções candidatas {00000, 00001, 00011, 00010, 00101, 00100, 00110, 00111} pode ser representados pelo esquema $\mathbf{h} = 00^{***}$; as soluções do subconjunto ditas instâncias do mesmo esquema.

A ordem de um esquema é dada de acordo com a quantidade de bits diferentes de *, ou seja, para o esquema $\mathbf{h} = 1^{**}1^{*}$, a ordem correspondente é 2. Ainda, para cada esquema, seu comprimento (C_h) é definido de acordo com a distância entre as posições fixas extremas, assim para o esquema anteriormente citado seria $C_h = 4 - 1$. Além disso, é possível realizar 2^n combinações de bits e descrever 3^n esquemas, onde n representa o número de bits da cadeia (MELANIE, 1999).

A Teoria dos Esquemas apresentada por John Holland (HOLLAND, 1992), busca explicar o crescimento dos esquemas após as recombinações realizadas a cada nova geração

pelos algoritmos genéticos (Seção 2.2). Em suma, essa teoria implica que esquemas de baixa ordem que possuem *fitness* acima da média crescem exponencialmente na população, aumentando assim o número de instância dos mesmos (MELANIE, 1999).

Esquemas de baixa ordem bem definidos que contribuem de maneira positiva para o *fitness* da solução são chamados blocos construtivos (*building blocks*) (GOLDBERG; HOLLAND, 1988; SIVANANDAM; DEEPA, 2007). Esses blocos, por contribuírem para o *fitness* da solução se propagam nas gerações subsequentes, ocasionando a justaposição de pequenos blocos até compor uma solução ótima, se reconhecida, formando um único bloco construtivo do tamanho da cadeia. Essa noção apresentada por John Holland (HOLLAND, 1992) recebe o nome de Teoria dos Blocos Construtivos, sendo apresentada como justificativa para a eficiência dos algoritmos genéticos.

2.2 Algoritmos Genéticos

O Algoritmo Genético é uma técnica evolutiva desenvolvida por John Holland e seu grupo de pesquisa (HOLLAND, 1975), que combina a sobrevivência dos indivíduos mais aptos com o cruzamento dos mesmos para gerar novos indivíduos, incorporando assim a teoria da evolução de Darwin e a genética Mendeliana. Nesse contexto os indivíduos representam possíveis soluções para problemas, descritos em um espaço de busca finito, que é explorado através de seus pontos pela combinação dos indivíduos.

O primeiro passo na execução de um AG, assim como apresentado pela Figura 2, é a inicialização da população de soluções candidatas, gerada de forma aleatória ou de maneira mais específica com conhecimento *a priori* de potenciais soluções. Após avaliar toda a população inicial através da função de *fitness*, os indivíduos que irão participar da nova população na próxima geração são selecionados, conforme o valor do *fitness* dos mesmos. A seleção obedece uma quantidade fixa de indivíduos e pode variar de acordo com a implementação e finalidade do algoritmo.

Os componentes básicos para se trabalhar com AGs são (CARR, 2014):

- Uma forma de avaliar o quão bom os indivíduos são como possíveis soluções para o problema, descrito como função de *fitness*.
- Uma população inicial de indivíduos.
- Operador de seleção dos indivíduos, geralmente de acordo com o valor de *fitness* com preferência aos maiores.
- Operador de recombinação dos indivíduos (*crossover*), descrito na Seção 2.2.1.
- Operador de mutação.

Em seguida, o subconjunto de indivíduos selecionados são recombinaados para gerar a nova população. Essa recombinação pode variar de acordo com a implementação e finalidade do algoritmo; comumente o operador de recombinação mais simples usado é o *crossover* de um ponto, que divide as cadeias de bits em um determinado ponto e recombina as diferentes partes para criar os indivíduos descendentes, como ilustrado na Figura 3 (A). Outros dois tipos de operadores de recombinação comumente utilizados na literatura e também utilizados neste trabalho são: o operador de 2-pontos, que similar ao de 1-ponto divide a cadeia de bits em dois pontos todavia e o operador uniforme, que realiza troca dos bits dos progenitores de acordo com uma taxa fixa pré-definida.

Figura 2 – Diagrama de estados de um algoritmo genético.

Posteriormente, o operador de mutação mantém a diversidade da população, geralmente através da modificação de um valor de gene aleatório no subconjunto de indivíduos recombinaados, como mostra a Figura 3 (B) e é responsável por gerar alelos não presentes nos pais.

Figura 3 – Ilustração do funcionamento do *crossover* de um ponto (A) e do operador de mutação denominado *bit-flip* (B), onde x_i representa um gene na posição i do indivíduo.

Por fim, a nova prole de indivíduos é reinserida na população original. Caso alguma condição de parada pré-estabelecida tenha sido alcançada ou uma solução ideal tenha sido descoberta o algoritmo finaliza sua execução. Do contrário executa a etapa de seleção e suas subsequentes em sequência até tal condição ser alcançada. Cada iteração recebe o nome de geração, ou seja, ao final de 5 iterações o algoritmo atingiu a geração 5 e todas as gerações juntas correspondem a uma execução ou corrida (*runs*).

2.2.1 Operadores Genéticos

Os operadores genéticos podem apresentar variações de acordo com a técnica evolutiva escolhida, porém para os AGs existem três tipos de operadores principais: seleção, recombinação e mutação. O operador de seleção, como introduzido na seção anterior, é responsável por selecionar os indivíduos que poderão participar do processo de recombinação e posteriormente serem inseridos na população.

Entre os métodos seleção usualmente utilizados estão o método da roleta (*Roulette Wheel Selection* - RWS), além do Torneio de Seleção Restrita (*Restricted Tournament Selection* - RTS) que será abordado na Seção 2.3.1. O método RWS, funciona atribuindo uma probabilidade de seleção para cada indivíduo da população corrente proporcional ao seu *fitness*, criando uma abstração de roleta cuja probabilidade do indivíduo ser escolhido corresponde a uma fatia da mesma, indicada a cada novo giro da roleta.

Dessa forma, os indivíduos com maior *fitness* terão uma parcela maior na roleta e consequentemente mais chances de se propagar nas seguintes gerações. O método RTS por sua vez funciona selecionando aleatoriamente k indivíduos da população que irão competir em um "torneio" e o melhor, de acordo com o *fitness*, será escolhido (HARIK, 1995). Além disso, se usa elitismo no qual uma quantidade fixa de indivíduos com *fitness* mais elevado da população é escolhido para integrar a próxima população.

Os operadores de recombinação são análogos à recombinação genética e assim como os operadores de seleção, possuem ampla diversidade de métodos, variando de acordo com a estrutura de dados utilizada. Neste trabalho serão considerados apenas dados no formato de cadeia de caracteres binários. Um exemplo é o *crossover* de partição PX que será descrito na Seção 2.2.2 e o operador de um ponto descrito anteriormente e ilustrado na Figura 3(A).

Os operadores de mutação são análogos à mutação genética, trabalhando como etapa importante para garantir a diversidade genética dos indivíduos da população através de mudanças em sua sequência de genes. Comumente o método *bit-flip* é utilizado. Esse método como apresentado pela Figura 3(B), atua em cada gene realizando uma mudança de valor de acordo com uma taxa de mutação fixa pré-estabelecida.

Vale ressaltar que o *design* do AE pode diferir de acordo com a aplicação no domínio do problema. A decisão pela escolha dos operadores bem como seus respectivos parâmetros são sensíveis ao domínio e refletem a habilidade desejada do mesmo (EIBEN; SMITH, 2011).

2.2.2 Partition Crossover (PX)

O operador de recombinação PX apresentado por Whitley et al. (WHITLEY; HAINS; HOWE, 2009) para o problema do Caixeiro Viajante e posteriormente expandido para funções pseudo-Booleanas k -restritas em (TINÓS; WHITLEY; CHICANO, 2015), funciona através do uso das sequências de genes que não são comuns nos indivíduos progenitores para particionar o grafo de interações entre as variáveis de decisão.

Nos problemas pseudo-Booleanos a epistasia entre as variáveis de decisão pode ser representada através de um grafo direcionado ² denominado VIG (*variable interaction graph*), onde os vértices correspondem as variáveis de decisão do problema e suas aretas a interação entre as variáveis. Esta informação é obtida pela análise da função de custo. Portanto, PX pode ser aplicado apenas em problemas em que o grafo de interações é conhecido a priori.

A principal característica desse operador é a possibilidade de decidir qual trecho

² Um grafo não-direcionado pode ser utilizado também, dependendo da implementação do PX.

dos indivíduos progenitores a prole irá herdar através da decomposição da função de *fitness*, selecionando de maneira a maximizar ou minimizar, à depender do problema, a mesma.

Como resultado, esse operador, ao recombinar soluções que são localmente ótimas, gera com grande frequência soluções que também são ótimos locais, existindo a possibilidade de quando as mesmas estão perto o bastante do ótimo global, ao recombinar seja possível atingi-lo.

2.2.3 PX para Problemas de Otimização Pseudo-Booleana k -restritos

Se a função pseudo-Booleana tem a forma $f : \mathbb{B}^n \rightarrow \mathbb{R}$ e a função de avaliação puder ser escrita como uma soma de subfunções que dependem de no máximo k variáveis, então dizemos que a função é também k -restrita. Considere por exemplo o modelo NK, descrito anteriormente.

O modelo NK é considerado difícil para algoritmos genéticos com operadores clássicos de recombinação e mutação (HECKENDORN; RANA; WHITLEY, 1999). Em relação à recombinação, a maior dificuldade é que os operadores geralmente não exploram a interação entre as variáveis do modelo para definir quais elementos devem ser herdados de um ou outro pai. Desta forma, os descendentes têm avaliação geralmente pior do que a dos pais porque os blocos de ligação entre as variáveis já encontrados são rompidos.

O operador PX explora o grafo de interações (ou epistasia) do modelo NK para particionar os elementos das soluções pai. O *fitness* do descendente é computado como uma soma de avaliações parciais de partições de variáveis herdadas dos pais.

No PX, o grafo de interações (Figura 4.a) é modificado removendo-se os elos que partem de vértices que representam elementos cujos bits de ambos os pais são comuns. Aplicando-se um algoritmo para encontrar os componentes conectados no grafo resultante (grafo de recombinação), partições recombinantes são encontradas. Cada partição é definida por elementos da solução influenciados somente por outros elementos dentro da partição ou por elementos fora da partição com bit semelhante para ambos os pais (Figura 4.b). Assim, a avaliação parcial para cada partição é independente da escolha de um ou outro pai para os elementos fora da partição. Então, os bits da solução descendente são escolhidos de acordo com a melhor escolha dos bits dos pais em cada partição (Figura 4.c).

Figura 4 – Aplicação do operador proposto para o modelo NK com $N = 6$ e $K = 2$. a) Grafo de interações. Neste exemplo, as soluções pais são $\mathbf{x} = [1, 1, 0, 0, 1, 1]^T$ e $\mathbf{y} = [0, 1, 0, 1, 1, 1]^T$. b) De modo a identificar as partições recombinantes, um grafo não-direcionado G é criado a partir do grafo de interações removendo-se todos os elos dos vértices i nos quais $x_i = y_i$. As partições recombinantes são os componentes conectados com mais de um vértice ou nos quais $x_i \neq y_i$. Aqui duas partições recombinantes são identificadas: a primeira com os vértices 4,5 e 6; a segunda com os vértices 1,2 e 3. c) A melhor solução descendente é gerada pela recombinação das melhores soluções parciais. Neste exemplo, o descendente herda os elementos da primeira partição de $\mathbf{y}$ e os elementos da segunda partição de $\mathbf{x}$.

2.3 Algoritmos de Estimação de Distribuição

Algoritmos de Estimação de Distribuição (*estimation distribution algorithms* – EDAs) também referenciados como Algoritmos Genéticos de Construção de Modelos Probabilísticos (PMBGAs), da mesma forma como os AGs descritos na Seção 2.2, funcionam como ferramentas evolutivas que selecionam as soluções mais promissoras em espaços de busca. Exposto a princípio por Mühlenbein e Paass (MÜHLENBEIN; PAASS, 1996), essa técnica difere das usuais uma vez que os operadores evolutivos de recombinação e mutação são substituídos por um modelo probabilístico de acordo com a distribuição de probabilidade dos indivíduos com maior *fitness*.

Em contrapartida aos operadores genéticos usuais, nos EDAs, a geração de novos indivíduos se dá através da amostragem do modelo estimado, como é possível observar na Figura 5. Uma das maneiras propícias observadas de se gerar novas soluções sem fazer uso dos operadores tradicionais, seria verificar as variáveis que cada solução do conjunto

inicial tem em comum e a partir disso, utilizar essa informação para gerar novas soluções para o problema (CROCOMO, 2012).

Considere por exemplo, um conjunto inicial de soluções de um problema P descrito pela Tabela 3. Note que nas duas primeiras soluções é possível observar que as mesmas compartilham o mesmo valor na variável v_1 . Dessa forma é possível inferir que 66% das três soluções mais bem avaliadas possuem $v_1 = 1$ e que 33% das mesmas possuem $v_2 = 1$. Logo, é factível estimar a distribuição de valores para cada variável da população de indivíduos, assim como apresentado na Tabela 4.

Figura 5 – Diagrama dos estados de um algoritmo de estimação de distribuição.

O modelo univariado em questão, por ser um modelo de primeira ordem, desconsidera a ligação entre as variáveis, ou seja, suas correlações, e consequentemente apresenta o problema da quebra dos blocos construtivos. Assim, a probabilidade de amostragem das variáveis da solução é decomposta pelo produto de cada uma singularmente como visto na Equação 2.1 (HAUSCHILD; PELIKAN, 2011). Um exemplo desse modelo é o Algoritmo de Distribuição Marginal Univariada (UMDA) (MÜHLENBEIN; PAASS, 1996) e o Algoritmo Genético Compacto (CGA) (HARIK; LOBO; GOLDBERG, 1999). Ambos fornecem a distribuição da probabilidade dos valores das variáveis sem referenciar

as dependências com as demais, ou seja:

$$P(x_1, x_2, \dots, x_n) = P(x_1)P(x_2) \dots P(x_n) \quad (2.1)$$

Diferente do modelo anterior, o modelo de interações multivariado como o próprio nome já enuncia, procura estabelecer uma relação entre as variáveis e evitar o problema da quebra dos blocos construtivos citados anteriormente. Uma vez que se tem conhecimento sobre a interação das variáveis, é factível utilizar essa informação para estimar a distribuição dos valores de cada grupo de variáveis. Essa informação acerca da relação entre as variáveis porém, nem sempre é conhecida *a priori*. Então, é tarefa do EDA encontrar a relação que melhor represente o conjunto de variáveis para o problema.

O Algoritmo de Otimização Bayesiano (BOA) (PELIKAN; GOLDBERG; CANTÚ-PAZ, 1999), juntamente com o Algoritmo de Otimização Bayesiano Hierárquico (HBOA) (PELIKAN; GOLDBERG; TSUTSUI, 2003), são exemplos desse modelo e serão introduzidos nas seções seguintes. Por fim, vale destacar que a maior parte dos EDAs oferece vantagens através do uso de modelos probabilísticos, na representação da população de maneira mais eficiente em relação a outros AEs e meta-heurísticas (HAUSCHILD; PELIKAN, 2011).

Tabela 3 – Soluções que constituem a população inicial, onde v_i representa o valor para a i -ésima posição da solução.

<i>Rótulo da Solução</i>	x_1	x_2	x_3	<i>fitness</i>
$\mathbf{x}_1$	1	0	0	10
$\mathbf{x}_2$	1	0	1	8
$\mathbf{x}_3$	0	1	1	7
$\mathbf{x}_4$	0	0	1	4
$\mathbf{x}_5$	1	1	1	2
$\mathbf{x}_6$	0	1	0	2

Tabela 4 – Distribuição de probabilidades para variáveis com valores recorrentes entre as soluções mais bem avaliadas. Note que, a probabilidade para uma opção alternativa de valor de variável é complemento da corrente e deve somar 1.

$P(x_1) = 1$	$P(x_2) = 0$	$P(x_3) = 1$
0.666	0.666	0.666

2.3.1 Algoritmo de Otimização Bayesiano

O Algoritmo de Otimização Bayesiano (BOA), como citado na seção anterior, faz parte da classe de algoritmos de estimação de distribuição. Descrito inicialmente por Pelikan et al. (PELIKAN; GOLDBERG; CANTÚ-PAZ, 1999; PELIKAN; GOLDBERG; CANTÚ-PAZ, 2000), esse algoritmo utiliza Redes Bayesianas (*Bayesian Networks*, BNs) para expressar a relação de dependência entre as variáveis da solução através de um VIG, onde cada variável de decisão corresponde a um vértice do grafo e sua relação fica evidente através de arestas direcionadas, ou setas, que as conectam.

A forma como o BOA estima a distribuição de probabilidade das variáveis combina o conhecimento *a priori* do problema, que para esse modelo não é essencial, e o conjunto de soluções promissoras selecionadas a partir do conjunto inicial (PELIKAN; GOLDBERG; CANTÚ-PAZ, 1999). Ou seja, quando o grafo de interações não é conhecido, o BOA estima este grafo a partir das soluções geradas. O mesmo pode ser aplicado então, a problemas em que a representação da solução se dá através de uma cadeia de *strings* de tamanho fixo, sobre um alfabeto binário, ou seja $\{0, 1\}$, além de problemas do tipo caixa-preta, em que não é estritamente introduzido o modelo interno exato de avaliação dos dados (EIBEN; SMITH, 2008; PELIKAN; GOLDBERG, 2002).

Os passos do algoritmo BOA são descrito a seguir e estão ilustrados na Figura 6.

1. O primeiro passo do algoritmo BOA é a geração aleatória do conjunto de soluções candidatas, $C(t)$, onde t corresponde a geração.
2. Logo em seguida são selecionadas as soluções do conjunto $C(t)$ com maior *fitness*. Para os experimentos de Pelikan et al. (PELIKAN; GOLDBERG; CANTÚ-PAZ, 1999), foi utilizado o método de seleção por truncamento descrito por De Jong (JONG, 2006) e que seleciona 50% das melhores soluções.
3. Mediante isso as soluções selecionadas são então usadas para construção do modelo através da BN, que utiliza uma métrica de pontuação e um procedimento de busca como descrito em Heckerman et al. (HECKERMAN; GEIGER; CHICKERING, 1995).
4. Após a construção da BN, as soluções são amostradas através da distribuição conjunta da mesma para assim compor um novo conjunto de soluções, $C(t + 1)$.
5. Com isso, acontece a substituição das soluções do conjunto $C(t)$ pelas novas soluções do conjunto $C(t) \leftarrow C(t + 1)$.
6. Se a condição de parada definida para o algoritmo for alcançada, a execução é então encerrada, do contrário é realizado uma nova execução a partir do passo 2.

Figura 6 – Funcionamento do BOA.

2.3.1.1 Redes Bayesianas

As redes Bayesianas (PEARL, 2014) são modelos probabilísticos baseados em grafos que reproduzem um relacionamento de dependências condicionais entre as variáveis do problema. Tal rede é definida como um grafo direcionado, também conhecido como dígrafo, acíclico, no qual os vértices ou nós do grafo correspondem as variáveis do problema, que para o caso em questão, são os genes do cromossomo, e um arco entre os nós no sentido V_i para V_j , equivale a V_i "causa" V_j (MURPHY, 2001). Matematicamente a BN pode ser descrita como uma associação das distribuições de probabilidade conjuntas, dadas por:

$$P(X) = \prod_{i=0}^{n-1} p(x_i | \Pi_{x_i}) \quad (2.2)$$

onde,

- $X = (x_0, \dots, x_{n-1})$ é um vetor de variáveis,
- Π_{x_i} é o conjunto de parentes de x_i , ou seja, os vértices que apontam para x_i ,
- $p(x_i | \Pi_{x_i})$ é a probabilidade condicionada de x_i as variáveis de Π_{x_i} .

2.3.1.2 Construindo Redes Bayesianas

Os componentes fundamentais de uma rede Bayesiana são (PEARL, 2014):

- **Estrutura** - como descrito anteriormente, sua estrutura é composta por um grafo dígrafo acíclico, onde cada nó corresponde a uma variável de decisão do problema.

Dessa forma, para que uma rede Bayesiana seja criada, é necessário que a estrutura codificada de dependência entre as variáveis seja determinada.

- **Parâmetros** - corresponde a um conjunto de probabilidades condicionais relacionados às variáveis do problema, para o nosso caso, as posições do cromossomo. Assim como sua estrutura, os valores das probabilidades condicionais devem ser aprendidos.

Apesar de se tratar de uma abordagem probabilística complexa, a real complexidade se encontra na aprendizagem da estrutura da rede, uma vez que os parâmetros, portanto os valores das probabilidades condicionais, podem ser obtida através do *fitness* das soluções candidatas e frequência dos dados modelados. Então, para construir a estrutura de uma BN são necessários:

1. **Uma métrica de *score*** - responsável por mensurar o quão boa a rede foi modelada e onde o conhecimento prévio do problema pode ser incorporado, para reforçar sua qualidade, uma vez que a melhor BN, em termos de pontuação e dado um conjunto de variáveis, é um problema NP, sendo difícil identificar a melhor solução (SANTOS et al., 2007; CROCOMO, 2012).
2. **Um procedimento de busca** - responsável por, vasculhar todo o espaço de busca possível de combinação para estrutura da rede e através da métrica de *score* escolhida, encontrar a rede com o melhor avaliação (PELIKAN; GOLDBERG, 2002). Novamente, o conhecimento prévio sobre o problema trabalhado pode ser inserido para restringir o espaço de busca do método utilizado.

Para o BOA de (PELIKAN; GOLDBERG; CANTÚ-PAZ, 1999), um algoritmo de busca heurística chamado K2 (COOPER; HERSKOVITS, 1992) é utilizado como métrica de *score* do aprendizado da rede Bayesiana (B_s), sobre o quão bom o modelo gerado representa um conjunto de dados (D), objetivando maximizar assim, a qualidade mediante avaliação entre todas as possibilidades $P(B_s, D)$. Como procedimento de busca, um algoritmo de busca gulosa simples, mostrou-se efetivo e tem sido usado em tarefas difíceis de aprendizado de máquina, buscando aprimoramentos na rede utilizada até que não seja mais possível realizar mais melhorias (PELIKAN; GOLDBERG, 2002).

Bayesian Partition Crossover

Nos capítulos anteriores foram apresentados os conceitos dos algoritmos genéticos e dos algoritmos de estimação distribuição, ambos algoritmos evolutivos que diferem na forma como amostram novos indivíduos da população de soluções candidatas. O conhecimento sobre a estrutura mais adequada do problema pode otimizar a forma com que os AEs trabalham, além de criar novas abordagens como a descrita neste trabalho. Dessa forma, através do conceito de recombinação utilizado pelo PX e do uso das redes Bayesianas no BOA para modelagem da estrutura das variáveis de decisão do problema, é proposto neste trabalho um novo operador de recombinação denominado Operador de Partição Bayesiano (*Bayesian Partition Crossover*, BPX). Esse novo operador faz uso da informação da estrutura do problema modelada para realizar suas recombinações.

3.1 BPX

Repare que no PX descrito na Seção 2.2.2, assim como nos outros operadores de recombinação por decomposição já propostos, é necessário o conhecimento das relações entre as variáveis de decisão, ou seja, o grafo de interações deve ser conhecido *a priori*. O novo operador de recombinação BPX proposto neste trabalho, é baseado no operador PX e utiliza redes Bayesianas, assim como o BOA, para estimar o grafo de interação entre as variáveis do problema escolhido. Uma vez estimado o grafo, torna-se possível então aplicar o conceito de partições recombinantes nos indivíduos da população.

A grande vantagem é que, diferentemente do PX, este operador pode ser aplicado a qualquer problema pseudo-Booleano, uma vez que não é necessário conhecer *a priori* o grafo de interações entre as variáveis de decisão do problema. O PX original é utilizado apenas em problemas do tipo caixa-cinza já que é necessário conhecer *a priori* as relações entre as variáveis. Já o BPX pode ser aplicado em problema do tipo caixa-preta, já que o conhecimento sobre as relações entre as variáveis será estimado durante o processo de otimização.

Diferentemente do BOA todavia, a confecção de uma nova BN não se dá a cada nova geração, mas sim de acordo com um intervalo fixo entre gerações determinado previamente através de experimentações realizadas. Dessa forma, assim como acontece no BOA é esperado que os erros do modelo de rede gerado no BPX, também sejam corrigidos com modelos de gerações futuras. A BN criada utiliza do algoritmo K2 (COOPER; HERSKOVITS, 1992) como métrica de *score* e busca gulosa como procedimento de busca, assim como abordado no BOA descrito anteriormente.

No Algoritmo 1 abaixo, é possível analisar com mais detalhes através do pseudo-código do PX, descrito na Seção 2.2.2 e disponível em (TINÓS; WHITLEY; CHICANO, 2015), o passo a passo das ações que o operador realiza e seus respectivos parâmetros. Assumindo que o VIG está disponível, o primeiro passo do operador (linha 6) é transferir os bits comuns em ambos os pais para a prole. Assim, na linha 7, o grafo de recombinação é criado e o subconjunto dos componentes conectados π é encontrado usando busca em largura¹. Na linha 8 a 16, a avaliação parcial, g^i , da solução parcial dentro do componente conectado π para cada pai é empregada para selecionar os bits correspondentes de um dos pais. Dentro de cada componente (conectado) recombinante, a prole herda os bits de apenas um dos pais.

Algorithm 1 Partition Crossover (PX) - Assumindo Maximização

```

1: função PX( $f, G, x, y$ )
2: Entrada:  $f$                                 ▷ função pseudo-Booleana
3: Entrada:  $G$                                 ▷ grafo de interação das variáveis de decisão do problema
4: Entrada:  $x$  e  $y$                             ▷ soluções candidatas progenitoras
5: Saída:  $z$                                 ▷ prole
6:  $z \leftarrow x \wedge y$                         ▷ Transferindo os bits comuns
7:  $\pi \leftarrow \text{computandoPartição}(G[x \oplus y])$ 
8: Seja  $f = \sum_{i=1}^{|\pi|} g^i$ 
9:   para  $i = 1$  até  $|\pi|$  faça
10:     se  $g^i(x) > g^i(y)$  então
11:        $z \leftarrow z \vee (x \wedge \pi_i)$ 
12:     senão
13:        $z \leftarrow z \vee (y \wedge \pi_i)$ 
14:     fim se
15:   fim para
16: fim função

```

O PX é um operador guloso: para cada componente, é selecionada a solução parcial que resulta em uma melhor avaliação parcial g^i . Assim, se p componentes recombinantes

¹ Outro algoritmo para encontrar os componentes conectados pode ser utilizado nesse passo.

são encontrados, PX retorna o melhor entre 2^p proles possíveis. Por exemplo, se $p = 10$, o melhor entre $2^{10} = 1024$ indivíduos é encontrado. A complexidade de tempo para uma aplicação do PX é $O(N)$ quando o grau máximo do vértice é limitado pela constante (K). Já no Algoritmo 2 é possível analisar os passos executados pelo operador BPX, aproveitando-se da estrutura de recombinação do PX e da criação das redes Bayesianas do BOA. Um dos parâmetros configuráveis da BN é o grau máximo dos vértices, descrito na linha 6 do algoritmo. Esse atributo restringe as dependências condicionais das variáveis de decisão do problema, limitando a apenas 5 relações de dependência para os experimentos, ou seja, o número de arestas incidentes em cada vértice. Um número alto de grau dos vértices, implica em um espaço de busca maior de combinações e consequentemente um custo computacional mais alto (SCANAGATTA et al., 2015). O valor para o grau máximo do vértice utilizado neste trabalho foi definido através de experimentos realizados em diversas instâncias.

Já durante a execução do operador BPX, caso a rede Bayesiana não tenha sido gerada anteriormente, a mesma pode ser construída em intervalos de gerações para diminuir o esforço computacional. Para os experimentos desse trabalho, o intervalo definido para confecção da BN foi de 300 gerações como apresentado no Algoritmo 2. Isto posto, é então realizada sua construção, linha 10, seguindo os conceitos descritos na Seção 2.3.1.2. Caso a BN já esteja disponível, seu grafo direcionado é então utilizado como parâmetro para o PX e assim são recombinados os indivíduos x e y para formação de z , linha 12.

Algorithm 2 Bayesian Partition Crossover (BPX)

```

1: função BPX( $f, S, M, B, d, x, y$ )
2: Entrada:  $f$                                 ▷ função pseudo-Booleana
3: Entrada:  $S$                                 ▷ soluções promissoras selecionados por torneio
4: Entrada:  $M$                                 ▷ métrica de score da BN
5: Entrada:  $B$                                 ▷ algoritmo de busca no espaço para BN
6: Entrada:  $d$                                 ▷ grau máximo dos vértices da BN
7: Entrada:  $x$  e  $y$                             ▷ soluções candidatas progenitoras
8: Saída:  $z$                                 ▷ prole
9:   se  $G$  não existe ou  $n_{gen} \% 300$  igual a 0 então
10:      $G \leftarrow \text{RedeBayesiana}(S, M, B, d)$   ▷ construindo a rede e recuperando o grafo
       direcionado
11:   senão
12:      $z \leftarrow \text{PX}(f, G, x, y)$             ▷ executando o PX com o grafo gerado pela BN
13:   fim se
14: fim função

```

Para a experimentação com o modelo NK foi confeccionado uma variação do

operador BPX denominada de BPEX, onde as subfunções de avaliação do modelo PB, que outrora são definidas no início da execução e preservadas até o fim, são somadas a dois arranjos dinâmicos. Essa dinamicidade é observada através da utilização de estruturas para armazenar a frequência de aparição das máscaras binárias de cromossomo dos indivíduos e o valor estimado de cada subfunção.

Logo, ao ser realizado o cálculo do *fitness* de cada indivíduo, a toda nova aparição de uma máscara binária do mesmo, sua posição correspondente na tabela de frequência é somada a 1. Bem como sua posição na tabela de estimação de subfunção recebe o valor integral do *fitness* do indivíduo, computado de acordo com a função padrão do modelo. Em seguida, o valor atribuído a cada partição dos indivíduos, corresponde ao somatório da média entre o valor da estimação da subfunção e sua frequência correspondente.

Dessa forma, são escolhidas qual a melhor partição a prole vai herdar de cada pai e apesar do mecanismo de estimação, o real valor de cada indivíduo a ser considerado no final é o obtido através da função de *fitness* padrão do modelo NK. Todavia, essa abordagem com estimação do BPEX em comparação com o BPX, é limitada, visto que sua aplicação é restrita a problemas em que se conhece o valor de n e de K , caso contrário o custo computacional para alocação do vetor dinâmico seria muito alto, além do tempo de execução do mesmo.

Além da variação do BPX, o BPEX, duas outras abordagens foram utilizadas neste trabalho. Desta vez, aplicadas aos problemas da Mochila 0-1 e funções armadilhas, combinando os operadores de recombinação uniforme e 2-pontos com o BPX, denominados BPX+UN e BPX+2PT respectivamente. Essas combinações foram utilizadas na pretensão de fazer proveito das características exploradoras desses operadores tradicionais, juntamente com o potencial do novo operador proposto neste trabalho.

No Algoritmo 3 é possível observar com mais detalhes sua execução, onde a condição se a geração é número par ou ímpar, intercala a utilização de cada operador com o BPX. Na linha 16, é realizado a chamada para o operador combinando sua recombinação com o BPX, sendo utilizado o uniforme ou 2-pontos por experimento.

Algorithm 3 Bayesian Partition Crossover (BPX) com Uniforme ou 2-Pontos

```

1: função BPX( $f, S, M, B, d, x, y$ )
2: Entrada:  $f$                                 ▷ função pseudo-Booleana
3: Entrada:  $S$                                 ▷ soluções promissoras selecionados por torneio
4: Entrada:  $M$                                 ▷ métrica de score da BN
5: Entrada:  $B$                                 ▷ algoritmo de busca no espaço para BN
6: Entrada:  $d$                                 ▷ grau máximo dos vértices da BN
7: Entrada:  $x$  e  $y$                             ▷ soluções candidatas progenitoras
8: Saída:  $z$                                 ▷ prole
9: seja  $t \leftarrow 0$                             ▷ contador de gerações
10:   se  $G$  não existe ou  $n_{gen} \% 300$  igual a 0 então
11:      $G \leftarrow \text{RedeBayesiana}(S, M, B, d)$     ▷ constrói a rede e recupera o grafo dir
12:   senão
13:     se  $t \% 2 = 0$  então
14:        $z \leftarrow PX(f, G, x, y)$         ▷ executando o PX com o grafo gerado pela BN
15:     senão
16:        $z \leftarrow UN \text{ ou } 2\text{-Pontos}(x, y)$ 
17:     fim se
18:   fim se
19: fim função

```

3.2 Experimentos

Para a experimentação com o modelo NK, foi desenvolvido uma variação do operador proposto neste trabalho denominada BPEX descrito na seção anterior. Essa variação do operador estima as subfunções de avaliação do modelo NK, em busca de otimizar a escolha da melhor partição que a prole irá herdar de seus progenitores. Já para o experimento com os problemas da mochila 0-1 e armadilha que serão descritos no próximo capítulo, foi experimentada outras duas combinações de operadores de recombinação na mesma execução, alternando entre gerações ímpares e pares sua aplicação. As combinações foram BPX com o uniforme (BPX+UN) e BPX com o 2-pontos (BPX+2PT), descritas anteriormente.

A etapa de preparação para execução do algoritmo acontece quando os indivíduos que compõem a população, são gerados de maneira aleatória. Em seguida, uma variação do algoritmo de busca local é aplicado, realizando transformações singulares nos bits de acordo com um vetor permutado das posições dos bits de cada indivíduo. Ao ser feita a transformação e não ser alcançado um melhoramento no *fitness*, o bit é revertido novamente ao valor original. Esse processo se repete até ser percorrida todas as posições possíveis dos indivíduo da população ou realizado um aprimoramento no indivíduo e denomina-se busca local com primeiro melhoramento *first improvement* - (*LSFI*). Embora o BPX realize a modelagem da população através das redes Bayesianas em seu ciclo de execução, as etapas executadas até o critério de parada ser atingido são simples:

1. As soluções da população ($P(t)$) são avaliadas para identificação dos indivíduos que são mais prósperos de acordo com seu *fitness*. A seleção utilizada nos experimentos foi a de torneio, onde a partir de um valor de janela (ω) igual a 3 definido *a priori*, são selecionados ω indivíduos aleatórios na população para competir de acordo com seu *fitness*. O indivíduo com maior *fitness* é então selecionado para ser progenitor.
2. A cada 300 gerações, valor definido após experimentações, é gerada uma nova rede Bayesiana para representar o modelo, ou seja, para gerar o grafo de interações entre as variáveis. Para que a mesma seja gerada, é efetuada novamente uma seleção por torneio na população $P(t)$ até compor uma amostra de mesmo tamanho. Uma restrição de grau máximo dos vértice que compõem a rede Bayesiana é definida com valor igual a 5, bem como a métrica K2 de *score* e o algoritmo guloso para construção da BN, descritos anteriormente.
3. Uma vez criada a rede Bayesiana, os indivíduos selecionados são então recombinados utilizando o BPX. O uso desse operador se faz possível uma vez que o grafo de decisão utilizado como estrutura da rede Bayesiana é utilizado para descrever as interações entre as variáveis de decisão.

4. Assim como a recombinação, a mutação é uma operação que está sujeita a uma porcentagem que representa a taxa de chances de acontecer o evento. Para os experimentos deste trabalho, foi definido que ambas as operações aconteceriam de forma intercalada, ou seja, caso aconteça a recombinação, a mutação não irá ocorrer e vice-versa. Isto é interessante pois o PX já estima o *fitness* do indivíduo durante sua execução.
5. Os novos indivíduos são então inseridos na população, porém, assim como na etapa de criação da BN, um intervalo foi definido para que fosse realizado dois tipos de substituição na população em alternância: a substituição completa e a com uma migração aleatória. Na substituição completa, todos os novos indivíduos gerados pela etapa anterior são inseridos na nova população sem exceção.
6. Em vias de preservar a diversidade na população, a abordagem de migração aleatória dos indivíduos descrita em (TINÓS; YANG, 2007) como substituição na população foi adotada. Nessa abordagem, uma porcentagem dos indivíduos é substituída por novos gerados aleatoriamente, em analogia a migração de vizinhos antigos e a chegada de novos na vizinhança. Para os experimentos deste trabalho, foi definido que 90% dos indivíduos seriam migrados aleatoriamente e os outros 10%, incluso o elitista, seriam conservados. Essa substituição acontecerá a cada intervalo de 10 gerações em todas as execuções.

O Algoritmo 4, apresenta por sua vez o pseudo-código do AG com o novo operador proposto nesta pesquisa. Observe que na linha 3 é executado uma busca local do tipo LSFI. Com isso, as soluções candidatas são aprimoradas tal como descrito anteriormente e logo em seguida prosseguem na estrutura de fluxo do AG. Para garantir a substituição das BNs ao decorrer da execução foi utilizado os intervalos de gerações. Na linha 5 são apresentadas as condições a serem alcançadas, número de gerações seja igual a 0 ou a cada 300 gerações, caso contrário a última BN criada continua a ser utilizada.

Para a BN gerada na linha 6 é realizada um seleção por torneio com 3 indivíduos, seguido das métricas e restrições do BOA, como descrito anteriormente. Na linha 7, a prole é gerada utilizando recombinação ou mutação de maneira concorrente. Caso seja efetuada a recombinação (taxa de recombinação Rec_{rate} maior que um valor aleatório gerado R), o operador BPX utiliza da lógica do operador PX, como visto no Algoritmo 2, usando da BN como grafo de interação das variáveis de decisão do problema. Repare na linha 8, ao realizar a chamada do BPX a BN já está criada, dessa forma é possível adaptar o *input* do BPX para receber a BN e utilizar o grafo direcionado, sem necessidade dos demais parâmetros de criação da BN.

Por fim, na linha 11, é definido um intervalo para aplicação da migração voluntária e posterior otimização da nova população com LSFI, linhas 12-13. Ao final é realizada

uma substituição dos novos indivíduos na população $P(t + 1)$, linha 14, e caso a condição de parada não seja alcançada, é repetido o passo 4 até a ser.

Algorithm 4 AG com Bayesian Partition Crossover (BPX) - Assumindo Maximização

- 1: seja $t \leftarrow 0$
 - 2: inicia população aleatória $P(0)$
 - 3: executa LSFI em $P(0)$
 - 4: seleciona os indivíduos mais promissores $S(t)$ de $P(t)$
 - 5: **if** $t \% 300 = 0$ **or** $t = 0$ **then**:
 - 6: $BN \leftarrow \text{RedeBayesiana}(S(t), M, B, d)$
 - 7: **if** $Rec_{rate} > R$ **then**:
 - 8: $P(t)_i \leftarrow \text{BPX}(BN, x_i, y_i)$
 - 9: **else** :
 - 10: muta o indivíduo
 - 11: **if** $t \% 25 = 0$ **then**:
 - 12: garante a diversidade de $P(t)_i$ com migração voluntária
 - 13: executa LSFI em $P(t)$
 - 14: substitui $P(t)$ em $P(t + 1)$ e define $t \leftarrow t + 1$
 - 15: se a condição de parada não for alcançada, repita novamente o passo 1 em diante
-

Resultados

O novo operador de recombinação por decomposição para problemas pseudo-Booleanos proposto nesta pesquisa, e denominado BPX, foi aplicado inicialmente para fins de validação no modelo NK. Em seguida, o BPX foi aplicado em problemas pseudo-Booleanos em que o grafo real de interações não é conhecido. Dessa forma, foram escolhidos os problemas da Mochila 0-1, descrito na Seção 1.1.1 e uma função armadilha enganosa, descrito na Seção 1.1.2. Os resultados obtidos com o BPX neste problema foram então comparados com os operadores de recombinação Uniforme e de 2-pontos em todos os problemas. Para o modelo NK, o PX original foi também utilizado para comparação já que neste problema o grafo de interações é conhecido *a priori*.

4.1 Critérios e Recursos Utilizados

O critério de avaliação comumente utilizado para determinar a eficiência dos algoritmos evolutivos, é o valor de *fitness* dos indivíduos. Por se tratar de experimentos que possuem sequências de execuções cada qual com uma sequência de gerações de tamanho pré-definido, neste trabalho, foram adotados os seguintes critérios:

1. **Valor médio de *fitness*:** durante todas as execuções são armazenados os *fitness* dos indivíduo elitista de cada execução, ou seja, o valor *fitness* do indivíduo mais apto. Ao final de todas as execuções, os melhores indivíduos de cada geração são somados e é calculada uma média aritmética.
2. **Porcentagem de indivíduos melhores que os pais:** a cada geração após recombinação e avaliação da nova prole, é verificado se a mesma possui *fitness* maior que seus progenitores, dada vez que o filho é melhor que o pais, um contador (C_{pr}) é incrementado. Independente a ser superior ou não porém, é incrementado também a um contador do total de recombinações (c_{to}) para possibilitar o cálculo da média.

Ao final de cada execução, é então calculado a média aritmética do total da prole melhor que os progenitores gerados por recombinação ($\frac{c_{pr}}{c_{to}}$). Essa análise todavia não acontece até atingir o número de soluções máximas ou a condição de parada, e sim possui uma restrição de até alcançar 1000 gerações. Isso se deve ao fato de operadores de transformação em AEs gerarem mais descendentes bons nos inícios das execuções do que no fim delas. O resultado do critério então é a média das médias de cada execução.

3. **Porcentagem de indivíduos melhores que o melhor:** após recombinação e avaliação do *fitness* da nova prole, é realizada a comparação com o *fitness* do indivíduo elitista; caso a prole seja superior, o novo indivíduo é definido como o novo elitista e acrescido o valor de 1 ao contador (c_p). Ao final de cada execução, é então calculado uma média aritmética do total da prole melhor que o elitista gerados por recombinação ($\frac{c_p}{c_{to}}$). Assim como no critério anterior porém, a restrição é de 1000 gerações como limite de análise. O resultado do critério então é a média das médias de cada execução.
4. **Tempo de execução:** em vias de analisar o tempo gasto durante cada execução dos mais diversos operadores abordados neste trabalho, são recuperados em segundos a média de cada execução, cada qual com seu número definido de gerações. Este critério de avaliação todavia, foi aplicado apenas nos experimentos com o modelo NK.

Além dos demais critérios descritos acima, foram calculados outros relacionados a quantidade de indivíduos transformados através da mutação e recombinação, esses valores todavia não apresentaram resultados fora do esperado e/ou consideráveis. Os procedimentos experimentais comuns a todos os testes realizados durante a pesquisa e definidos a partir de experimentos prévios são:

- O critério de parada definido foi o número de gerações previamente definido como sendo 20 mil.
- O operador de seleção utilizado foi o seleção por Torneio, com tamanho da janela igual a 3. A taxa de recombinação definida foi de 60%.
- O operador de mutação utilizado foi o bit-flip singular, com taxa de mutação definida como $\frac{1}{n}$, onde n é o tamanho da solução candidata.
- As operações de recombinação e mutação ocorrem de forma alternada, ou seja, ao ser realizada a recombinação não é feita a mutação e assim sucessivamente.
- Em cada um dos experimentos realizados, foi considerado o total de 50 execuções.

- Para os experimentos envolvendo o modelo NK e Mochila 0-1, foi adotado o algoritmo de busca local com primeiro melhoramento (LSFI) descrito na seção anterior.

Ademais, a linguagem de programação utilizada foi Java e os experimentos foram executados sob as mesmas condições em relação as configurações de hardware, dentro de um servidor com 1 processador Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz com 8 núcleos e 16 *threads* e 64 GB de RAM do Departamento de Computação e Matemática da Universidade de São Paulo.

4.2 Grafos de Interação Real e Obtido com Modelo NK

As redes Bayesianas, como apresentado anteriormente, são utilizadas para descrever dependências probabilísticas entre variáveis do problema ou do sistema. Essas redes possuem a estrutura de grafos de decisão direcionados, acíclicos, no qual cada vértice representa uma variável e os arcos a influência direta entre as variáveis, o que nos permite interpretar dependências verdadeiras, refletindo a percepção natural do sistema (PEARL, 2014).

Para fins de validação do BPX, o mesmo foi aplicado inicialmente no modelo NK, para os casos com vizinhança por adjacência e aleatória com $n = 15$ e $K = 3$. Esses valores foram escolhidos como forma de melhor apresentar visualmente os exemplos em questão, uma vez que nos demais experimentos, descritos na Seção 4.3, foram utilizados os valores para n de 100, 300 e 500. Já para K , os valores de 1,2,3,5 e 10 foram adotados.

Na Figura 7 é possível observar o grafo esperado para o Modelo NK com vizinhança por adjacência e o valor de n e k definidos anteriormente. Já na Figura 8 é exposto o grafo gerado pelo BPX no intervalo de gerações igual a 300.

As Figuras 9 e 10 respectivamente, apresentam o grafo esperado para o modelo aleatório e o obtido pelo BPX também no intervalo de geração igual a 300. Nota-se, que os modelos estimados pelo método Bayesiano em geral diferem dos grafos de interação levantados pela análise da função de *fitness*. Isto principalmente porque:

- o grafo de interações (real) não leva em conta o valor das contribuições de *fitness* relacionadas às interações. Assim, apesar de eventualmente a variável v_a interagir com v_b , a contribuição desta interação pode ser muito pequena quando comparada com outras interações;
- a rede Bayesiana é construída a partir de uma amostra pequena (tamanho da população), fazendo que algumas interações sejam priorizadas em relação a outras e

que outras possam ser incorretamente estimadas. Além disso, a estimação leva em conta apenas a situação atual.

- iii - a rede Bayesiana é construída levando-se em conta os valores das variáveis ao passo que o grafo de interações reais não o é. Vale ressaltar, entretanto, que apesar dos modelos serem diferentes, as ligações estimadas podem representar indicativos de que as variáveis ligadas através de arestas devam ficar juntas quando ocorrem as recombinações.

Figura 7 – Grafo direcionado das variáveis de decisão do problema esperado para o modelo NK com $n = 10$, $K = 3$ e vizinhança por adjacência.

Figura 8 – Grafo direcionado das variáveis de decisão do problema, obtido através das configurações da BN com $n = 10$, $K = 3$ e vizinhança por adjacência.

Figura 9 – Grafo direcionado das variáveis de decisão do problema para o modelo NK com $n = 10$, $K = 3$ e vizinhança aleatória. O grafo gerado de forma aleatória no início da execução do AG permanece o mesmo até o fim em uma execução padrão com outros operadores.

Figura 10 – Grafo direcionado das variáveis de decisão do problema, obtido através das configurações da BN com $n = 10$, $K = 3$ e vizinhança aleatória.

4.3 Resultados: Eficiência dos Operadores

Esta seção está organizada em 3 subseções descrevendo três experimentos envolvendo o modelo NK, o problema da Mochila 0-1 e uma função armadilha enganosa em sequência, todos descritos anteriormente e seguindo as configurações descritas na Seção 4.1. Além da apresentação dos resultados, é discorrido também sobre o que pode ter influenciado os valores obtidos com os mesmos.

4.3.1 Modelo NK

O modelo NK, como descrito na Seção 1.2.1, é um problema pseudo-Booleano k -restrito que apresenta uma relação epistática entre as variáveis de decisão do problema. Isso significa que uma mudança no valor dos genes dos indivíduos que interagem epistaticamente, pode influenciar na perda do ótimo devido a diminuição da contribuição do gene na função de *fitness* (AGUIRRE; TANAKA, 2003). Esse fato torna a otimização de todos os genes simultaneamente uma tarefa difícil, bem como a utilização de operadores de mutação, que apesar de garantir a diversidade da população, assim como os operadores de recombinação de 2-pontos e uniforme não preservam os BBs.

Em ambos os experimentos realizados neste trabalho com o modelo NK, seja com vizinhança por adjacência ou aleatória, os valores definidos para n e K foram todos comparados. Dessa forma, foi possível observar o comportamento do operador BPX e BPEX em relação a outros operadores de recombinação utilizados. Os resultados dos experimentos seguindo os critérios descritos anteriormente podem ser observados nas Tabelas 5 - 12.

4.3.1.1 Uniforme e 2-pontos

O operador uniforme é caracterizado pela sua aptidão em preservar a diversidade na população caminhando por uma multiplicidade de indivíduos no espaço de busca. Todavia, essa habilidade faz com que não sejam preservados os BBs formados por variáveis adjacentes, favorecendo assim o seu rompimento. Já o operador 2-pontos, por possuir um conceito de conservar os BBs adjacentes com cortes de tamanhos fixos nos genes dos indivíduos, apresentou em relação ao uniforme melhores resultados de *fitness* como é possível observar nas Tabelas 5 e 6.

Já na porcentagem de soluções melhores que os pais e melhores que o melhor, apresentados nas Tabelas 7 - 9 houve predominância do operador uniforme em relação ao 2-pontos. Isto pode ser explicado pela diversidade dos indivíduos das populações e a

rápida convergência do operador de 2-pontos para ótimos. Em relação ao custo de tempo, o operador de 2-pontos por efetuar recombinações em blocos de genes dos indivíduos ao invés de percorrer todos os genes da indivíduo como faz o uniforme, apresentando desempenho melhor que o uniforme ao passo que o valor de n é aumentado.

4.3.1.2 PX, BPX e BPEX

O conceito de partição adotado pelo PX, favorece a utilização dos BBs para aprimorar as variáveis de decisão do problema nos indivíduos da população. Essa característica contudo, também favorece com que a prole indivíduos compartilhe de muitas características dos progenitores, evitando assim a criação de diversidade nos indivíduos com o processo de recombinação. Isso no entanto não prejudicou o desempenho do operador PX, que obteve, em geral, melhores resultados de *fitness* se comparado a todos os demais operadores utilizados. Esse desempenho se refletiu também na porcentagem de geração de melhores soluções por recombinação que os progenitores e que o melhor da população, como é possível observar nas Tabelas 7 - 10.

Apesar de ter apresentado desempenho inferior em geral, ambos BPX e BPEX obtiveram frequentemente melhores resultados quando comparados ao PX para $K = 10$. Isto ocorre porque o número de arestas no grafo de interações gerado pela função de *fitness* é muito grande, fazendo que o PX não ache muitos componentes conectados no grafo de interações. Entretanto, no BPX e no BPEX, o grau máximo é limitado, possibilitando uma maior decomposição do grafo de interações.

Através da análise do conjunto de Tabelas 5 - 10, fica claro que apesar dos novos operadores propostos (BPX e BPEX) não obterem sempre os melhores resultados de *fitness* em relação aos operadores tradicionais, os mesmos produziram com frequência soluções melhores que o elite e que os progenitores através da recombinação. Como era de se esperar, o PX obteve os melhores resultados para recombinações eficientes (filhos melhores que os pais e filhos melhores que o melhor indivíduo já encontrado). Na sequência, os melhores resultados foram para BPX e BPEX.

Uma explicação possível para o fato dos AGs com BPX e BPEX obterem pior desempenho em relação ao *fitness* quando comparados com os AGs com recombinação de 2-pontos e uniforme, apesar de BPX e BPEX obterem um número maior de recombinações eficientes, é que os algoritmos resultantes por possuírem características gulosas, teriam enviesado a busca por novas soluções. Isso no entanto pode ter sido suavizado com mecanismos de preservação de diversidade, como a dinâmica de migração aleatória. O custo de tempo do operador BPX e BPEX como apresentado nas Tabelas 11 e 12 ultrapassou os demais. Isso se deve ao fato de realizarem operações mais complexas como a criação das redes Bayesianas a cada intervalo definido.

Tabela 5 – Tabela com valor médio de *fitness* para 50 execuções do GA com Modelo NK, vizinhança por adjacência e condição de parada 20 mil gerações. O símbolo *s* indica que o valor encontrado é estatisticamente significativo, seguido de + se o BPX obteve melhor resultado ou – caso contrário.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.7169 ± 0.0172 (-)	0.7184 ± 0.0173 (-)	0.7187 ± 0.0172 (-)	0.7162 ± 0.0173	0.7164 ± 0.0175 (-)
100	2	0.7385 ± 0.0136 (+)	0.7459 ± 0.0140 (s-)	0.7479 ± 0.0138 (s-)	0.7394 ± 0.0160	0.7381 ± 0.0152 (+)
100	3	0.7469 ± 0.0135 (-)	0.7577 ± 0.0113 (s-)	0.7630 ± 0.0112 (s-)	0.7442 ± 0.0141	0.7453 ± 0.0134 (-)
100	5	0.7444 ± 0.0139 (-)	0.7620 ± 0.0111 (s-)	0.7670 ± 0.0113 (s-)	0.7417 ± 0.0120	0.7438 ± 0.0136 (-)
100	10	0.7296 ± 0.0118 (-)	0.7407 ± 0.0100 (s-)	0.7245 ± 0.0145 (+)	0.7288 ± 0.0122	0.7276 ± 0.0122 (+)
300	1	0.7158 ± 0.0100 (-)	0.7172 ± 0.0099 (-)	0.7201 ± 0.0098 (s-)	0.7155 ± 0.0102	0.7156 ± 0.0099 (-)
300	2	0.7323 ± 0.0109 (+)	0.7374 ± 0.0103 (s-)	0.7473 ± 0.0095 (s-)	0.7336 ± 0.0106	0.7322 ± 0.0110 (+)
300	3	0.7379 ± 0.0092 (-)	0.7438 ± 0.0080 (s-)	0.7632 ± 0.0067 (s-)	0.7379 ± 0.0082	0.7377 ± 0.0079 (+)
300	5	0.7345 ± 0.0085 (+)	0.7415 ± 0.0071 (s-)	0.7596 ± 0.0059 (s-)	0.7351 ± 0.0076	0.7326 ± 0.0084 (+)
300	10	0.7144 ± 0.0080 (s+)	0.7220 ± 0.0077 (s-)	0.7138 ± 0.0080 (s+)	0.7178 ± 0.0077	0.7142 ± 0.0069 (s+)
500	1	0.7114 ± 0.0082 (-)	0.7132 ± 0.0090 (-)	0.7199 ± 0.0084 (s-)	0.7112 ± 0.0085	0.7105 ± 0.0093 (+)
500	2	0.7250 ± 0.0079 (-)	0.7291 ± 0.0086 (s-)	0.7473 ± 0.0074 (s-)	0.7247 ± 0.0091	0.7246 ± 0.0087 (+)
500	3	0.7280 ± 0.0063 (+)	0.7341 ± 0.0069 (s-)	0.7641 ± 0.0047 (s-)	0.7286 ± 0.0056	0.7286 ± 0.0072 (+)
500	5	0.7259 ± 0.0070 (-)	0.7289 ± 0.0058 (s-)	0.7575 ± 0.0055 (s-)	0.7241 ± 0.0063	0.7231 ± 0.0057 (+)
500	10	0.7046 ± 0.0058 (+)	0.7090 ± 0.0066 (s-)	0.7038 ± 0.0063 (+)	0.7057 ± 0.0065	0.7052 ± 0.0060 (+)

Tabela 6 – Tabela com valor médio de *fitness* para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.7119 ± 0.0184 (+)	0.7121 ± 0.0186 (+)	0.7146 ± 0.0179 (-)	0.7121 ± 0.0187	0.7124 ± 0.0181 (-)
100	2	0.7391 ± 0.0120 (-)	0.7375 ± 0.0136 (-)	0.7420 ± 0.0132 (-)	0.7366 ± 0.0144	0.7369 ± 0.0131 (-)
100	3	0.7498 ± 0.0164 (-)	0.7496 ± 0.0168 (-)	0.7456 ± 0.0150 (-)	0.7450 ± 0.0162	0.7483 ± 0.0168 (-)
100	5	0.7478 ± 0.0184 (+)	0.7525 ± 0.0143 (-)	0.7470 ± 0.0140 (+)	0.7498 ± 0.0146	0.7471 ± 0.0155 (+)
100	10	0.7272 ± 0.0137 (-)	0.7295 ± 0.0128 (-)	0.7268 ± 0.0146 (+)	0.7268 ± 0.0153	0.7238 ± 0.0117 (+)
300	1	0.7120 ± 0.0112 (-)	0.7115 ± 0.0112 (-)	0.7157 ± 0.0109 (s-)	0.7111 ± 0.0111	0.7114 ± 0.0112 (-)
300	2	0.7308 ± 0.0105 (-)	0.7297 ± 0.0103 (+)	0.7352 ± 0.0102 (s-)	0.7305 ± 0.0098	0.7293 ± 0.0090 (+)
300	3	0.7395 ± 0.0087 (-)	0.7390 ± 0.0111 (-)	0.7366 ± 0.0091 (+)	0.7372 ± 0.0086	0.7380 ± 0.0106 (-)
300	5	0.7407 ± 0.0079 (-)	0.7411 ± 0.0090 (-)	0.7399 ± 0.0082 (-)	0.7391 ± 0.0083	0.7405 ± 0.0085 (-)
300	10	0.7252 ± 0.0112 (-)	0.7245 ± 0.0084 (-)	0.7239 ± 0.0088 (-)	0.7230 ± 0.0100	0.7258 ± 0.0103 (-)
500	1	0.7059 ± 0.0081 (+)	0.7056 ± 0.0082 (+)	0.7142 ± 0.0076 (s-)	0.7060 ± 0.0078	0.7054 ± 0.0081 (+)
500	2	0.7252 ± 0.0071 (-)	0.7252 ± 0.0080 (-)	0.7342 ± 0.0071 (s-)	0.7240 ± 0.0080	0.7252 ± 0.0073 (-)
500	3	0.7316 ± 0.0068 (+)	0.7320 ± 0.0074 (+)	0.7329 ± 0.0056 (-)	0.7323 ± 0.0065	0.7332 ± 0.0070 (-)
500	5	0.7352 ± 0.0094 (+)	0.7366 ± 0.0061 (-)	0.7362 ± 0.0073 (+)	0.7358 ± 0.0075	0.7341 ± 0.0077 (+)
500	10	0.7211 ± 0.0073 (-)	0.7230 ± 0.0070 (-)	0.7200 ± 0.0079 (+)	0.7204 ± 0.0077	0.7220 ± 0.0078 (-)

Tabela 7 – Tabela com porcentagem de soluções geradas melhores que os pais para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.0855 $\pm$ 0.0068 (-)	0.0854 $\pm$ 0.0072 (-)	0.3842 $\pm$ 0.0163 (s-)	0.0821 $\pm$ 0.0209	0.0347 $\pm$ 0.0081 (s+)
100	2	0.0739 $\pm$ 0.0069 (s-)	0.0802 $\pm$ 0.0067 (s-)	0.3454 $\pm$ 0.0180 (s-)	0.0585 $\pm$ 0.0154	0.0350 $\pm$ 0.0096 (s+)
100	3	0.0616 $\pm$ 0.0058 (s-)	0.0754 $\pm$ 0.0066 (s-)	0.3016 $\pm$ 0.0165 (s-)	0.0486 $\pm$ 0.0130	0.0354 $\pm$ 0.0094 (s+)
100	5	0.0491 $\pm$ 0.0069 (s-)	0.0708 $\pm$ 0.0071 (s-)	0.1409 $\pm$ 0.0179 (s-)	0.0355 $\pm$ 0.0083	0.0396 $\pm$ 0.0099 (-)
100	10	0.0346 $\pm$ 0.0043 (s-)	0.0607 $\pm$ 0.0051 (s-)	0.0423 $\pm$ 0.0115 (s-)	0.0244 $\pm$ 0.0061	0.0203 $\pm$ 0.0088 (s+)
300	1	0.1175 $\pm$ 0.0085 (s+)	0.0981 $\pm$ 0.0067 (s+)	0.3979 $\pm$ 0.0144 (s-)	0.1400 $\pm$ 0.0175	0.0372 $\pm$ 0.0060 (s+)
300	2	0.1045 $\pm$ 0.0078 (s+)	0.0917 $\pm$ 0.0074 (s+)	0.3559 $\pm$ 0.0182 (s-)	0.1124 $\pm$ 0.0200	0.0518 $\pm$ 0.0101 (s+)
300	3	0.0926 $\pm$ 0.0072 (s+)	0.0887 $\pm$ 0.0064 (s+)	0.3217 $\pm$ 0.0167 (s-)	0.0998 $\pm$ 0.0116	0.0701 $\pm$ 0.0109 (s+)
300	5	0.0802 $\pm$ 0.0072 (s+)	0.0808 $\pm$ 0.0063 (s+)	0.2881 $\pm$ 0.0179 (s-)	0.0874 $\pm$ 0.0133	0.0955 $\pm$ 0.0153 (s-)
300	10	0.0574 $\pm$ 0.0062 (s+)	0.0710 $\pm$ 0.0059 (s-)	0.1260 $\pm$ 0.0191 (s-)	0.0624 $\pm$ 0.0131	0.0783 $\pm$ 0.0155 (s-)
500	1	0.1320 $\pm$ 0.0074 (s+)	0.1030 $\pm$ 0.0056 (s+)	0.4048 $\pm$ 0.0154 (s-)	0.1596 $\pm$ 0.0169	0.0326 $\pm$ 0.0054 (s+)
500	2	0.1178 $\pm$ 0.0065 (s+)	0.0963 $\pm$ 0.0054 (s+)	0.3779 $\pm$ 0.0183 (s-)	0.1288 $\pm$ 0.0170	0.0458 $\pm$ 0.0059 (s+)
500	3	0.1108 $\pm$ 0.0066 (s+)	0.0924 $\pm$ 0.0045 (s+)	0.3359 $\pm$ 0.0173 (s-)	0.1220 $\pm$ 0.0144	0.0692 $\pm$ 0.0071 (s+)
500	5	0.0946 $\pm$ 0.0071 (s+)	0.0860 $\pm$ 0.0065 (s+)	0.3112 $\pm$ 0.0195 (s-)	0.1049 $\pm$ 0.0111	0.1129 $\pm$ 0.0143 (s-)
500	10	0.0738 $\pm$ 0.0075 (s+)	0.0768 $\pm$ 0.0057 (s+)	0.1722 $\pm$ 0.0180 (s-)	0.0806 $\pm$ 0.0108	0.1187 $\pm$ 0.0181 (s-)

Tabela 8 – Tabela com porcentagem de soluções geradas melhores que os pais para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.0894 $\pm$ 0.0082 (s+)	0.0767 $\pm$ 0.0070 (s+)	0.3815 $\pm$ 0.0154 (s-)	0.0974 $\pm$ 0.0201	0.0325 $\pm$ 0.0082 (s+)
100	2	0.0758 $\pm$ 0.0072 (s+)	0.0661 $\pm$ 0.0064 (s+)	0.2901 $\pm$ 0.0191 (s-)	0.0667 $\pm$ 0.0155	0.0337 $\pm$ 0.0111 (s+)
100	3	0.0662 $\pm$ 0.0068 (s+)	0.0561 $\pm$ 0.0051 (s+)	0.1507 $\pm$ 0.0209 (s-)	0.0572 $\pm$ 0.0116	0.0373 $\pm$ 0.0098 (s+)
100	5	0.0515 $\pm$ 0.0062 (s+)	0.0471 $\pm$ 0.0052 (s+)	0.0713 $\pm$ 0.0155 (s-)	0.0394 $\pm$ 0.0119	0.0397 $\pm$ 0.0111 (s-)
100	10	0.0346 $\pm$ 0.0047 (s-)	0.0347 $\pm$ 0.0048 (s-)	0.0168 $\pm$ 0.0082 (s+)	0.0264 $\pm$ 0.0066	0.0184 $\pm$ 0.0061 (s+)
300	1	0.1186 $\pm$ 0.0080 (s+)	0.0938 $\pm$ 0.0056 (s+)	0.3963 $\pm$ 0.0142 (s-)	0.1827 $\pm$ 0.0239	0.0355 $\pm$ 0.0064 (s+)
300	2	0.1060 $\pm$ 0.0065 (s+)	0.0850 $\pm$ 0.0049 (s+)	0.3671 $\pm$ 0.0200 (s-)	0.1339 $\pm$ 0.0163	0.0489 $\pm$ 0.0094 (s+)
300	3	0.0941 $\pm$ 0.0074 (s+)	0.0761 $\pm$ 0.0063 (s+)	0.2841 $\pm$ 0.0199 (s-)	0.1076 $\pm$ 0.0166	0.0652 $\pm$ 0.0097 (s+)
300	5	0.0807 $\pm$ 0.0074 (+)	0.0631 $\pm$ 0.0055 (s+)	0.1776 $\pm$ 0.0171 (s-)	0.0827 $\pm$ 0.0106	0.0940 $\pm$ 0.0197 (s-)
300	10	0.0558 $\pm$ 0.0065 (s+)	0.0476 $\pm$ 0.0052 (s+)	0.0989 $\pm$ 0.0162 (s-)	0.0639 $\pm$ 0.0136	0.0692 $\pm$ 0.0107 (-)
500	1	0.1366 $\pm$ 0.0067 (s+)	0.1017 $\pm$ 0.0049 (s+)	0.3956 $\pm$ 0.0181 (s-)	0.2114 $\pm$ 0.0202	0.0243 $\pm$ 0.0046 (s+)
500	2	0.1218 $\pm$ 0.0072 (s+)	0.0909 $\pm$ 0.0050 (s+)	0.3859 $\pm$ 0.0161 (s-)	0.1567 $\pm$ 0.0160	0.0203 $\pm$ 0.0049 (s+)
500	3	0.1082 $\pm$ 0.0071 (s+)	0.0821 $\pm$ 0.0054 (s+)	0.3427 $\pm$ 0.0175 (s-)	0.1252 $\pm$ 0.0187	0.0231 $\pm$ 0.0050 (s+)
500	5	0.0935 $\pm$ 0.0072 (s+)	0.0709 $\pm$ 0.0045 (s+)	0.2257 $\pm$ 0.0165 (s-)	0.1031 $\pm$ 0.0119	0.0238 $\pm$ 0.0049 (s+)
500	10	0.0714 $\pm$ 0.0057 (s+)	0.0552 $\pm$ 0.0049 (s+)	0.1464 $\pm$ 0.0158 (s-)	0.0751 $\pm$ 0.0095	0.0146 $\pm$ 0.0040 (s+)

Tabela 9 – Tabela com porcentagem de soluções geradas melhores o melhor da execução para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.0186 $\pm$ 0.0038 (s+)	0.0181 $\pm$ 0.0040 (s+)	0.0707 $\pm$ 0.0114 (s-)	0.0318 $\pm$ 0.0109	0.0057 $\pm$ 0.0029 (s+)
100	2	0.0159 $\pm$ 0.0035 (s+)	0.0159 $\pm$ 0.0031 (s+)	0.0779 $\pm$ 0.0108 (s-)	0.0221 $\pm$ 0.0076	0.0096 $\pm$ 0.0052 (s+)
100	3	0.0127 $\pm$ 0.0029 (s+)	0.0135 $\pm$ 0.0027 (s+)	0.0630 $\pm$ 0.0130 (s-)	0.0171 $\pm$ 0.0057	0.0120 $\pm$ 0.0044 (s+)
100	5	0.0090 $\pm$ 0.0029 (s+)	0.0109 $\pm$ 0.0027 (s+)	0.0371 $\pm$ 0.0124 (s-)	0.0112 $\pm$ 0.0044	0.0159 $\pm$ 0.0067 (s-)
100	10	0.0050 $\pm$ 0.0019 (+)	0.0067 $\pm$ 0.0021 (s-)	0.0202 $\pm$ 0.0078 (s-)	0.0051 $\pm$ 0.0029	0.0087 $\pm$ 0.0062 (s-)
300	1	0.0408 $\pm$ 0.0053 (s+)	0.0337 $\pm$ 0.0037 (s+)	0.1022 $\pm$ 0.0117 (s-)	0.0675 $\pm$ 0.0103	0.0084 $\pm$ 0.0029 (s+)
300	2	0.0363 $\pm$ 0.0041 (s+)	0.0303 $\pm$ 0.0035 (s+)	0.1257 $\pm$ 0.0127 (s-)	0.0525 $\pm$ 0.0121	0.0172 $\pm$ 0.0053 (s+)
300	3	0.0314 $\pm$ 0.0043 (s+)	0.0288 $\pm$ 0.0037 (s+)	0.1219 $\pm$ 0.0118 (s-)	0.0444 $\pm$ 0.0080	0.0279 $\pm$ 0.0059 (s+)
300	5	0.0268 $\pm$ 0.0043 (s+)	0.0227 $\pm$ 0.0031 (s+)	0.1002 $\pm$ 0.0136 (s-)	0.0363 $\pm$ 0.0089	0.0442 $\pm$ 0.0106 (s-)
300	10	0.0176 $\pm$ 0.0035 (s+)	0.0157 $\pm$ 0.0035 (s+)	0.0705 $\pm$ 0.0150 (s-)	0.0206 $\pm$ 0.0069	0.0412 $\pm$ 0.0107 (s-)
500	1	0.0514 $\pm$ 0.0043 (s+)	0.0406 $\pm$ 0.0029 (s+)	0.1228 $\pm$ 0.0140 (s-)	0.0819 $\pm$ 0.0114	0.0073 $\pm$ 0.0023 (s+)
500	2	0.0449 $\pm$ 0.0033 (s+)	0.0365 $\pm$ 0.0028 (s+)	0.1613 $\pm$ 0.0124 (s-)	0.0640 $\pm$ 0.0101	0.0152 $\pm$ 0.0034 (s+)
500	3	0.0424 $\pm$ 0.0041 (s+)	0.0337 $\pm$ 0.0029 (s+)	0.1483 $\pm$ 0.0121 (s-)	0.0588 $\pm$ 0.0090	0.0280 $\pm$ 0.0048 (s+)
500	5	0.0358 $\pm$ 0.0037 (s+)	0.0293 $\pm$ 0.0037 (s+)	0.1274 $\pm$ 0.0146 (s-)	0.0460 $\pm$ 0.0065	0.0546 $\pm$ 0.0100 (s-)
500	10	0.0260 $\pm$ 0.0039 (s+)	0.0223 $\pm$ 0.0033 (s+)	0.1002 $\pm$ 0.0129 (s-)	0.0304 $\pm$ 0.0058	0.0665 $\pm$ 0.0130 (s-)

Tabela 10 – Tabela com porcentagem de soluções geradas melhores o melhor da execução para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações.

N	K	Uniform	2-Points	PX	BPX	BPEX
100	1	0.0201 $\pm$ 0.0033 (s+)	0.0169 $\pm$ 0.0042 (s+)	0.0696 $\pm$ 0.0090 (s-)	0.0370 $\pm$ 0.0112	0.0060 $\pm$ 0.0037 (s+)
100	2	0.0164 $\pm$ 0.0038 (s+)	0.0135 $\pm$ 0.0034 (s+)	0.0704 $\pm$ 0.0127 (s-)	0.0254 $\pm$ 0.0087	0.0093 $\pm$ 0.0045 (s+)
100	3	0.0154 $\pm$ 0.0038 (s+)	0.0105 $\pm$ 0.0029 (s+)	0.0584 $\pm$ 0.0133 (s-)	0.0226 $\pm$ 0.0076	0.0123 $\pm$ 0.0055 (s+)
100	5	0.0104 $\pm$ 0.0034 (s+)	0.0084 $\pm$ 0.0022 (s+)	0.0363 $\pm$ 0.0114 (s-)	0.0131 $\pm$ 0.0071	0.0153 $\pm$ 0.0064 (s-)
100	10	0.0051 $\pm$ 0.0016 (+)	0.0048 $\pm$ 0.0018 (+)	0.0078 $\pm$ 0.0054 (-)	0.0058 $\pm$ 0.0031	0.0070 $\pm$ 0.0037 (-)
300	1	0.0415 $\pm$ 0.0046 (s+)	0.0335 $\pm$ 0.0033 (s+)	0.1032 $\pm$ 0.0123 (s-)	0.0895 $\pm$ 0.0147	0.0081 $\pm$ 0.0028 (s+)
300	2	0.0371 $\pm$ 0.0037 (s+)	0.0302 $\pm$ 0.0030 (s+)	0.1489 $\pm$ 0.0181 (s-)	0.0626 $\pm$ 0.0109	0.0163 $\pm$ 0.0055 (s+)
300	3	0.0322 $\pm$ 0.0042 (s+)	0.0264 $\pm$ 0.0030 (s+)	0.1311 $\pm$ 0.0121 (s-)	0.0491 $\pm$ 0.0096	0.0268 $\pm$ 0.0055 (s+)
300	5	0.0276 $\pm$ 0.0034 (s+)	0.0210 $\pm$ 0.0031 (s+)	0.1004 $\pm$ 0.0130 (s-)	0.0341 $\pm$ 0.0076	0.0433 $\pm$ 0.0112 (s+)
300	10	0.0183 $\pm$ 0.0033 (s+)	0.0141 $\pm$ 0.0027 (s+)	0.0531 $\pm$ 0.0107 (s-)	0.0238 $\pm$ 0.0077	0.0353 $\pm$ 0.0079 (s-)
500	1	0.0531 $\pm$ 0.0038 (s+)	0.0407 $\pm$ 0.0030 (s+)	0.1178 $\pm$ 0.0134 (s-)	0.1089 $\pm$ 0.0120	0.0017 $\pm$ 0.0011 (s+)
500	2	0.0469 $\pm$ 0.0043 (s+)	0.0362 $\pm$ 0.0027 (s+)	0.1790 $\pm$ 0.0145 (s-)	0.0779 $\pm$ 0.0115	0.0027 $\pm$ 0.0016 (s+)
500	3	0.0421 $\pm$ 0.0036 (s+)	0.0324 $\pm$ 0.0031 (s+)	0.1685 $\pm$ 0.0135 (s-)	0.0609 $\pm$ 0.0120	0.0046 $\pm$ 0.0020 (s+)
500	5	0.0359 $\pm$ 0.0037 (s+)	0.0270 $\pm$ 0.0024 (s-)	0.1340 $\pm$ 0.0130 (s-)	0.0452 $\pm$ 0.0076	0.0055 $\pm$ 0.0022 (s+)
500	10	0.0267 $\pm$ 0.0034 (s+)	0.0195 $\pm$ 0.0025 (s+)	0.0827 $\pm$ 0.0111 (s-)	0.0289 $\pm$ 0.0058	0.0032 $\pm$ 0.0018 (s+)

Tabela 11 – Tabela com médio de tempo em segundos por execução do operador para 50 execuções do GA com Modelo NK, vizinhança adjacente e condição de parada 20 mil gerações.

N	K	Uniforme	2-Pontos	PX	BPX	BPEX
100	1	4.1720 $\pm$ 0.17	3.4233 $\pm$ 0.13	5.0852 $\pm$ 0.38	8.2683 $\pm$ 0.23	193.6303 $\pm$ 19.61
100	2	4.2830 $\pm$ 0.09	3.5627 $\pm$ 0.06	5.7741 $\pm$ 0.60	8.9337 $\pm$ 0.97	223.8307 $\pm$ 22.53
100	3	4.4033 $\pm$ 0.08	3.7179 $\pm$ 0.05	5.8128 $\pm$ 0.19	8.3246 $\pm$ 0.15	251.6302 $\pm$ 21.13
100	5	4.6892 $\pm$ 0.09	4.0105 $\pm$ 0.07	6.3803 $\pm$ 0.28	8.7437 $\pm$ 0.27	265.7306 $\pm$ 22.53
100	10	5.6769 $\pm$ 0.11	4.9914 $\pm$ 0.08	9.5975 $\pm$ 1.07	11.5676 $\pm$ 1.26	272.2043 $\pm$ 8.02
300	1	11.9576 $\pm$ 0.26	9.0724 $\pm$ 0.25	13.7927 $\pm$ 0.32	36.0741 $\pm$ 0.52	65.472 $\pm$ 1.45
300	2	12.3088 $\pm$ 0.21	10.0890 $\pm$ 0.12	16.3977 $\pm$ 1.25	36.9794 $\pm$ 0.99	70.6542 $\pm$ 0.87
300	3	12.7849 $\pm$ 0.37	10.5670 $\pm$ 0.15	19.2538 $\pm$ 2.22	38.1170 $\pm$ 0.72	83.1350 $\pm$ 0.31
300	5	14.2815 $\pm$ 0.47	11.7881 $\pm$ 0.18	18.6059 $\pm$ 0.52	39.6905 $\pm$ 0.86	96.0912 $\pm$ 0.56
300	10	18.5671 $\pm$ 0.32	15.6444 $\pm$ 0.18	24.9140 $\pm$ 1.04	43.7127 $\pm$ 0.59	138.86 $\pm$ 2.23
500	1	19.7574 $\pm$ 0.40	14.6877 $\pm$ 0.30	24.0915 $\pm$ 0.87	76.1580 $\pm$ 1.50	142.5421 $\pm$ 2.44
500	2	20.6337 $\pm$ 0.42	15.4136 $\pm$ 0.17	28.0403 $\pm$ 0.94	76.1891 $\pm$ 1.41	152.5542 $\pm$ 5.66
500	3	21.3994 $\pm$ 0.40	16.1541 $\pm$ 0.39	31.1686 $\pm$ 1.12	76.6949 $\pm$ 1.27	167.8273 $\pm$ 0.49
500	5	23.9527 $\pm$ 0.48	19.0930 $\pm$ 0.45	34.3170 $\pm$ 1.77	81.2532 $\pm$ 3.66	250.1011 $\pm$ 0.92
500	10	33.8230 $\pm$ 0.60	26.8943 $\pm$ 0.32	49.9365 $\pm$ 3.55	90.9716 $\pm$ 1.76	380.1924 $\pm$ 0.99

Tabela 12 – Tabela com médio de tempo em segundos por execução do operador para 50 execuções do GA com Modelo NK, vizinhança aleatória e condição de parada 20 mil gerações..

N	K	Uniforme	2-Pontos	PX	BPX	BPEX
100	1	4.4230 $\pm$ 0.15	3.4930 $\pm$ 0.07	5.3514 $\pm$ 0.22	8.4723 $\pm$ 0.21	9.8561 $\pm$ 0.14
100	2	4.4928 $\pm$ 0.11	3.6261 $\pm$ 0.09	5.9266 $\pm$ 0.19	8.6113 $\pm$ 0.22	10.6780 $\pm$ 0.21
100	3	4.6454 $\pm$ 0.10	3.7749 $\pm$ 0.07	6.1686 $\pm$ 0.22	8.8054 $\pm$ 0.23	12.0368 $\pm$ 0.15
100	5	4.9463 $\pm$ 0.08	4.0651 $\pm$ 0.07	6.8157 $\pm$ 0.27	9.1457 $\pm$ 0.19	29.8681 $\pm$ 13.58
100	10	6.1938 $\pm$ 0.16	5.2584 $\pm$ 0.14	8.8163 $\pm$ 0.39	10.5045 $\pm$ 0.25	17.0546 $\pm$ 0.10
300	1	12.8736 $\pm$ 0.49	9.2598 $\pm$ 0.13	14.8490 $\pm$ 0.49	34.7063 $\pm$ 0.49	62.6132 $\pm$ 0.97
300	2	13.2838 $\pm$ 0.51	10.6664 $\pm$ 0.88	17.1644 $\pm$ 0.43	35.8829 $\pm$ 0.53	74.1921 $\pm$ 0.87
300	3	13.7167 $\pm$ 0.49	11.4413 $\pm$ 1.20	18.3590 $\pm$ 0.61	40.2079 $\pm$ 3.95	83.1432 $\pm$ 0.12
300	5	14.5246 $\pm$ 0.45	11.3473 $\pm$ 0.15	20.7298 $\pm$ 0.74	38.5172 $\pm$ 0.68	76.0912 $\pm$ 1.12
300	10	19.7169 $\pm$ 0.52	15.7560 $\pm$ 0.17	28.2237 $\pm$ 1.10	43.2471 $\pm$ 0.90	92.1235 $\pm$ 0.76
500	1	22.1069 $\pm$ 0.65	15.4830 $\pm$ 0.47	27.3261 $\pm$ 1.22	74.4617 $\pm$ 1.43	96.8721 $\pm$ 0.32
500	2	23.9801 $\pm$ 0.50	16.3409 $\pm$ 0.64	32.1880 $\pm$ 1.84	76.5578 $\pm$ 1.41	99.1298 $\pm$ 0.98
500	3	24.8052 $\pm$ 0.51	16.6947 $\pm$ 0.43	34.3525 $\pm$ 1.94	77.1608 $\pm$ 1.40	113.4122 $\pm$ 1.41
500	5	29.6501 $\pm$ 2.61	18.8808 $\pm$ 0.58	38.7859 $\pm$ 2.33	81.5831 $\pm$ 1.38	125.1762 $\pm$ 0.76
500	10	42.7098 $\pm$ 3.33	29.0369 $\pm$ 0.64	55.1728 $\pm$ 4.95	91.1578 $\pm$ 2.33	472.189 $\pm$ 11.95

4.3.2 Mochila 0-1

Para o problema da Mochila 0-1, os operadores utilizados foram o BPX, uniforme, 2-pontos, BPX+2PT e BPX+UN, descritos anteriormente. Dessa forma, utilizando os critérios e configurações descritos anteriormente, Os experimentos para o problema em questão foram realizados com e sem LSFI, assim os resultados podem ser observados nas Tabelas 13 - 15. O tamanho do cromossomo (n) e da população todavia, tem valores igual a 100 e 200 respectivamente.

Os parâmetros utilizados para inicialização do problema são:

- pesos e *profit* são reais e não negativos, escolhidos aleatoriamente entre um intervalo definido de 5 a 20 para os pesos e de 40 a 100 para o *profit*.
- capacidade = $(0.5 \times S_w)$, onde S_w corresponde a soma de todos os pesos.

Como observado na Tabela 13, o operador de recombinação uniforme obteve um resultado melhor que os demais, seguido do 2-pontos e BPX, seja para o caso com LSFI e sem. Contudo, é possível constatar também que o operador BPX obteve melhor resultado de *fitness* que os demais para o experimento sem LSFI.

Já a taxa de sucesso na recombinação de cada tipo operador para o caso do problema da mochila, como apresentado na Tabela 14, descreve um cenário diferente do modelo NK exposto na seção anterior, onde o operador BPX obteve uma taxa de sucesso na recombinação maior que os demais operadores tradicionais.

Tabela 13 – Tabela com valor médio de *fitness* para o problema da mochila 0-1 para configuração com busca local primeiro melhoramento e sem. O símbolo *s* indica que o valor encontrado é estatisticamente significativo, seguido de + se o BPX obteve melhor resultado ou – caso contrário.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Com LSFI	4557.15 ± 106.11 (-)	4556.85 ± 105.03 (-)	4555.05 ± 103.50	4556.20 ± 106.70 (-)	4555.25 ± 104.75 (-)
Sem LSFI	4553.26 ± 106.74 (+)	4553.64 ± 105.17 (+)	4553.87 ± 103.51	4553.05 ± 105.23 (+)	4551.69 ± 102.45 (+)

Esse comportamento também se repete na Tabela 15, onde é apresentado a porcentagem de indivíduos gerados por recombinação, melhores que o melhor indivíduo encontrado até o momento. A explicação para o pior resultado para o BPX neste problema é que, por submeter sua função de avaliação a determinadas restrições, no caso em questão a capacidade total da mochila, o operador BPX pode ter sido levado a realizar cortes em indivíduos promissores da população, enquanto que os demais operadores, por possuírem características mais exploradoras sem vieses estruturais, conseguiram um melhor desempenho.

Ressalta-se que a estimação das contribuições das partições (componentes conectados) não se limita pela restrição da capacidade total da mochila. Assim, por selecionar gulosamente os melhores subconjuntos de variáveis (para um ou outro pai) em cada componente conectado, o BPX gera um número maior de indivíduos que não respeitaram as restrições. Isto demonstra uma desvantagem do método BPX atual e a possibilidade de possíveis melhores no futuro para problemas com restrições.

Tabela 14 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema da mochila 0-1 com configuração junto com busca local primeiro melhoramento e sem.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Com LSF _i	0.0795 ± 0.007 (s-)	0.0554 ± 0.004 (s-)	0.0109 ± 0.002	0.0386 ± 0.004 (s-)	0.0324 ± 0.0037 (s-)
Sem LSF _i	0.0353 ± 0.007 (s-)	0.0220 ± 0.004 (s-)	0.0100 ± 0.002	0.0257 ± 0.004 (s-)	0.0202 ± 0.004 (s-)

Tabela 15 – Tabela com valor médio de soluções melhores que o melhor da população para o problema da mochila 0-1 com configuração junto com busca local primeiro melhoramento e sem.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Com LSF _i	0.0031 ± 0.000 (s-)	0.0027 ± 0.000 (s-)	0.0016 ± 0.000	0.0038 ± 0.001 (s-)	0.0029 ± 0.001 (s-)
Sem LSF _i	0.0084 ± 0.002 (s-)	0.0062 ± 0.001 (s-)	0.0022 ± 0.001	0.0067 ± 0.001 (s-)	0.0059 ± 0.002 (s-)

4.3.3 Funções Enganosas

Através da função armadilha descrita na Seção 1.1.2 é possível criar funções enganosas, como a que possui um ótimo global com todos os genes igual a 1 e um ótimo local enganoso com todos os genes igual a 0. Em ambas as funções enganosas com valor de n igual a 10 e 50 baseadas na função armadilha abordada neste trabalho, a dificuldade do GA para encontrar o ótimo local depende das demais variáveis a , b e z . Caso o valor de n seja igual a 10, então os valores de a e z são respectivamente 0.82 e 8. Para n igual a 50, foi adotado a igual a 0.80 e z igual a 48, assim como no experimento de (TINÓS; YANG, 2007). O valor de b todavia, como representa o ótimo global, é sempre igual a 1.0. Dessa forma, é possível acompanhar o resultado dos experimentos nas Tabelas 16 - 21.

Ao analisar qual das execuções com os operadores conseguiu alcançar mais vezes o ótimo global, foi verificado que para $n = 10$, todos alcançaram ao menos 1 vez. O BPX contudo, conseguiu alcançar o ótimo global um total de 3 vezes sobressaindo-se também em valor de *fitness*, como apresentado na Tabela 16, sobre os demais.

Já para $n = 50$, em nenhuma execução foi possível alcançar o ótimo global com a condição de parada definida, o que fica reforçado com os valores iguais de média de *fitness* vistos na Tabela 17. O fato do desempenho dos operadores ser maior com um valor pequeno de n , pode ser explicado pela complexidade do problema no que diz respeito ao número de genes a serem otimizados, deixando-se levar também pela bacia de atração do ótimo local enganoso.

Nas Tabelas 18 e 19 é apresentado a porcentagem de indivíduos gerados por recombinação melhor que seus progenitores, descrito na Seção 4.1. Como é visto, o operador BPX, obteve melhores resultados que os demais operadores utilizados. Já nas Tabelas 20 e 21 são apresentadas a porcentagem de indivíduos gerados por recombinação que são melhores que o melhor. Novamente, o BPX alcançou resultados superiores se comparados com os demais operadores.

Esse resultado é de fato interessante, visto que o problema armadilha abordado é do tipo PB e livre de restrições, o que não acontece no caso da mochila 0-1 por exemplo. Isso reforça então a ideia de que o operador apesar de ser passível de aprimoramento no futuro, o BPX tem potencial para ser aplicado em problemas pseudo-Booleanos que o PX original não pode ser aplicado,, unindo técnicas de EDAs com operadores genéticos de recombinação nos AGs de maneira eficiente.

Tabela 16 – Tabela com valor médio dos melhores *fitness* para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$. O símbolo *s* indica que o valor encontrado é estatisticamente significativo, seguido de + se o BPX obteve melhor resultado ou – caso contrário.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.8236 ± 0.0252 (s+)	0.8272 ± 0.0353 (s+)	0.8308 ± 0.0427	0.8236 ± 0.0252 (s+)	0.8236 ± 0.0252 (s+)

Tabela 17 – Tabela com valor médio dos melhores *fitness* para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00	0.8000 ± 0.00

Tabela 18 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.0148 ± 0.008 (s+)	0.0198 ± 0.04 (s+)	0.0545 ± 0.11	0.0303 ± 0.052 (s+)	0.0311 ± 0.045 (s+)

Tabela 19 – Tabela com valor médio de soluções melhores que os pais por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.0511 ± 0.0068 (s+)	0.0389 ± 0.007 (s+)	0.0643 ± 0.012	0.0618 ± 0.011 (s+)	0.0579 ± 0.009 (s+)

Tabela 20 – Tabela com valor médio de soluções melhores que o melhor por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 10$.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.0020 ± 0.002 (s+)	0.0022 ± 0.005 (s+)	0.0128 ± 0.045	0.0031 ± 0.003 (s+)	0.0030 ± 0.003 (s+)

Tabela 21 – Tabela com valor médio de soluções melhores que o melhor por recombinação para o problema armadilha com configuração sem busca local primeiro melhoramento para $n = 50$.

	Uniforme	2-pontos	BPX	BPX+UN	BPX+2PT
Sem LsFi	0.0133 ± 0.003 (s+)	0.0107 ± 0.003 (s+)	0.0206 ± 0.007	0.0179 ± 0.006 (s+)	0.0186 ± 0.006 (+)

Conclusões

Neste trabalho foi apresentado um novo operador de recombinação para problemas pseudo-Booleanos do tipo caixa-preta baseado nos conceitos do PX e do BOA denominado BPX. Os resultados apresentados na seção anterior, mostram o potencial do novo operador para problemas pseudo-Booleanos, onde também foi apresentado pontos fracos que podem ser trabalhados.

5.1 Contribuições da Pesquisa

A pesquisa por operadores de recombinação que respeitem os blocos de informação básicos das soluções é recente na área de computação evolutiva, sendo estes operadores ainda pouco estudados no Brasil. Uma das áreas que tem despertado grande interesse na computação evolutiva é a área que investiga algoritmos de estimação de distribuição. Nestes algoritmos, é essencial a identificação dos blocos de informação básicos, que são então re combinados muitas vezes.

É importante ressaltar também que a recombinação não é exclusividade de computação evolutiva, sendo utilizada em outras meta-heurísticas populacionais e algoritmos de otimização. Recombinação pode ser usada para recombinar soluções produzidas em diferentes execuções de um algoritmo ou para recombinar soluções encontradas por diferentes algoritmos. Assim, a investigação por métodos de recombinação eficientes é importante também para a área de otimização de maneira geral. Os códigos desenvolvidos para fins de pesquisa durante o projeto, irão estar disponibilizados no Github ¹ como forma de disseminar o novo operador proposto.

¹ <https://github.com/>

5.2 Vantagens e Desvantagens

Como requisito para a aplicação dos operadores PX e BPX, é necessário a utilização do grafo de interação entre as variáveis de decisão do problema como um parâmetro dos mesmos. A principal vantagem BPX em relação ao PX é o fato de poder ser aplicado em problemas do tipo caixa-preta, onde a informação entre variáveis de decisão não é conhecida. Os bons resultados com as funções enganosas mostram este fato. Em relação aos operadores tradicionais, BPX e PX apresentam como vantagem o fato de criarem a máscara de recombinação de acordo com o grafo de interações e com as informações contidas nos pais. Os bons resultados no problemas com o modelo NK mostram o potencial de BPX e PX em relação aos operadores de recombinação tradicionais.

No BPX são utilizados os grafos direcionados das redes Bayesianas como no BOA, com a diferença que a cada geração uma nova rede Bayesiana é amostrada no mesmo e já no BPX isso acontece em intervalos de 300 gerações. A adoção dessa medida pode ter influenciado em três fatores importantes e limitantes do operador:

1. A qualidade da rede Bayesiana gerada - o desempenho de um EDA está diretamente ligado a qualidade do modelo que ele gera e isto por sua vez é relacionado com a diversidade da população, com ao menos uma instância ótima e as métricas utilizadas bem definidas da BN (CROCOMO, 2012). Já a qualidade da BN gerada, influi diretamente no resultado do BPX, uma vez que um modelo com partições não consolidadas pode favorecer o rompimento dos blocos construtivos encontrados. Dessa forma, o uso em intervalos contínuos da mesma BN ao invés da confecção a cada nova geração como no BOA, pode ter prejudicado limitando o resultado do BPX.
2. O tempo de execução e alto custo computacional - a construção de uma nova BN durante um intervalo pré-determinado de gerações certamente dificultou no tempo total de execução do AG com o operador BPX, como apresentado nas Tabelas 11 e 12. Ao passo que nesses intervalos definidos, procedimentos de avaliação e busca de novos modelos da BN são executados repetidamente, distinguindo assim o BPX dos demais operadores de recombinação utilizados, que executam ações menos complexas e apresentam um menor tempo de execução.
3. Aplicação em problemas com restrições - como visto na Seção 1.1.1, o operador BPX não obteve, no geral, resultados satisfatório se comparado aos demais operadores utilizados. Esse resultado todavia é atribuído as restrições da função de avaliação, que para o caso do problema em questão, era o tamanho da mochila.

5.3 Trabalhos Futuros

Este trabalho apresentou um novo operador de recombinação por decomposição para a classe dos problemas pseudo-Booleanos, utilizando uma abordagem caixa-preta, onde faz-se uso do conhecimento adquirido sobre a estrutura do problema para alcançar melhores resultados. Como descrito anteriormente ao longo deste trabalho, uma das operações fundamentais para o uso do operador é a criação da rede Bayesiana, como forma de mapear o relacionamento condicional entre as variáveis de decisão do problema em um grafo direcionado. Os grafos gerados pelo operador todavia, demonstraram-se divergentes dos esperados para o modelo NK adjacente e aleatório.

Dessa forma, apesar dos resultados ainda não favorecerem totalmente o uso do operador BPX e BPEX para os problemas como o da mochila 0-1 e o modelo NK, é esperado que através do aprimoramento por meio de técnicas de detecção de comunidade em redes Bayesianas descritas em (CROCOMO, 2012) por exemplo, seja possível representar mais fielmente a estrutura das variáveis de decisão do problema. Dessa forma, as novas BNs confeccionadas com novas técnicas podem ser superiores às obtidas nestes experimentos e consequentemente os valores de *fitness* obtidos.

Assim, será estudado também, quais as melhores formas de garantir a qualidade da rede Bayesianas em um tempo e custos computacionais razoáveis. Por fim, a aplicação em novos problemas pseudo-Booleanos para examinar o comportamento do operador BPX será analisada, contudo os mesmos ainda não foram definidos.

Referências

AGUIRRE, H.; TANAKA, K. A study on the behavior of genetic algorithms on nk-landscapes: Effects of selection, drift, mutation, and recombination. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, The Institute of Electronics, Information and Communication Engineers, v. 86, n. 9, p. 2270–2279, 2003.

BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. [S.l.]: Oxford university press, 1996.

BEYER, H.-G.; SCHWEFEL, H.-P. Evolution strategies—a comprehensive introduction. *Natural computing*, Springer, v. 1, n. 1, p. 3–52, 2002.

BOROS, E.; HAMMER, P. L. Pseudo-boolean optimization. *Discrete applied mathematics*, Elsevier, v. 123, n. 1, p. 155–225, 2002.

CARR, J. An introduction to genetic algorithms. *Senior Project*, p. 1–40, 2014.

COLIN, R. R.; JONATHAN, E. *Genetic algorithms-Principles and perspectives, A guide to GA Theory*. [S.l.]: Springer US, 2002. v. 20. 332 p.

COOK, W. *In pursuit of the traveling salesman: mathematics at the limits of computation*. [S.l.]: Princeton University Press, 2012.

COOPER, G. F.; HERSKOVITS, E. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, Springer, v. 9, n. 4, p. 309–347, 1992.

CROCOMO, M. K. *Algoritmo de otimização bayesiano com detecção de comunidades*. Tese (Doutorado) — Universidade de São Paulo, 2012.

DEB, K.; GOLDBERG, D. E. Sufficient conditions for deceptive and easy binary functions. *Annals of mathematics and Artificial Intelligence*, Springer, v. 10, n. 4, p. 385–408, 1994.

EÉN, N.; SORENSSON, N. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, v. 2, p. 1–26, 2006.

EIBEN, A.; SMITH, J. *Introduction to evolutionary computing (natural computing series)*. [S.l.]: Springer, 2008.

EIBEN, A. E.; SMITH, J. E. et al. *Introduction to evolutionary computing*. [S.l.]: Springer, 2003. v. 53.

EIBEN, A. E.; SMITH, S. K. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 1, p. 19–31, 2011.

- FOGEL, D. B. Foundations of evolutionary computation. In: . [s.n.], 2006. v. 6228, p. 622801–622801–13. Disponível em: <<http://dx.doi.org/10.1117/12.669679>>.
- GEARD, N. et al. A comparison of neutral landscapes-nk, nkp and nkq. In: IEEE. *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. [S.l.], 2002. v. 1, p. 205–210.
- GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning*, Springer, v. 3, n. 2, p. 95–99, 1988.
- HAINS, D. R.; WHITLEY, L. D.; HOWE, A. E. Revisiting the big valley search space structure in the tsp. *Journal of the Operational Research Society*, Springer, v. 62, n. 2, p. 305–312, 2011.
- HARIK, G. R. Finding multimodal solutions using restricted tournament selection. In: *ICGA*. [S.l.: s.n.], 1995. p. 24–31.
- HARIK, G. R.; LOBO, F. G.; GOLDBERG, D. E. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, IEEE, v. 3, n. 4, p. 287–297, 1999.
- HAUSCHILD, M.; PELIKAN, M. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 3, p. 111–128, 2011.
- HECKENDORN, R. B.; RANA, S.; WHITLEY, D. Test function generators as embedded landscapes. *Foundations of Genetic Algorithms*, Morgan Kaufmann, v. 5, p. 183–198, 1999.
- HECKERMAN, D.; GEIGER, D.; CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, Springer, v. 20, n. 3, p. 197–243, 1995.
- HELGAUN, K. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, Springer, v. 1, n. 2-3, p. 119–163, 2009.
- HOLLAND, J. H. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, 1975.
- HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT press, 1992.
- JONG, K. A. D. *Evolutionary computation: a unified approach*. [S.l.]: MIT press, 2006.
- KAUFFMAN, S. A. *The origins of order: Self-organization and selection in evolution*. [S.l.]: Oxford University Press, USA, 1993.
- LARRAÑAGA, P.; LOZANO, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation*. [S.l.]: Springer Science & Business Media, 2001. v. 2.
- MELANIE, M. *An introduction to genetic algorithms*. [S.l.: s.n.], 1999. v. 3. 62–75 p.
- MITCHELL, M. Genetic algorithms: An overview. *Complexity*, Wiley Online Library, v. 1, n. 1, p. 31–39, 1995.

- MÖBIUS, A. et al. Combinatorial optimization by iterative partial transcription. *Physical Review E*, APS, v. 59, n. 4, p. 4667, 1999.
- MÜHLENBEIN, H.; PAASS, G. From recombination of genes to the estimation of distributions i. binary parameters. In: SPRINGER. *International Conference on Parallel Problem Solving from Nature*. [S.l.], 1996. p. 178–187.
- MURPHY, K. An introduction to graphical models. *Rap. tech.*, p. 1–19, 2001.
- OCHOA, G. et al. Tunnelling crossover networks. In: ACM. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2015. p. 449–456.
- PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. [S.l.]: Courier Corporation, 1982.
- PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. [S.l.]: Morgan Kaufmann, 2014.
- PELIKAN, M.; GOLDBERG, D. E. *Bayesian optimization algorithm: From single level to hierarchy*. Tese (Doutorado) — PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002. Also IlliGAL Report, 2002.
- PELIKAN, M.; GOLDBERG, D. E.; CANTÚ-PAZ, E. Boa: The bayesian optimization algorithm. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. [S.l.], 1999. p. 525–532.
- PELIKAN, M.; GOLDBERG, D. E.; CANTÚ-PAZ, E. Bayesian optimization algorithm, population sizing, and time to convergence. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2000. p. 275–282.
- PELIKAN, M. et al. Robust and scalable black-box optimization, hierarchy, and using spin glasses. *IlliGAL Report*, n. 2003019, 2003.
- PELIKAN, M.; GOLDBERG, D. E.; TSUTSUI, S. Hierarchical bayesian optimization algorithm: toward a new generation of evolutionary algorithms. In: IEEE. *SICE 2003 Annual Conference*. [S.l.], 2003. v. 3, p. 2738–2743.
- PELIKAN, M. et al. Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap. In: ACM. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. [S.l.], 2009. p. 851–858.
- PISINGER, D. The quadratic knapsack problem—a survey. *Discrete applied mathematics*, Elsevier, v. 155, n. 5, p. 623–648, 2007.
- SALKIN, H. M.; KLUYVER, C. A. D. The knapsack problem: a survey. *Naval Research Logistics Quarterly*, Wiley Online Library, v. 22, n. 1, p. 127–144, 1975.
- SANTANA, R. Gray-box optimization and factorized distribution algorithms: where two worlds collide. *arXiv preprint arXiv:1707.03093*, 2017.
- SANTOS, E. B. d. et al. A ordenação das variáveis no processo de otimização de classificadores bayesianos: Uma abordagem evolutiva. Universidade Federal de São Carlos, 2007.

- SCANAGATTA, M. et al. Learning bayesian networks with thousands of variables. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 1864–1872.
- SIVANANDAM, S.; DEEPA, S. *Introduction to genetic algorithms*. [S.l.]: Springer Science & Business Media, 2007.
- SUTTON, A. M.; WHITLEY, L. D.; HOWE, A. E. Computing the moments of k-bounded pseudo-boolean functions over hamming spheres of arbitrary radius in polynomial time. *Theoretical Computer Science*, Elsevier, v. 425, p. 58–74, 2012.
- TINÓS, R.; OCHOA, G.; WHITLEY, D. A new generalized partition crossover for the traveling salesman problem: tunneling between local optima. In: *Submitted to Evolutionary Computation*. [S.l.: s.n.], 2018.
- TINÓS, R.; WHITLEY, D.; CHICANO, F. Partition crossover for pseudo-boolean optimization. In: *ACM. Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. [S.l.], 2015. p. 137–149.
- TINÓS, R.; WHITLEY, D.; OCHOA, G. Generalized asymmetric partition crossover (gapx) for the asymmetric tsp. In: *ACM. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2014. p. 501–508.
- TINÓS, R.; YANG, S. A self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, Springer, v. 8, n. 3, p. 255–286, 2007.
- TINÓS, R. et al. Nk hybrid genetic algorithm for clustering. *IEEE Transactions on Evolutionary Computation*, IEEE, 2018.
- WELCH, J. J.; WAXMAN, D. The nk model and population genetics. *Journal of theoretical biology*, Elsevier, v. 234, n. 3, p. 329–340, 2005.
- WHITLEY, D.; CHEN, W. Constant time steepest descent local search with lookahead for nk-landscapes and max-ksat. In: *ACM. Proceedings of the 14th annual conference on Genetic and evolutionary computation*. [S.l.], 2012. p. 1357–1364.
- WHITLEY, D.; HAINS, D.; HOWE, A. Tunneling between optima: partition crossover for the traveling salesman problem. In: *ACM. Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. [S.l.], 2009. p. 915–922.
- WHITLEY, L. D.; CHICANO, F.; GOLDMAN, B. W. Gray box optimization for mk landscapes (nk landscapes and max-ksat). *Evolutionary computation*, MIT Press, v. 24, n. 3, p. 491–519, 2016.
- ZHAO, J. et al. Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In: *IEEE. Genetic and Evolutionary Computing, 2009. WGECC'09. 3rd International Conference on*. [S.l.], 2009. p. 105–107.