UNIVERSIDADE FEDERAL DO PAMPA

BRUNO ERNESTO TECHERA DA MOTTA

BRANGUS SELECTION: ALGORITMO DE OTIMIZAÇÃO PARA SELEÇÃO DE ACASALAMENTOS DE BOVINOS COM FOCO NA MAXIMIZAÇÃO DOS GANHOS FINANCEIROS

Bagé
2021

BRUNO ERNESTO TECHERA DA MOTTA

BRANGUS SELECTION: ALGORITMO DE OTIMIZAÇÃO PARA SELEÇÃO DE ACASALAMENTOS DE BOVINOS COM FOCO NA MAXIMIZAÇÃO DOS GANHOS FINANCEIROS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada como requisito parcial para a obtenção do título de Mestre em Computação Aplicada.

Orientadora: Ana Paula Lüdtke Ferreira
Coorientador: Marcos Jun-Iti Yokoo

Bagé
2021

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

M921 Motta, Bruno Ernesto Techera da

Brangus Selection: algoritmo de otimização para seleção de acasalamentos de bovinos com foco na maximização dos ganhos financeiros / Bruno Ernesto Techera da Motta.

95 f.: il.

Orientadora: Ana Paula Lüdtke Ferreira
Coorientador: Marcos Jun-Iti Yokoo
Dissertação (Mestrado) - Universidade Federal do Pampa, Programa de Pós-Graduação em Computação Aplicada, 2021.

1. Melhoramento genético. 2. Índice de seleção econômico. 3. Complexidade computacional. 4. Otimização combinatória. 5. Programação inteira. I. Título.

BRUNO ERNESTO TECHERA DA MOTTA

BRANGUS SELECTION: ALGORITMO DE OTIMIZAÇÃO PARA SELEÇÃO DE ACASALAMENTOS DE BOVINOS COM FOCO NA MAXIMIZAÇÃO DOS GANHOS FINANCEIROS

Dissertação apresentada ao Programa de Pós-graduação em Computação Aplicada da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Computação Aplicada.

Dissertação defendida e aprovada em: 22 de outubro de 2021.

Banca examinadora:

_____

Profª Drª Ana Paula Lüdtke Ferreira

Orientadora

Universidade Federal do Pampa

_____

Profª Drª Arione Augusti Boligon

Universidade Federal de Pelotas

Prof. Dr. Marilton Sanchotene Aguiar

Universidade Federal de Pelotas

Prof. Dr. Sandro da Silva Camargo

Universidade Federal do Pampa

Assinado eletronicamente por **ANA PAULA LUDTKE FERREIRA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/12/2021, às 16:43, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **SANDRO DA SILVA CAMARGO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/12/2021, às 17:31, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0683924** e o código CRC **16806D10**.

Dedico este trabalho a Deus, a minha esposa
e aos meus pais.

AGRADECIMENTO

Agradeço primeiramente a Deus que me deu forças e sabedoria para concluir este trabalho; à professora Ana Paula Lüdtke Ferreira pelos ensinamentos e dedicação; ao professor Marcos Jun-Iti Yokoo e todos os professores e colegas do PPGCAP; à minha esposa Cibele que me auxiliou em todos os momentos; aos meus pais Luis e Graciela pelo incentivo; ao IFSul pela confiança; à Unipampa e Embrapa pela oportunidade.

“Posso todas as coisas em Cristo que me fortalece.”
Filipenses 4:13

RESUMO

O crescimento da população mundial, estimada em 9,7 bilhões até o ano 2050, determinará uma maior demanda por alimentos. Isso implicará novos desafios para a agropecuária, no sentido de produzir mais com menos recursos em um cenário onde a disponibilidade de água e de terras será cada vez mais escassa para essas atividades. Atualmente, o Brasil possui um dos maiores rebanhos bovinos do mundo, sendo um dos maiores produtores e exportadores de carne bovina, ao lado dos EUA e da União Europeia. Uma forma de atender a demanda por carne do país e manter a competitividade no mercado internacional é melhorar o potencial genético dos rebanhos para obter melhores resultados de produção. A criação de animais consiste em um conjunto de procedimentos seletivos cujo objetivo é melhorar continuamente as características das próximas gerações. Este trabalho apresenta um algoritmo de otimização para determinar um sistema de acasalamento ideal para bovinos Brangus com base em um índice de seleção. Um índice de seleção expressa, por meio de um único valor, o mérito genético total de cada animal ou, em outras palavras, determina a contribuição do animal para o rebanho considerando um conjunto de características ponderadas como meio para atingir os objetivos de seleção, que são o propósito final do melhoramento genético que trará o retorno financeiro ao sistema produtivo. A busca pela melhor combinação de acasalamentos, dado um índice de seleção, é um problema de otimização. Um dos objetivos desta dissertação é investigar a complexidade do problema de seleção de acasalamento, em busca de um algoritmo ótimo com um tempo de execução polinomial no número de animais envolvidos. Além disso, este trabalho difere por otimizar acasalamentos usando um índice de seleção baseado em valores econômicos. Valores econômicos possibilitam avaliar a importância econômica das características dos animais em um sistema produtivo, expressando em valores monetários o retorno financeiro que resulta da modificação de um característica através do melhoramento genético. A solução proposta baseia-se no uso de programação linear e técnicas *branch-and-bound* como forma combinada de resolver o problema original, modelado como uma instância de programação inteira. Os resultados mostram que o algoritmo fornece a solução ótima para o problema dentro de um limite de tempo polinomial.

Palavras-chave: Melhoramento genético. Índice de seleção econômico. Complexidade computacional. Otimização combinatória. Programação inteira.

ABSTRACT

The growth of the world population, estimated at 9.7 billion by 2050, will determine a greater demand for food, implying new challenges for agriculture and livestock, in the sense of producing more with fewer resources in a scenario where the availability of water and land will be increasingly scarce for these activities. Currently, Brazil has one of the largest bovine herds globally, being one of the leading producers and exporters of beef, together with the USA and the European Union. One way to meet the country's demand for meat and maintain competitiveness in the international market is to improve the genetic potential of the herds to obtain better production results. Animal breeding consists of selective procedures whose objective is to improve the next generations' characteristics continuously. This work presents an optimization algorithm based on a selection index to determine an ideal mating system for Brangus cattle. Through a single value, a selection index expresses the total genetic merit of each animal. In other words, it determines the animal's contribution to the herd considering a specific set of characteristics in a weighted manner that allows the achievement of selection objectives, which are the ultimate purpose of the genetic improvement that will bring the financial return to the production system. The search for the best combination of matings given a selection index is an optimization problem. One of this dissertation's goals is to investigate the complexity of the mating selection problem in search of an optimal algorithm with a polynomial execution time in the number of animals involved. Also, this work differs by optimizing matings using a selection index based on economic values. Economic values make it possible to assess the economic importance of animal characteristics in a productive system, expressing in monetary values – the financial return that results from modifying a feature through genetic improvement. The proposed solution relies on linear programming and *branch-and-bound* techniques as a combined way of solving the original problem, modeled as an integer programming instance. The results show that the algorithm provides the optimal solution to the problem within a polynomial-time limit.

Keywords: Animal breeding, Economic selection index, Computational complexity, Combinatorial optimization, Integer programming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AOL	Área de olho de lombo
DEP	Diferença Esperada na Progênie
DEPG	Diferença Esperada na Progênie aprimorada pela Genômica
EG	Espessura de gordura subcutânea
EGP8	Espessura de gordura subcutânea na garupa
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
ESALQ	Escola Superior de Agricultura Luiz de Queiroz
IBGE	Instituto Brasileiro de Geografia e Estatística
MAPA	Ministério da Agricultura, Pecuária e Abastecimento
TEC	Tonelada Equivalente Carcaça
USDA	*United States Department of Agriculture*
USP	Universidade de São Paulo

SUMÁRIO

1 INTRODUÇÃO

1.1 Processos de melhoramento animal

A bovinocultura de corte brasileira se estabeleceu como uma das principais fontes de alimentos, tanto no cenário nacional como internacional. O Brasil tem um rebanho bovino de aproximadamente 215 milhões de cabeças, o que representa 14,7% do total no mundo[1]. De acordo com dados do USDA (*United States Department of Agriculture*), em 2020 foram produzidas 10,1 milhões de toneladas equivalente carcaça[2] (TEC) em território brasileiro, o que torna o país um dos principais produtores de carne bovina do mundo juntamente com os EUA e União Europeia, que produziram no mesmo ano 12,3 e 6,8 milhões de TEC, respectivamente[3].

Conforme o Ministério da Agricultura, Pecuária e Abastecimento (MAPA), o valor bruto da produção agropecuária brasileira em 2020 foi de R$ 1 trilhão, sendo R$ 666,60 bilhões derivados da Agricultura e R$ 336,22 bilhões da Pecuária. No mesmo período, o Brasil exportou neste setor US$ 100,7 bilhões dos quais a soja e carnes representam 34,99% e 17,04%, respectivamente. De acordo com dados do Centro de Estudos Avançados em Economia Aplicada da ESALQ/USP, o setor agropecuário brasileiro é responsável por empregar aproximadamente 18,5 milhões de pessoas, o que equivale a 8% da população do país ou 17% da população economicamente ativa. Segundo a Associação Brasileira das Indústrias Exportadoras de Carnes[4], em 2020 o país exportou 2,0 milhões toneladas de carne bovina, sendo que 85,66% corresponderam à carne *in natura*. Entre os principais destinos da carne bovina brasileira estão China, Hong Kong, Egito, União Europeia e Chile. Resultados da Pesquisa da Pecuária Municipal do IBGE mostram que o Rio Grande do Sul figura entre as unidades da federação com maior produção de gado bovino em 2020, com 11,12 milhões de cabeças ou aproximadamente 5,5% do rebanho total do Brasil[5].

Conforme os dados apresentados, a agropecuária impacta o desenvolvimento do país tanto pelo volume de produção e recursos movimentados como pela quantidade de pessoas envolvidas.

---

[1] http://www.fao.org/faostat/en/#data/QCL

[2] Tonelada equivalente carcaça é uma medida para padronizar a pesagem de carne bovina. Tem a finalidade de simplificar a transformação do peso dos diferentes tipos de carne bovina produzidos, como carne *in natura* com osso, carne *in natura* sem osso e carne industrializada, em uma unidade em comum.

[3] https://apps.fas.usda.gov/psdonline/app/index.html#/app/downloads

[4] http://abiec.com.br/exportacoes/

[5] https://www.ibge.gov.br/estatisticas/economicas/agricultura-e-pecuaria/

Com o crescimento da população mundial, estimada em 9,7 bilhões até 2050, haverá maior demanda por alimentos, o que implicará em novos desafios para a agricultura e para a pecuária, no sentido de produzir mais com menos recursos em um cenário onde a disponibilidade de água e terra será cada vez mais escassa para estas atividades (FAO, IFAD, UNICEF, WFP and WHO, 2021; FAO, 2009). Uma das maneiras de suprir as demandas por carne no país e se manter competitivos no mercado internacional é melhorar o potencial genético dos rebanhos, de forma a alcançar melhores resultados de produção.

O melhoramento genético animal consiste em um conjunto de processos seletivos cuja finalidade é melhorar continuamente as características da próxima geração de animais produzidos. Por meio do aumento da frequência dos genes desejáveis e, consequentemente, da diminuição dos genes indesejáveis, é possível proporcionar o crescimento da produtividade média da população (ELER, 2017a). Uma das vantagens que o melhoramento genético proporciona ao sistema de produção é a possibilidade de obter ganhos cumulativos e perduráveis na produtividade das gerações posteriores do rebanho, visto que a superioridade genética dos pais pode ser transmitida aos filhos (NIETO; ALENCAR; ROSA, 2013; CARDOSO, 2009). O melhoramento genético bovino está fundamentado em dois princípios: seleção e sistemas de acasalamento (ELER, 2017a; CARDOSO, 2009).

A seleção consiste na escolha dos melhores animais para serem os pais da próxima geração do rebanho (CARDOSO, 2009) e está associada a dois conceitos básicos: genótipo e fenótipo. O termo genótipo é a constituição genética do animal ou o conjunto dos seus genes (CARDOSO, 2009). O fenótipo é a expressão do genótipo mais a influencia do ambiente em que o animal se desenvolve (ELER, 2017a), em outras palavras, é manifestação de alguns genes através de características observáveis ou que podem ser avaliadas. O foco da seleção é no genótipo, uma vez que os animais recebem uma amostra aleatória dos genes dos seus progenitores que perduram por toda a vida (CARDOSO, 2009). De forma geral, os animais que apresentarem superioridade genética ou maior potencial de produção são selecionados para reprodução (ELER, 2017b; CARDOSO, 2009). De qualquer forma, para que os animais possam expressar plenamente seu potencial genético, o produtor precisa garantir condições ambientais apropriadas, como por exemplo, alimentação e sanidade adequadas. Como as atividades de manejo são de natureza transitória, isto é, seus efeitos são temporários, devem ser repetidas constantemente ao longo do tempo (NIETO; ALENCAR; ROSA, 2013).

Sistemas de acasalamento são estratégias para determinar de que forma os animais

selecionados serão acasalados, determinando os pares de touros e fêmeas disponíveis no rebanho que deverão reproduzir (CARDOSO, 2009). Sistemas de acasalamento objetivam direcionar e controlar a reprodução, combinando reprodutor (macho) com matriz (fêmea) de forma que seja gerado um produto (prole) com maiores chances de atender aos objetivos do produtor (ELER, 2017c). O sistema de acasalamento configurado de forma que os melhores machos se reproduzam com as melhores fêmeas e, por consequência, os piores machos com as piores fêmeas, é denominado acasalamento entre semelhantes e objetiva obter produtos extremos, ou seja, animais excepcionais em certas características. Por outro lado, o sistema de acasalamento estruturado de forma que os indivíduos acasalados diferem entre si em desempenho, como por exemplo, um macho excelente em uma determinada característica é acasalado com uma fêmea deficiente nessa mesma característica, é denominado acasalamento compensatório ou corretivo e objetiva obter a homogeneidade da população (ELER, 2017c; CARDOSO, 2009).

Cabe ressaltar que a seleção continuada sem controle da reprodução – ou de acasalamentos ao acaso – e sem introdução de novas linhagens pode ter um efeito negativo no longo prazo, devido à diminuição da variabilidade genética e correspondente aumento da consanguinidade do rebanho (ROSA; MENEZES; EGITO, 2013). A consanguinidade – ou endogamia – é decorrente do acasalamento de indivíduos aparentados (ELER, 2017c). A endogamia, apesar de ser utilizada no início do desenvolvimento das raças, como meio para a fixar o biótipo, pode trazer problemas em características relacionadas à fertilidade e rusticidade[6] dos animais (CARDOSO, 2009), por aumentar a chance de expressão de doenças e problemas de saúde recessivos.

Um objetivo de seleção é uma característica desejável que se quer obter ou melhorar no futuro rebanho. A definição dos objetivos de seleção é o primeiro passo para dar início a um programa de melhoramento genético. A definição dos objetivos, que geralmente são de médio e longo prazo, envolve especificar o que deve ser melhorado no rebanho, a partir de um diagnóstico do sistema de produção e da análise das demandas do mercado (NIETO; ALENCAR; ROSA, 2013). Os objetivos de seleção podem compreender características genéticas, mas também aspectos econômicos. Conforme Simões et al. (2020), em sistemas de produção de bovinos de corte os objetivos de seleção de maior importância econômica usualmente estão relacionados com a habilidade de sobrevivência, a fertilidade e o consumo eficiente de alimentos.

Os critérios de seleção são as características pelas quais os animais são avaliados

[6]Capacidade de sobrevivência diante dos intempéries do meio ambiente.

para que os objetivos de seleção sejam atingidos. Programas de melhoramento genético procuram produzir, para cada animal participante, um valor para cada característica considerada critério de seleção. De forma geral, as características consideradas são de caráter biológico, podendo ser mensuradas qualitativamente ou quantitativamente. Dentre os diversos critérios de seleção, podem ser citados como exemplos: idade do primeiro parto nas fêmeas ou perímetro escrotal nos machos, que são associados a características de reprodução; peso à desmama e peso ao sobreano[7], referentes a valores de produção; área de olho de lombo e espessura da gordura subcutânea, relacionadas às características de produto final; altura e conformação, concernentes às características de biótipo (NIETO; ALENCAR; ROSA, 2013).

As características que constituem os objetivos de seleção são um fim ou objetivo final do melhoramento genético que trará o retorno financeiro ao sistema produtivo, as características escolhidas como critérios de seleção são os meios que permitem concretizar tais objetivos (ABREU; SONOHATA; LOPES, 2013; SIMÕES et al., 2020).

Critérios de seleção podem ser combinados, de forma ponderada, gerando um índice de seleção que expressa, por meio de um único valor, o mérito genético total de cada animal (SIMÕES et al., 2020). Um índice de seleção permite classificar os animais de uma população (ELER, 2017b), por meio de uma média dos critérios de seleção definidos, ponderados pela sua importância frente aos objetivos de seleção. Atualmente, os índices de seleção são utilizados pelos diversos programas de melhoramento genético, visto que facilitam a decisão de seleção dos reprodutores (CARDOSO, 2009).

Conforme relatado por Simões et al. (2020), a maior parte dos programas de melhoramento genético utiliza índices de seleção ponderados de forma empírica, pelo qual esta abordagem pode resultar em um ganho genético do rebanho, mas não garante a maximização dos ganhos financeiros (lucros) no sistema de produção. Ao utilizar um índice de seleção econômico, o processo de melhoramento genético dos animais será focado nos ganhos financeiros. A elaboração de um índice de seleção econômico depende da obtenção dos valores econômicos para as características de interesse. Valores econômicos permitem avaliar a importância econômica das características biológicas dos animais em um sistema produtivo específico e eleger quais conformarão os objetivos de seleção (FORMIGONI, 2002). Valor econômico é definido como o retorno financeiro esperado, em termos monetários, resultante da modificação de uma característica específica através da prática do melhoramento genético (SIMÕES et al., 2020). O modelo

[7] Peso aos 18 meses

bioeconômico é sugerido por diversos autores (BOURDON, 1998; GIBSON; WILTON, 1998; WOLFOVÁ et al., 2005) como metodologia que possibilita a estimação de valores econômicos. Um modelo bioeconômico descreve detalhadamente aspectos econômicos e produtivos, de forma conjunta, relacionando informações sobre receitas, custos, dados biológicos e de manejo, representando precisamente o sistema produtivo (FORMIGONI, 2002).

1.2 Objetivos

Este trabalho tem como objetivo construir um algoritmo de otimização com a finalidade de prover um processo de seleção para um sistema de acasalamento direcionado a bovinos da raça Brangus, utilizando como base as características dos animais que maximizam os valores econômicos relacionados à produção pecuária.

O índice de seleção econômico que será utilizado neste trabalho foi elaborado em um trabalho conjunto da Embrapa Pecuária Sul e da Associação Brasileira de Brangus. Os objetivos de seleção que fundamentam este índice foram identificados em (SIMÕES et al., 2020), por meio do mapeamento entre a obtenção de valores econômicos de produção e as correspondentes características de interesse do rebanho, focados em sistemas de produção de animais da raça Brangus. Adicionalmente, o objetivo de seleção temperamento foi integrado na composição do índice de seleção.

São, ainda, objetivos específicos deste trabalho:

1. dar apoio à caracterização da classe de complexidade do problema de otimização de acasalamentos, por meio de heurísticas de organização dos dados e análise do tempo de execução dos algoritmos resultantes;
2. construir um algoritmo de otimização da seleção de acasalamentos, parametrizável por um índice de seleção genérico;
3. fazer uso do índice de seleção econômico desenvolvido pela EMBRAPA Pecuária Sul, com base nos valores armazenados no banco de dados do programa de melhoramento genético da raça Brangus.

A busca pela melhor combinação de acasalamentos dado um índice de seleção econômico pode ser visto como um *problema de otimização*. De forma geral, problemas de otimização consistem na busca dos melhores valores possíveis para um conjunto de variáveis de forma a atingir certos objetivos propostos, minimizando ou maximizando

uma determinada função, chamada *função objetivo* ou *função de custo*, que pode estar ou não sujeita a uma série de restrições (BLUM; ROLI, 2003). Problemas de otimização podem ser categorizados em três tipos: *otimização contínua*, quando as variáveis envolvidas no problema possuem valores reais; *otimização combinatória*, quando as variáveis somente podem assumir valores inteiros ou enumerados; e *otimização mista*, quando algumas das variáveis do problema são discretas e outras são contínuas[8] (PAPADIMITRIOU; STEIGLITZ, 1998; BLUM; ROLI, 2003; BECCENERI, 2008). A saída do problema do algoritmo BRANGUSSELECTION, que foi construído no desenvolvimento deste trabalho, é um conjunto de pares de indivíduos machos e e indivíduos fêmeas, caracterizando o problema como de otimização combinatória.

É esperado que o algoritmo de otimização BRANGUSSELECTION contribua diretamente com os produtores, não somente melhorando a genética dos animais, mas especificamente mediante a seleção de acasalamentos de bovinos que proporcione a maximização dos lucros, em termos monetários, provenientes da progênie do seu rebanho. Consequentemente, é esperado que este trabalho colabore com a segurança alimentar da população mundial, uma vez que contribui com a produção eficiente de alimentos ao permitir que os produtores brasileiros supram as demandas por carne no país e ao mesmo tempo permaneçam competitivos no mercado internacional.

O aprofundamento sobre as técnicas pertinentes para a solução de problemas de otimização combinatória é realizado no Capítulo 2 e a apresentação da estrutura conceitual e o fluxo de informação do algoritmo BRANGUSSELECTION é feita no Capítulo 3.

1.3 Estrutura e organização do texto

Esta seção apresenta a estrutura do texto desta dissertação: o Capítulo 1 está organizado em três seções, a saber: a Seção 1.1 traz a caracterização do problema que será abordado neste trabalho. Em especial, as definições relevantes sobre seleção de indivíduos, sistemas de acasalamento, objetivos de seleção, critérios de seleção e índices de seleção; a Seção 1.2 traz a descrição do objetivo geral e dos objetivos específicos do trabalho; esta Seção apresenta a organização do texto, sumarizando o que é tratado em cada parte do trabalho.

[8] Uma variável é dita discreta se somente admite valores inteiros e contínua quando admite qualquer valor real.

O Capítulo 2 está organizado em duas seções: a Seção 2.1 caracteriza formalmente o que é um problema de otimização, exemplifica o que é ótimo local e ótimo global, descreve o significado de correção e eficiência de algoritmos, diferencia algoritmos exatos e aproximados (heurísticas) e descreve algumas das principais classes de complexidade de problemas; a Seção 2.2 descreve os trabalhos relacionados, que buscam resolver o mesmo problema ou um problema similar ao desta dissertação.

O Capítulo 3 caracteriza o trabalho realizado e traz a descrição do material, métodos e ferramental tecnológico utilizados para a construção da solução algorítmica para o problema que deve ser resolvido. O capítulo está organizado em quatro seções: a Seção 3.1 caracteriza a pesquisa e lista a suas fases; a Seção 3.2 descreve o protocolo utilizado para a revisão sistemática da literatura; a Seção 3.3 apresenta a estrutura da solução do problema que deverá ser resolvido no âmbito do trabalho de dissertação; a Seção 3.4 informa o ferramental tecnológico utilizado para o desenvolvimento; a Seção 3.5 apresenta a fonte dos dados usados nesta dissertação e descreve a base de dados e forma como foi importada para ambiente de programação e utilizadas no sistema, apresentando ainda a análise descritiva das variáveis quantitativas da base de dados.

O Capítulo 4 descreve o desenvolvimento do algoritmo BRANGUSSELECTION. Esse capitulo está estruturado em três seções como segue: a Seção 4.1 apresenta formalmente o problema de maximização de seleção de acasalamentos e demais definições necessárias para o entendimento correto do problema e sua solução; a Seção 4.2 traz a descrição do modelo na forma de um problema de programação linear/inteira e apresenta um exemplo simples para ilustrar a formalização do problema; a Seção 4.3 mostra os algoritmos desenvolvidos neste trabalho, explicando a codificação do modelo para a linguagem de programação R, descrevendo as principais funções criadas, com suas entradas e saídas.

O Capítulo 5 apresenta e discute os resultados desta dissertação. O capitulo está organizado em duas seções. a Seção 5.1 mostra os resultados em relação a eficiência do algoritmo de otimização desenvolvido, apresentando como a função do tempo cresce na medida que o tamanho das entradas aumenta e descreve o método utilizado para mensurar os tempos de execução de cada instância do problema; a Seção 5.2 apresenta a solução proporcionada pelo BRANGUSSELECTION por meio da apresentação de cinco exemplos, analisando seus resultados.

O Capítulo 6 traz a conclusão da dissertação e descreve as expectativas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Problemas de otimização

A otimização consiste em um processo de busca e seleção da melhor solução possível dentro de um conjunto de alternativas viáveis para resolver um problema (BAGHEL; AGRAWAL; SILAKARI, 2012). Muitos dos problemas de otimização que possuem relevância teórica e prática são de natureza combinatória (BLUM; ROLI, 2003). A resolução eficiente de problemas de otimização combinatória é uma área de interesse, visto que os problemas dessa classe estão presentes em várias áreas importantes, do ponto de vista social e econômico: Engenharia, Agricultura, Produção, Economia, entre outros (BAGHEL; AGRAWAL; SILAKARI, 2012). Entre os exemplos de problemas de otimização estão problemas de roteamento, projetos de redes de comunicação, projetos de circuitos eletrônicos, linhas de montagem em fábricas, robótica, distribuição de contêineres em portos, construção de aeronaves, alocação de tarefas a funcionários, entre outros (GOLDBARG; LUNA, 2005).

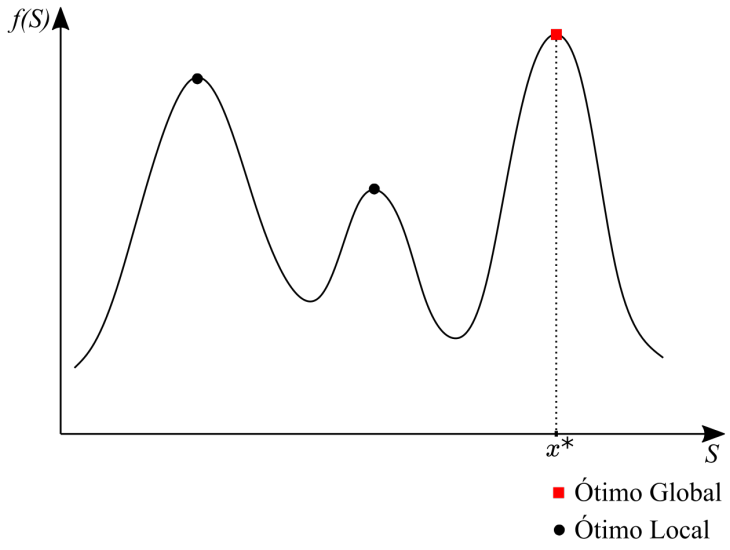
A quantidade de problemas de otimização combinatória é tratada dentro da teoria por meio da definição de problemas teóricos, com definição mais simples e, por essa razão, mais fáceis de tratar matematicamente (GAREY; JOHNSON, 1979). Exemplos clássicos de problemas combinatórios são o problema do *Caixeiro Viajante* e o problema da *Mochila* (CORMEN et al., 2002). A formulação de ambos os problemas é simples: no primeiro caso, o problema se resume em encontrar um roteiro de viagem composto por $n \geq 1$ cidades, onde o trajeto entre duas cidades quaisquer tem um custo associado, de modo que o percurso seja o de menor custo possível, considerando que só é possível visitar cada cidade uma única vez com exceção da cidade de partida, à qual se deve retornar; no segundo caso, o problema consiste em colocar dentro de uma mochila, com capacidade k, os itens disponíveis – com peso e valor associado – que maximizem o valor da soma dos valores dos itens lá colocados, sem exceder a capacidade da mochila. Apesar da formulação ser simples, não se conhece algoritmo eficiente para resolver nenhum dos dois problemas mencionados (CUNHA; BONASSER; ABRAHÃO, 2002; CORMEN et al., 2002). O estudo de um problema real, realizado a partir de problemas teóricos, é feito por meio de um processo chamado *redução* (GAREY; JOHNSON, 1979; ARORA; BARAK, 2009; CORMEN et al., 2002).

Formalmente, um problema de otimização é definido como um conjunto S,

denominado *domínio* ou *espaço de busca* e de uma *função objetivo* ou *função ganho/custo* $f : S \rightarrow \mathbb{R}^+$ (ganho é quando o problema visa maximizar o valor da função objetivo, custo quando se pretende minimizar o valor). A solução de uma instância[1] de um problema de otimização $P = (S, f)$ consiste em encontrar $x^* \in S$ tal que $f(x^*) \geq f(x)$, para qualquer outro $x \in S$, no caso de se pretender maximizar f, ou tal que $f(x^*) \leq f(x)$, para todo $x \in S$, no caso de se pretender minimizar f (PAPADIMITRIOU; STEIGLITZ, 1998).

O conjunto S engloba todas as soluções possíveis para o problema P. O elemento x representa uma solução factível. O elemento x^* é denominado de *ótimo global* ou de solução ótima (PAPADIMITRIOU; STEIGLITZ, 1998). O ótimo global é o valor máximo ou mínimo possível da função objetivo f, respeitando as restrições caso elas existam. O espaço de busca pode ter várias soluções próximas do ótimo global, chamadas de *ótimos locais*. Em um ótimo local qualquer movimento piora o valor da função objetivo, sendo assim, um ótimo local é a melhor solução dentro de um conjunto de soluções vizinhas, porém não é a solução ótima do problema. A Figura 1 ilustra os conceitos de ótimo global e ótimo local.

Figura 1 – Ótimo global e ótimo local



Fonte: Autor, com base em (PAPADIMITRIOU; STEIGLITZ, 1998)

De acordo com a Teoria da Computabilidade, problemas podem ser divididos em computáveis (ou decidíveis) e não computáveis (ou indecidíveis) (SIPSER, 2007).

[1] Uma instância do problema é um caso particular do problema, com uma configuração de valores específicos nos seus parâmetros.

Os problemas computáveis são os que estão no âmbito da solubilidade por meio de algoritmos. Formalmente, um problema computável é aquele que pode ser solucionado por uma Máquina de Turing (ARORA; BARAK, 2009), que é a definição formal de algoritmo (TOSCANI; VELOSO, 2012). Do ponto de vista prático, considera-se um algoritmo todo procedimento sistemático, bem definido, criado com o propósito de solucionar um problema específico. Um algoritmo se caracteriza por receber um valor ou conjunto de valores como *entrada* e gerar um valor ou conjunto de valores como *saída*, por meio de uma sequência finita e detalhada de passos (CORMEN et al., 2002). Um algoritmo é dito *correto* se consegue parar com a resposta correta, ou seja, o algoritmo finaliza corretamente quando recebe como entrada qualquer instância do problema (CORMEN et al., 2002).

Algoritmos podem ser classificados em exatos ou aproximados (BLUM; ROLI, 2003). Problemas de otimização, usualmente, admitem algoritmos exatos – no sentido de encontrarem, garantidamente, a melhor solução (ótimo global). A questão que deve ser investigada é se esses algoritmos são eficientes ou não. Supondo que os computadores fossem infinitamente rápidos e não houvesse limite de espaço de memória, qualquer algoritmo correto para resolver um determinado problema poderia ser utilizado (CORMEN et al., 2002). Embora a capacidade de processamento e memória dos computadores tenha evoluído rapidamente nos últimos anos, os recursos computacionais permanecem finitos e limitados. Conforme (AGUIAR, 1998) não é suficiente somente identificar um problema e garantir que exista um algoritmo que o resolva, ou seja, um algoritmo correto. É necessário saber se o problema pode ser resolvido pelo algoritmo com os recursos computacionais disponíveis, isto é, o algoritmo deve proporcionar uma solução de acordo com o tempo e o espaço de memória que estão à disposição. Por exemplo, para alguma determinada instância do problema cuja resolução demore um tempo de execução da ordem de séculos, não é útil nem viável para o usuário.

O tempo de execução necessário para resolução do problema pode ser considerado o fator de complexidade mais importante para determinar se um algoritmo é eficiente ao ponto de oferecer alguma utilidade prática (BROOKSHEAR, 2013). Um algoritmo é dito eficiente se possui um tempo de execução limitado superiormente por uma função polinomial (GAREY; JOHNSON, 1979).

Se a Teoria da Computabilidade procura identificar os problemas algoritmicamente resolvíveis, a disciplina de Complexidade Computacional procura estabelecer uma classificação para esses problemas (AGUIAR, 1998). A complexidade

de um problema é determinada pela complexidade do algoritmo mais eficiente que o resolve (BROOKSHEAR, 2013); em outros termos, a investigação da complexidade inerente dos problemas permite estabelecer limites para a eficiência dos algoritmos (OLIVEIRA et al., 2010).

Os problemas podem ser agrupados em *classes de complexidade*, que estabelecem o grau de dificuldade computacional para que o problema seja resolvido (SIPSER, 2007; TOSCANI; VELOSO, 2012). A classe de complexidade P compreende os problemas que podem ser resolvidos por algum algoritmo determinístico em tempo polinomial (CORMEN et al., 2002; AGUIAR, 1998). Um algoritmo determinístico se caracteriza por produzir os mesmos resultados em cada execução, caso as entradas do algoritmo sejam repetidas (YANG, 2010). Em termos de Máquina de Turing, os problemas da classe P são solúveis em tempo polinomial por uma Máquina de Turing determinística (GAREY; JOHNSON, 1979).

A classe NP contém os problemas que podem ser resolvidos por um algoritmo não determinístico em tempo polinomial (AGUIAR, 1998). Um algoritmo não determinístico, diante das mesmas entradas, pode produzir diferentes saídas em cada instância do problema, devido à presença de um elemento aleatório em seu processamento (YANG, 2010). De forma equivalente, um problema pertence à classe de complexidade NP se existe algoritmo determinístico é capaz de verificar se uma determinada entrada satisfaz o problema em tempo polinomial. Em termos de Máquina de Turing, os problemas da classe NP são solúveis em tempo polinomial por por uma Máquina de Turing não determinística ou verificados em tempo polinomial por uma Máquina de Turing determinística (ARORA; BARAK, 2009) .

Nos problemas da classe P podemos encontrar uma solução e evidentemente verificar a solução, ambos em tempo polinomial. Por outro lado, nos problemas da classe NP, dada uma candidata à solução, podemos verificar ou certificar em tempo polinomial se ela é ou não, de fato, uma solução. Verificar em tempo polinomial não implica em resolver em tempo polinomial, a menos que ambas as classes sejam idênticas[2]. Pelo que a classe NP compreende os problemas que podem ser verificados em tempo polinomial mesmo que não se conheça um algoritmo de tempo polinomial que os resolva (CORMEN et al., 2002). A classe NP-completa está contida na classe NP, sendo estes os problemas mais difíceis em NP. Um problema p é NP-Completo, se p pertencer a NP e se todos os problemas contidos em NP podem ser polinomialmente reduzidos a p (GAREY;

[2]A questão de as classes P e NP são iguais ou não é um dos problemas de Hilbert (https://www.britannica.com/science/Hilberts-23-problems), ainda sem resposta definitiva.

JOHNSON, 1979). Ou seja, se p admite um algoritmo polinomial todos os demais problemas em NP poderão ser resolvidos em tempo polinomial. A classe NP-difícil (ou NP-hard) compreende os problemas com dificuldades não menores aos problemas da classe NP-completa (AGUIAR, 1998). Um problema NP-difícil se diferencia de NP-completo ao não ser verificável em tempo polinomial por uma Máquina de Turing determinística e se caracteriza por seu problema de decisão equivalente pertencer à classe NP-completa (GAREY; JOHNSON, 1979).

Para os problemas que são notoriamente difíceis, não se conhecem algoritmos eficientes para resolvê-los (ARORA; BARAK, 2009). Nos casos em que os algoritmos exatos conhecidos são ineficientes, ou seja, possuem complexidade exponencial ou maior, o uso de algoritmos aproximados ou *heurísticas* é apropriado. A palavra heurística procede da língua grega e significa "encontrar" (GOLDBARG; LUNA, 2005). Heurísticas possibilitam descobrir, em tempo de execução polinomial, soluções para problemas de otimização cujos melhores algoritmos exatos conhecidos têm complexidade exponencial (YANG, 2010). Nesses casos, a natureza do problema impossibilita a análise de todas as possibilidades, fazendo com que não haja garantia de que a solução ótima seja encontrada. Boas heurísticas – ou boas estratégias de busca de soluções – permitem que boas soluções possam ser encontradas para o problema (BLUM; ROLI, 2003). O quão boa é a solução depende tanto da heurística quando da instância do problema a resolver e não é trivial mensurar a qualidade de uma heurística. Dentre as técnicas heurísticas modernas mais utilizadas nesta categoria, também chamadas de metaheurísticas, podem ser citadas a Busca Tabu, *Simulated Annealing*, Algoritmos Genéticos, Algoritmo de Otimização de Colônia de Formigas e GRASP (BLUM; ROLI, 2003; MELIÁN; PÉREZ; VEGA, 2003).

A investigação sobre a classe de complexidade de um problema deve, se possível, ser feita analiticamente, embora esse não seja um processo trivial. A investigação do problema, tanto com base em heurísticas quanto em técnicas de solução de problemas de otimização que garantidamente retornam uma solução ótima é capaz de dar pistas sobre a classe de complexidade à qual o problema pertence. Se a heurística usada retorna uma solução ótima todas as vezes em que o algoritmo é executado (quando a solução ótima é conhecida, naturalmente) pode ser possível provar que ela é de fato, ótima. Se uma técnica conhecida de resolução de problemas de otimização é usada e os algoritmos executam rapidamente ou demoram muito tempo, pode ser também uma pista de que o problema é, respectivamente, tratável ou intratável computacionalmente. Com essa informação em mãos, pode-se tentar provar uma coisa ou outra. Ou seja, construir uma heurística e

prová-la ótima ou construir uma redução que prove que o problema é NP-difícil.

2.2 Trabalhos relacionados

Esta seção apresenta um resumo dos trabalhos relacionados a esta dissertação. Ou seja, os trabalhos que buscam um esquema de acasalamento ótimo frente a critérios previamente estabelecidos.

Fontoura (2019) apresenta uma solução baseada em algoritmos genéticos para melhorar a escolha de acasalamentos bovinos utilizando as *diferenças esperadas de progênie* (DEP). O algoritmo faz uso de um conjunto de variáveis que resulta em uma recomendação de acasalamentos para o produtor. O autor afirma que a procura por uma combinação de acasalamentos que maximize os resultados genéticos levando em conta múltiplas características é um problema intratável computacionalmente, não admitindo solução em tempo polinomial, porém essa afirmativa não é demonstrada ou referenciada no trabalho. As validações foram feitas com simulações, nas quais foram modificados alguns parâmetros como número mínimo e máximo de utilização de touros disponíveis para monta e taxa de consanguinidade. A taxa de consanguinidade foi ajustada por padrão em 3% para todos os touros, sendo modificados para 4% quando o autor objetivava restringir um determinado touro na solução final do algoritmo para comparação com resultados das outras simulações. Entretanto, os resultados não foram comparados com soluções reais fornecidas por produtores ou até mesmo com a solução ótima, que poderia ter sido obtida por um algoritmo exato, ao menos para um conjunto restrito de animais.

Carvalho et al. (2016) descrevem um sistema de informação para melhoramento genético de caprinos e ovinos. O objetivo do trabalho foi maximizar o ganho de peso e minimizar o grau de parentesco do rebanho (endogamia). O sistema, chamado CAPRIOVI, foi desenvolvido em 2011 como um sistema de automação da escrituração zootécnica de rebanhos. Em razão da quantidade de informações que foram armazenadas ao longo do tempo, o sistema foi aprimorado para auxiliar os produtores na seleção de acasalamentos do animais, ao incorporar algoritmos de inteligência computacional que, no caso, foram algoritmos genéticos. O autor argumenta que analisar todas as combinações possíveis requer alto custo computacional, mesmo para rebanhos pequenos como por exemplo de 200 animais, pois geralmente as combinações a serem analisadas são o número de machos mais um elevado ao número de fêmeas; entretanto, não é mostrado a complexidade do problema para fundamentar a escolha da técnica (algoritmos

genéticos). O algoritmo foi testado com dados de um rebanho real de 193 animais pertencente à Universidade Federal do Piauí. Após a aplicação do algoritmo, foram identificados 14 machos e 79 fêmeas aptos para acasalamento, para os quais o sistema gerou as propostas de acasalamento. Essas propostas foram analisadas por especialistas e comparadas com propostas geradas manualmente por alguns dos especialistas, indicando que o algoritmo sugeriu acasalamentos com menor valor de endogamia e maior DEP esperado para a característica peso. O algoritmo desenvolvido apresenta a vantagem de predizer os valores de DEP e endogamia porém contempla somente a característica peso dos animais, não utilizando nenhum índice de seleção.

Neto et al. (2014) otimizou esquemas de acasalamento em ovinos da raça Santa Inês fazendo uso, também, de algoritmos genéticos. O algoritmo foi aplicado a uma base de dados de animais componentes de núcleos de seleção que possuem *pedigree* estruturado e valores genéticos estimados através de DEP-BLUP. A função objetivo do algoritmo penaliza o crescimento da taxa de consanguinidade dos acasalamentos, de forma que para graus maiores de endogamia há maior penalização. Para realizar o cálculo da taxa de consanguinidade foram utilizadas informações de *pedigree* dos animais presentes na base de dados. Em relação às DEP, o trabalho considerou somente a DEP para a característica *peso aos 60 dias*. Para avaliar a técnica empregada na esquematização dos acasalamentos, com a finalidade de verificar o ganho genético da progênie resultante dos acasalamentos sugeridos, foram realizados sete tratamentos, sendo um deles de acasalamentos ao acaso. Nos demais tratamentos foram alterados alguns parâmetros como por exemplo consanguinidade média da geração subsequente e número de machos utilizados nas montas. Ainda, foram definidas restrições, tais como o número máximo de uso de cada animal, que para as fêmeas foi estipulado o valor 1 e para os machos o valor 40, pois conforme o autor, esse número foi estabelecido para respeitar a capacidade biológica natural dos animais. Os resultados mostram que quando é ampliado o valor permitido da consanguinidade entre os pais, maior é o valor da contribuição genética da progênie para peso aos 60 dias. O autor argumenta que a medida que aumenta a quantidade de animais candidatos à seleção, aumentam os custos computacionais de forma exponencial, entretanto, não é discutida a complexidade do problema para fundamentação da técnica escolhida. A seleção foi feita a partir de uma única característica e não através de um índice de seleção que contemple diversas características o que pode levar a um resultado indesejado em longo prazo.

Carvalheiro, Queiroz e Kinghorn (2010) desenvolveram um programa baseado no

algoritmo de Evolução Diferencial com o objetivo de determinar a contribuição genética ótima com o intuito de melhorar a seleção de animais para reprodução. Conforme relatado pelos autores, a contribuição genética ótima foi definida por meio da otimização de uma função objetivo, que é composta pelo mérito genético esperado da futura progênie e pela coascendência média dos animais em reprodução. Para realizar os testes e validações do desempenho do programa desenvolvido foi realizado primeiramente uma simulação, composta por dados de 6 touros e 120 vacas. Posteriormente foram utilizados dados reais de animais da raça Nelore, dos quais foram selecionados 100 machos e 500 fêmeas. O trabalho tem um foco para determinar a contribuição genética ótima dos animais ao longo das gerações com restrição sobre a consanguinidade. Conforme os autores, o algoritmo se mostrou viável para aplicações práticas, entretanto, o trabalho não contempla o esquema de acasalamento, ou seja, a combinação dos animais candidatos a serem pais da próxima geração do rebanho.

Os trabalhos citados possuem principal ênfase em técnicas que utilizam estratégias evolucionárias, como os algoritmos genéticos, para a otimização de seleção de acasalamentos objetivando o melhoramento genético. Entretanto, estes trabalhos não discutem a complexidade do problema de otimização de seleção de acasalamentos como forma de fundamentar a técnica escolhida para resolver o problema, pelo que um dos objetivos desta dissertação é investigar a complexidade do problema de seleção de acasalamentos, na busca de um algoritmo ótimo com tempo de execução polinomial no número de animais envolvidos. Embora em alguns destes trabalhos encontra-se a afirmação de que o problema é computacionalmente intratável, não há referências ou prova da afirmação. Ademais, não foram encontrados trabalhos que otimizassem os acasalamentos bovinos a partir de valores econômicos, ou seja, que utilizem um índice de seleção elaborado por meio de um modelo bioeconômico. Os trabalhos encontrados utilizam, em sua maior parte, valores empíricos para as características de relevância dos animais, focando primeiramente o ganho genético e não o lucro, em termos monetários, como é o caso desta dissertação.

3 MATERIAL E MÉTODOS

3.1 Caracterização e fases da pesquisa

Esta pesquisa é caracterizada como uma pesquisa aplicada, levando em consideração que a saída do algoritmo gera uma solução que pode ser aplicada imediatamente para resolver o problema de seleção de acasalamentos bovinos que maximize os ganhos financeiros do sistema de produção. Uma pesquisa aplicada estuda a aplicabilidade prática do conhecimento científico a um determinado problema (LAKATOS; MARCONI, 2007). No caso deste trabalho, são aplicadas técnicas da computação na área da agropecuária, mais especificamente, no melhoramento genético bovino com foco na maximização dos ganhos financeiros. O trabalho fez uso de procedimentos de revisão sistemática da literatura (KITCHENHAM, 2004), adaptados para o tema do trabalho, para encontrar soluções similares ao problema colocado.

Os testes e verificação do algoritmo BRANGUSSELECTION foram feitos sobre a base de dados real do programa de melhoramento genético da raça Brangus pertencente à EMBRAPA, pelo qual este trabalho é considerado uma pesquisa com potencial experimental. Uma pesquisa experimental se caracteriza por submeter o objeto de estudo à influência de certas variáveis onde o pesquisador exerce intervenção direta (LAKATOS; MARCONI, 2007); ou ainda, pode ser definida como uma pesquisa na qual há intervenção sistemática por parte do pesquisador com a finalidade de observar os fenômenos provocados pelas intervenções (WAZLAWICK, 2009). Embora não tenha havido experimento com o resultado do algoritmo, a ideia é disponibilizar o sistema para uso pelos produtores.

O trabalho foi organizado nas seguintes fases:

1. Revisão bibliográfica sobre técnicas para solucionar problemas de otimização.
2. Revisão bibliográfica sobre métodos e técnicas de acasalamento de bovinos.
3. Caracterização formal do problema de maximização de seleção de acasalamentos.
4. Modelagem do problema em Programação Inteira.
5. Construção do algoritmo de força bruta, para comparação com os resultados providos pelo algoritmo BRANGUSSELECTION.
6. Implementação do algoritmo de otimização nas linguagens de programação R e C++.

7. Testes do desempenho do algoritmo em relação ao seu tempo de execução.
8. Análise dos resultados e comparação com resultados de outros trabalhos.

3.2 Revisão da literatura

Para realizar a revisão da literatura foram seguidas as seguintes etapas: elaboração das questões de pesquisa, escolha das palavras-chave, escolha das fontes de pesquisa, critérios de inclusão e exclusão de trabalhos e leitura e elaboração de resenha dos trabalhos.

As questões de pesquisa escolhidas para guiar o processo de revisão bibliográfica foram as seguintes:

Q1 Quais são as técnicas usadas para determinação da melhor estratégia de escolhas de acasalamentos?

Q2 Existem trabalhos que analisem a complexidade computacional do problema de escolha de acasalamentos?

As palavras-chave escolhidas em relação aos objetivos da revisão da literatura foram: otimização combinatória, complexidade computacional, programação inteira, índice de seleção econômico, melhoramento genético. Os termos utilizados em inglês foram *combinatorial optimization, computational complexity, integer programming, economic selection index, animal breeding*.

As fontes de consulta de trabalhos relacionados utilizadas foram Google Scholar, Scielo, *Science Direct*, Congresso Brasileiro de Agroinformática (SBIAgro), *Computer and Electronics in Agriculture*, *Applied Engineering in Agriculture* e *Journal of Integrative Agriculture*.

No sentido de aceitar ou não os trabalhos encontrados como trabalhos relacionados, foram estabelecidos critérios de inclusão e exclusão. Após leitura dos resumos dos trabalhos, foram estipulados níveis de similaridade em relação ao objetivo deste trabalho, sendo o valor 1 interpretado como de menor similaridade e o valor 5 como de maior similaridade. Os níveis de similaridade e suas respectivas descrições são os seguintes: 1. problema similar, outra técnica; 2. problema similar, mesma técnica; 3. mesmo problema, técnica diferente; 4. mesmo problema, técnica similar; 5. mesmo problema, mesma técnica. Os trabalhos marcados como de nível 3 a 5, que tratam do mesmo problema que esta dissertação, foram incluídos como trabalhos relacionados.

Como ferramenta para organizar os trabalhos encontrados nas fontes de pesquisa foi utilizada uma planilha eletrônica onde foram anotados a fonte de pesquisa, as palavras-chave utilizadas, o número total de resultados, número de títulos relacionados, número de resumos relevantes e número de trabalhos relacionados. Para os trabalhos relacionados encontrados, foram anotados o título, ano de publicação, como foi encontrado (palavras-chave utilizadas ou em referência de outro trabalho) e o nível de similaridade. Foram relacionados três trabalhos com nível de similaridade 4, 1 trabalho com nível de similaridade 3, e 12 trabalhos com nível de similaridade 1. Nenhum trabalho atingiu o nível de similaridade 5.

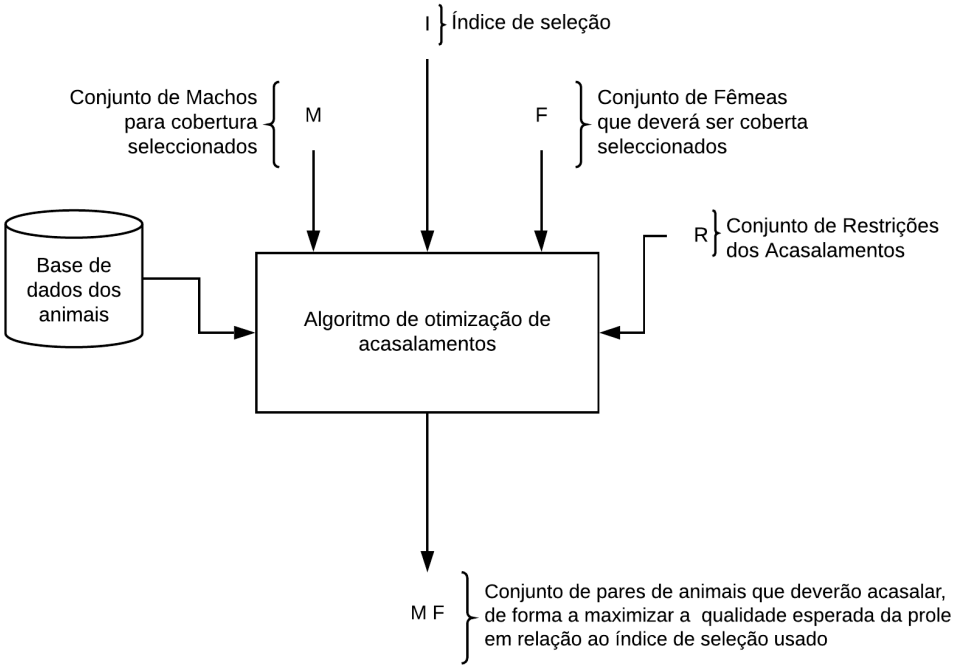
3.3 Modelo do sistema

A Figura 2 apresenta a estrutura conceitual e o fluxo de informação do algoritmo BRANGUSSELECTION. A entrada do sistema são os dados dos animais presentes na base de dados do programa de melhoramento genético e as restrições do modelo: (i) o índice de seleção que deverá ser usado, na forma das variáveis usadas e suas respectivas ponderações; (ii) o conjunto de fêmeas que deverá ser coberta; (iii) o conjunto de machos disponíveis para cobertura; (iv) número máximo que cada touro pode ser usado e (v) o percentual máximo de consanguinidade aceitável. A saída do algoritmo é o conjunto de pares de animais que deverão acasalar, de forma a maximizar a qualidade esperada da prole em relação ao índice de seleção usado.

O índice de seleção, como já foi mencionado na Seção 1.1, provém do modelo bioeconômico construído em (SIMÕES et al., 2020), acrescentado do temperamento. Os objetivos de seleção que foram utilizados neste trabalho estão descritos a seguir:

Contagem de carrapatos (TICK) : o carrapato é um ectoparasita que transmite agentes que causam diversas doenças aos animais resultando em perdas econômicas para o produtor, principalmente em regiões com predominância de clima tropical e subtropical, que são favoráveis ao parasitismo – como é o caso do Brasil (SIMÕES et al., 2020). O combate desse parasita feito exclusivamente por carrapaticidas pode se tornar ineficiente devido à capacidade do carrapato de desenvolver resistência aos produtos químicos (SIMÕES et al., 2020). Conforme Landim et al. (2006), o manejo inadequado do carrapaticida contribui com o problema da resistência das populações de carrapatos aos princípios ativos do remédio. Portanto, obter animais

Figura 2 – Esquema de entradas e saída do algoritmo BRANGUSSELECTION



Fonte: Autor (2021)

geneticamente resistentes ao carrapato é economicamente vantajoso.

Contagem de ovos de nematódeos por grama de fezes (OPG) : na pecuária extensiva e semi-intensiva, os animais estão suscetíveis a contrair infecções por larvas de nematódeos gastrointestinais e pulmonares, denominados endoparasitas. Como consequência dos danos ao sistema digestivo causado por estes vermes, os animais podem ter um crescimento tardio e baixa produtividade em decorrência da ineficiência da ingestão e digestão dos alimentos (SIMÕES et al., 2020). Ainda, de acordo com Simões et al. (2020), selecionar animais mais resistentes aos nematódeos gastrointestinais é relevante economicamente.

Peso de carcaça quente (PCQ) : refere-se ao peso da carcaça após todos os procedimentos de abate. Está vinculada ao rendimento de carcaça que pode ser definida como a relação entre o peso do animal vivo (antes do abate) e o peso da carcaça. Características associadas ao rendimento e qualidade de carcaça consideradas como critério de seleção são área de olho de lombo (AOL), espessura de gordura subcutânea (EG) e espessura de gordura subcutânea na garupa (EGP8) (NIETO; ALENCAR; ROSA, 2013).

Peso da vaca adulta (PVA) : característica relacionada ao crescimento do animal. Animais que apresentam um aumento excessivo de peso na fase adulta podem desenvolver problemas na reprodução e produção, ocasionando prejuízos econômicos (NIETO; ALENCAR; ROSA, 2013).

Taxa de prenhez (TP) : é um indicador do desempenho reprodutivo do rebanho. Constituído pelo número de fêmeas prenhes comparado ao número total de fêmeas aptas para reprodução. Taxas de prenhez baixas indicam um desempenho reprodutivo ineficiente e pode prejudicar o retorno financeiro do sistema de produção (SIMÕES et al., 2020).

Temperamento (T) : a resposta comportamental do animal durante o manejo do rebanho é conhecida como temperamento. Conforme Nieto, Alencar e Rosa (2013), bovinos com temperamento mais bravio são menos lucrativos para o sistema de produção devido a uma série de fatores tais como necessidade de maior demanda de mão de obra em decorrência do manejo mais demorado, aumento da incidência de acidentes entre animais e funcionários, maior desgaste das instalações da propriedade (currais, cercas, balanças, entre outros), diminuição da qualidade da carne devido a lesões na carcaça. O método quantitativo mais usual para avaliar esta característica é a velocidade de fuga (SILVA et al., 2013).

A Tabela 1 apresenta os objetivos de seleção, os critérios de seleção correspondentes e a ponderação usada no índice de seleção utilizado neste trabalho. Note-se que o algoritmo foi construído para aceitar um índice genérico: a alteração dos ponderadores e das variáveis do índice não altera a estrutura e o funcionamento do algoritmo BRANGUSSELECTION.

Tabela 1 – Índice de seleção *default* do algoritmo BRANGUSSELECTION

Objetivo de seleção	**Critério de seleção**	**Ponderador**
TICK	Carrapato	116.0879975
OPG	OPG	-9.1900268
PCQ	AOL	0.1866388
	EG	131.3764082
	EGP8	65.8745087
PVA	Peso adulto da vaca	-0.5495934
TP	Peso ao nascer	6.0030468
T	Temperamento	23.8195584

3.4 Ferramental tecnológico

O ambiente computacional onde foi desenvolvido o algoritmo, bem como os testes e validações, foi um computador com processador Intel Core i7-6500U de 2.50GHz; 12GB de memória RAM; Armazenamento SSD de 500GB. O sistema operacional usado foi a distribuição Linux Ubuntu 20.04 LTS 64-bit.

Para a implementação do algoritmo foi utilizada a linguagem de programação R (IHAKA; GENTLEMAN, 1996), versão 4.0.5. A linguagem R é *open source*, multiplataforma, adequada para ser utilizada em disciplinas como ciência de dados, inteligência artificial, aprendizado de máquina, estatística, entre outras. Uma das principais características da linguagem R é a vetorização, que possibilita utilizar funções em estruturas de dados como vetores ou matrizes, dispensando o uso de laços de repetição (MAZZONETTO; HÖLBIG, 2014). A razão da escolha da linguagem R se fundamenta em que o BRANGUSSELECTION será disponibilizado para a EMBRAPA, cuja infraestrutura computacional é compatível com esta linguagem, além de ser amplamente utilizada pela instituição. Como Ambiente de Desenvolvimento Integrado (IDE) para o R foi escolhido o RStudio 1.4 que é *open source* e multiplataforma. Cabe ressaltar que não foi utilizado multiprocessamento para a execução do algoritmo.

A linguagem R conta com uma comunidade grande de usuários, que colaboram no desenvolvimento da linguagem por meio da construção e disponibilização de pacotes que encapsulam funções com os mais variados propósitos. O repositório de pacotes para a linguagem R (<https://cran.r-project.org/web/packages/>) já conta com mais de 175.000 pacotes verificados e disponibilizados, evitando que os algoritmos já implementados precisem ser reprogramados.

Os pacotes utilizados para construção do algoritmo BRANGUSSELECTION foram os seguintes:

lpSolve (BERKELAAR et al., 2020): proporciona uma interface na linguagem R para o *lp_solve*. O *lp_solve* é um *solver* gratuito baseado no método *Simplex*, que resolve problemas de programação linear, e na abordagem *branch-and-bound*. A função (método) principal chama-se *lp* e possui os seguintes argumentos: *direction*:*string* indicando o tipo de problema com os caracteres *max* para maximização ou *min* para minimização; *objective.in*: vetor numérico com os coeficientes da função objetivo; *const.mat*: matriz com os coeficientes das restrições, devendo ser uma linha por restrição e uma coluna por variável; *const.dir*: vetor de caracteres contendo a

direção das restrições, onde cada valor deve indicar um operador de comparação como, por exemplo, "<="(menor ou igual); *const.rhs*: vetor numérico contendo os valores do lado direito das restrições. É possível especificar o tipo de variáveis do problema com o argumento *all.bin* com valor *TRUE* para variáveis binárias, por exemplo.

lpsymphony (KIM, 2020): fornece uma interface na linguagem R para o *SYMPHONY*. O *SYMPHONY* é um *solver* de código aberto, codificado em C++, para programação linear inteira-mista. A função (método) principal deste pacote chama-se *lpsymphony_solve_LP* cujos argumentos são similares à função *lp* do pacote *lpSolve*.

Rcpp (EDDELBUETTEL; BALAMUTA, 2018): é um pacote que permite integrar as linguagens de programação C++ e R. Códigos escritos em C++ podem ser executados dentro do programa R por meio do uso da função *cppFunction*, que recebe como argumento uma *string* que será interpretada como uma função em C++. A diferença de C++ para R é que C++ é uma linguagem compilada e de tipagem estática e R é uma linguagem interpretada e de tipagem dinâmica. Sendo assim, C++ apresenta a vantagem de ser mais rápido na execução de estruturas de repetição já que os tipos das variáveis são declaradas explicitamente e por isso conhecidos em tempo de compilação.

ggplot2 (WICKHAM, 2016): é um pacote para visualização de dados. A principal característica do pacote é que a criação de gráficos é feita de forma "gramática", ou seja, cada componente do gráfico (ou camada do gráfico) deve ser adicionado de forma que o gráfico é formado pela soma dos componentes (ou sobreposição das camadas). As principais funções são a função *ggplot()*, que recebe os dados que serão exibidos e a função *geom_point()* que recebe a especificação da forma geométrica dos dados e também recebe a função *aes()* que determina como serão mapeados os dados nos eixos x e y.

sampling (TILLÉ; MATEI, 2021): permite a criação de amostras. Este pacote é necessário para gerar amostras da base de dados que são utilizados nos experimentos onde é delimitado o número de animais. O método principal chama-se *strata*, o qual possui como argumentos um *dataframe* com a base de dados, um vetor com o tamanho da amostra, e o método de seleção dos elementos, que pode ser *srswor*, sem reposição ou *srswr*, com reposição.

A verificação do algoritmo BRANGUSSELECTION foi feita por meio da comparação de um esquema de acasalamento aleatório com a solução proporcionada pelo algoritmo. O valor esperado na progênie do rebanho do esquema de acasalamento otimizado deverá ser maior que o esquema de acasalamento aleatório. Também foi feita a comparação do resultado do BRANGUSSELECTION com o resultado obtido por um algoritmo de força bruta em um conjunto de dados reduzido.

Há que se ter um cuidado em relação às estratégias de teste e validação, visto que o algoritmo BRANGUSSELECTION implementa um índice de seleção específico. Comparações com outros índices não indicam necessariamente uma estratégia algorítmica pior, mas sim um índice que gera valores inferiores. A combinação das técnicas Simplex e *branch-and-bound*, usadas nesse trabalho, levam a uma solução ótima para o problema. A questão investigada foi o tempo de execução e a estrutura das árvores geradas na aplicação da técnica de *branch-and-bound*, como forma de tentar inferir a complexidade do problema.

3.5 Fontes de dados

A base de dados dos animais utilizada neste trabalho foi providenciada pela EMBRAPA em formato de planilha eletrônica com extensão *.xls* (arquivo nativo do software *Microsoft Excel*™). Antes da importação da base de dados para o ambiente de desenvolvimento integrado *RStudio*, a planilha foi transformada no formato de arquivo *.csv* (*comma-separated-values*). Este tipo de arquivo separa os valores por vírgulas em uma série de linhas, o que facilita a representação de dados tabulares. Para importar a base de dados, já no formato *.csv*, foi utilizada a função *read.csv* do R. Após a importação, a base de dados foi armazenada em uma variável chamada *dbase* do tipo *dataframe*. No R, um *dataframe* é uma estrutura de dados capaz de armazenar dados tabulares de diferentes tipos, como inteiros, decimais, caracteres, entre outros, onde cada linha corresponde a uma observação e cada coluna a uma variável.

A base de dados é constituída por dados referente a 1366 bovinos da raça *Brangus*, sendo 511 machos e 855 fêmeas. Os dados correspondem aos valores da Diferença Esperada na Progênie aprimorada pela Genômica (DEPG) das características dos animais que compõem o índice de seleção econômico conforme apresentado na Tabela 1. A Figura 3 apresenta a estrutura dos dados através de uma amostra base de dados.

A coluna *animal* apresenta a identificação única de cada animal. A coluna

Figura 3 – Extrato da base de dados

	Animal	Sexo	DEP_CARR	DEP_OPG	DEP_TEMP	DEP_PN	DEP_PAV	DEP_AOL	DEP_EG	DEP_EGP8
1	L823	Femea	0.11	-0.24	1.62	-2.33	-6.68	-2.70	0.39	0.49
2	L758	Femea	0.11	0.33	2.08	-1.75	-14.62	-0.44	0.24	0.50
3	L745	Femea	-0.03	0.20	0.83	-0.43	10.03	-3.61	0.44	0.46
4	L703	Femea	-0.03	-0.21	0.38	-0.81	-0.94	0.82	0.45	0.35
5	L259	Femea	-0.15	-0.33	2.25	2.44	-27.14	-1.28	0.10	-0.04
6	665887	Macho	0.15	0.08	1.56	-2.14	-1.36	-1.44	0.17	0.16
7	M181	Macho	0.02	0.12	0.34	0.74	-9.07	-2.40	0.26	0.31
8	K58	Femea	-0.03	-0.30	0.92	-2.37	-24.81	0.63	0.27	0.23
9	M173	Macho	0.11	-0.08	0.96	-2.02	-18.94	-5.44	0.15	0.26
10	L817	Femea	0.03	0.63	1.24	-3.84	-23.73	-0.33	0.23	0.31
11	L743	Femea	0.15	-0.34	1.53	-0.31	-12.14	-2.41	0.10	-0.09
12	L785	Femea	0.07	-0.09	0.67	-4.60	-20.02	-1.05	0.34	0.20
13	K522	Femea	-0.01	0.12	2.00	-0.78	-12.80	-0.22	0.16	-0.04
14	K395	Femea	0.00	-0.02	1.41	-2.19	-23.95	-4.92	0.17	0.15
15	L439	Macho	0.02	-0.08	1.76	-2.95	-23.52	-0.43	0.20	-0.04

Fonte: Autor (2021)

sexo apresenta o sexo de cada animal, identificados como macho ou fêmea. A coluna *DEP_CARR* apresenta os valores das DEPG da contagem total de carrapatos (logaritmo na base 10 do valor total). A coluna *DEP_OPG* apresenta os valores das DEPG da contagem de ovos de nematódeos por grama de fezes em log na base 10. A coluna *DEP_TEMP* apresenta os valores das DEPG do temperamento, que é medida por meio da velocidade de fuga em metros por segundo (m/s). A coluna *DEP_PN* apresenta os valores das DEPG do peso do animal ao nascer em quilogramas (kg). A coluna *DEP_PAV* apresenta os valores das DEPG do peso adulto da vaca em quilogramas (kg). A coluna *DEP_AOL* apresenta os valores das DEPG da área de olho de lombo obtida por ultrassom entre a décima-segunda e a décima-terceira costela medidas e centímetros quadrados (cm^2). A coluna *DEP_EG* apresenta os valores das DEPG da espessura de gordura subcutânea obtida por ultrassom em milímetros (*mm*). A coluna *DEP_EGP8* apresenta os valores das DEPG da espessura de gordura subcutânea na garupa obtida por ultrassom e também mensurada em milímetros (*mm*).

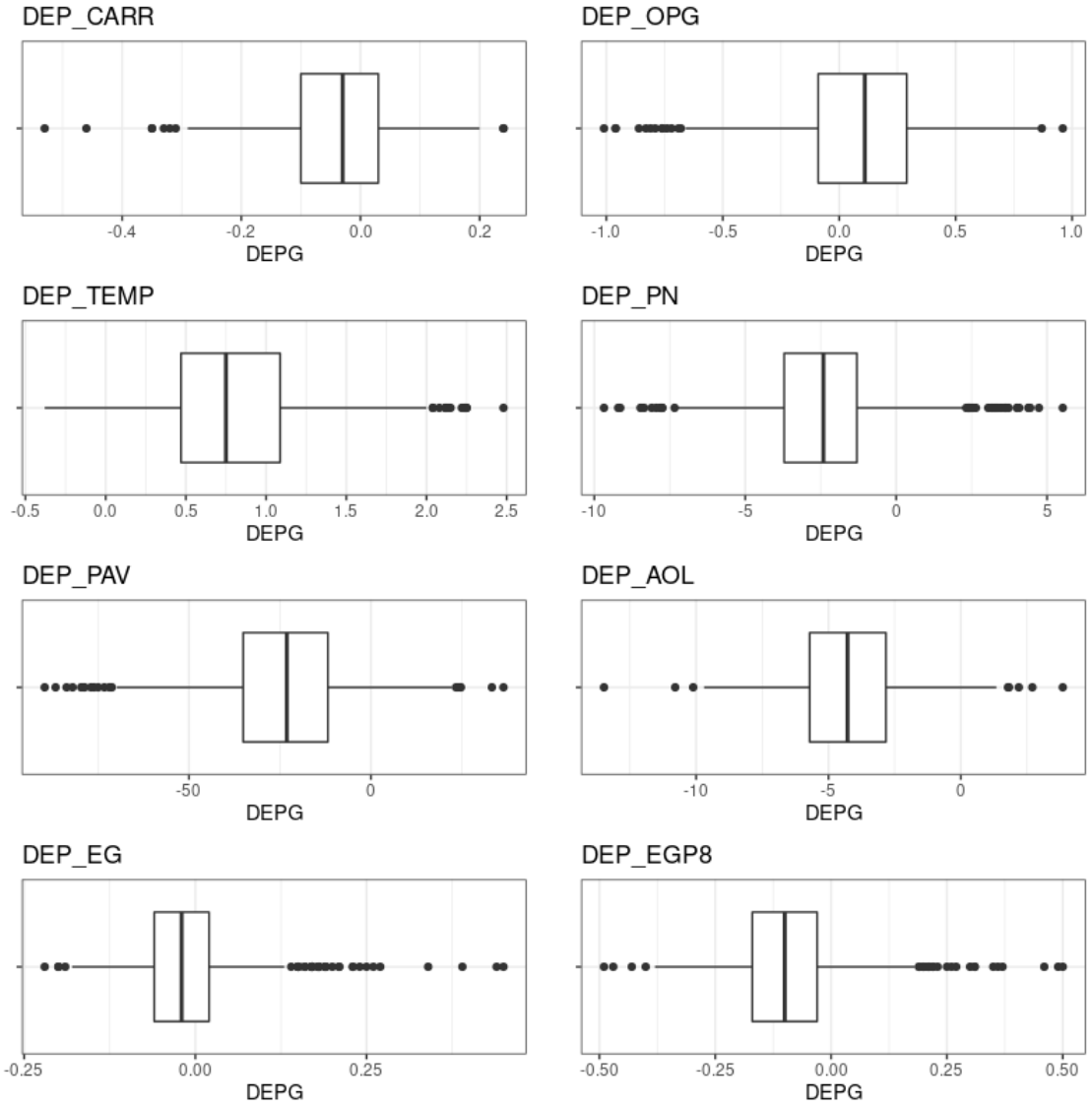
A Tabela 2 mostra a análise descritiva das variáveis quantitativas da base de dados, ou seja, de cada uma das características dos animais que possuem valores de DEPG. Cada linha da tabela indica uma medida na seguinte ordem: valor mínimo, primeiro quartil, segundo quartil ou mediana, terceiro quartil, valor máximo, média, desvio padrão e variância. Os valores da tabela estão expressos em DEPG e portanto correspondem às unidades de medida intrínsecas da característica em questão, por exemplo, os valores da coluna PN estão em kg.

Tabela 2 – Estatística descritiva da base de dados

	CARR	OPG	TEMP	PN	PAV	AOL	EG	EGP8
Mín.	-0,53	-1,01	-0,38	-9,68	-89,65	-13,48	-0,22	-0,49
Q_1	-0,1	-0,09	0,47	-3,71	-35,12	-5,71	-0,06	-0,17
Q_2	-0,03	0,11	0,75	-2,41	-23,13	-4,28	-0,02	-0,10
Q_3	0,30	0,29	1,08	-1,30	-11,84	-2,83	0,02	-0,03
Máx.	0,24	0,96	2,48	5,52	36,41	3,85	0,45	0,50
Méd.	-0,035	0,093	0,799	-2.462	-23.590	-4.282	-0.018	-0.095
DP.	0,095	0,298	0,468	2,049	18,245	2,168	0,069	0,113
Var.	0,009	0,089	0,219	4,200	332,880	4,702	0,004	0,013

Na Figura 4 observa-se um gráfico tipo *boxplot* para cada variável da base de dados. Em todos os gráficos é possível notar a presença de *outliers*, ou seja, valores discrepantes. A presença de *outliers* não quer dizer que alguns dados foram coletados erroneamente mas indica que alguns animais possuem características com valores extremos, tanto negativos como positivos, em relação à média da base de dados. De toda forma, em qualquer base de dados os *outliers* devem ser investigados para afastar a possibilidade de erro de mensuração ou de inserção. Esse procedimento não foi conduzido no trabalho pelo fato do desenvolvimento do algoritmo ser independente das instâncias recebidas.

Figura 4 – Boxplot das variáveis da base de dados



Fonte: Autor (2021)

4 O ALGORITMO BRANGUSSELECTION

4.1 Caracterização do problema

Esta seção apresenta formalmente o problema de maximização de seleção de acasalamentos, bem como as demais definições necessárias para o correto entendimento do problema e de sua solução. Todas as definições provêm ou são adaptadas de (FERREIRA; YOKOO; MOTTA, 2021), cujos resultados serão discutidos no Capítulo 5.

O problema de maximização de escolha de acasalamentos procura otimizar as opções de acasalamento de acordo com o índice de seleção escolhido, objetivando o maior índice possível da prole, calculado como a média dos índices dos progenitores. O processo de seleção dos animais que vão acasalar é feito previamente, sobre o índice de seleção escolhido e não será tratado nesta dissertação. A partir do processo de seleção é designado o conjunto de animais que deverão acasalar. A questão abordada é qual é o melhor pareamento de machos e matrizes que levará ao maior índice possível da futura prole. Por exemplo, um produtor pode escolher 100 machos e 300 fêmeas de seu rebanho para acasalar. A ideia é que o esquema final de acasalamentos seja o ótimo, isto é, o que traz o melhor resultado para o produtor, em acordo com o índice de seleção escolhido.

Neste trabalho, o índice de seleção utilizado objetiva trazer o maior retorno financeiro ao sistema de produção, mas a solução proposta independe de como o índice é construído e pode ser executada sobre qualquer índice. Atualmente, índices de seleção são utilizados amplamente pelos diversos programas de melhoramento genético para facilitar o processo de seleção dos pais da próxima geração do rebanho. Um índice de seleção consiste em um conjunto de características (desejadas ou não) que podem ocorrer nos animais, com um valor de ponderação sobre essas características. A Definição 4.1 apresenta essa ideia formalmente.

Definição 4.1 (Índice de Seleção) *Seja T um conjunto de características de animais. Um* índice de seleção *é um par $\mathscr{I} = (T, w)$ onde $w : T \to \mathbb{R}$ é a função de ponderação do índice.*

A função de ponderação do índice de seleção designa, para cada característica levada em consideração pelo índice, o peso relativo dessa característica. Não há restrições quanto aos valores que a ponderação das características e nem os valores precisam estar normalizados. Valores negativos são aceitáveis, visto que eles representam penalidades,

no sentido de que um valor alto para a característica correspondente torna o valor do índice do animal menor.

O objetivo do índice de seleção é estimar a contribuição de cada animal para a próxima geração do rebanho. A contribuição individual do animal é calculada pela soma do valor de suas características ponderadas pela importância relativa, que é definida pelo índice de seleção. A Definição 4.2 apresenta a formalização da contribuição individual.

Definição 4.2 (Contribuição individual) *Seja A um conjunto de animais, $\mathscr{I} = (T, w)$ um índice de seleção e $v : T \times A \to \mathbb{R}$ uma função que retorna o valor de uma característica para cada animal no conjunto A. A função $\iota^{\mathscr{I}} : A \to \mathbb{R}$ que para cada animal $a \in A$, retorna sua* contribuição individual *de acordo com o índice de seleção $\mathscr{I}$, é calculada como*

$$\iota^{\mathscr{I}}(a) = \sum_{k \in T} v(k,a) \cdot w(k)$$

Quando o índice de seleção é entendido a partir do contexto, escrevemos $\iota(a)$ em vez de $\iota^{\mathscr{I}}(a)$.

A contribuição de acasalamento é a média da contribuição individual dos progenitores (macho e matriz). Nos casos onde o grau de endogamia supera o limite máximo definido pelo produtor, é atribuído à contribuição o valor $-\infty$ (menos infinito) para excluir a possibilidade dessa opção de acasalamento compor a solução final. A Definição 4.3 apresenta formalmente a definição de contribuição de acasalamento.

Definição 4.3 (Contribuição de acasalamento) *Seja $A = M \cup F$ um conjunto de animais, onde M é o conjunto de machos e F é o conjunto de matrizes, com $M \cap F = \emptyset$, $\mathscr{I}$ o índice de seleção, $r : A \times A \to \mathbb{R}^+$ o grau em que dois animais estão relacionados, e l o limite máximo de endogamia. A* contribuição de acasalamento $\pi : M \times F \to \mathbb{R}$ *de um macho $m \in M$ e de uma matriz $f \in F$ é calculado da seguinte forma:*

$$\pi(m,f) = \begin{cases} (\iota^{\mathscr{I}}(m) + \iota^{\mathscr{I}}(f))/2 & r(m,f) \leq l \\ -\infty & r(m,f) > l \end{cases}$$

onde $\iota^{\mathscr{I}}$ é a contribuição individual de cada animal, dado um índice de seleção $\mathscr{I}$.

A Definição 4.3 permite que o produtor escolha o limite máximo de endogamia que considera aceitável na escolha dos acasalamentos. Se nenhum tipo de relação de parentesco é aceitável na escolha dos cruzamentos, basta considerar o limite máximo de endogamia $l = 0$.

Uma vez estabelecidas todas as definições necessárias, é possível formalizar o problema que trata este trabalho. Um problema pode ser formalmente definido pelas suas entradas e saída esperada em função das entradas recebidas. A Definição 4.4 apresenta o problema de maximização de seleção de acasalamento.

Definição 4.4 (Problema de maximização de seleção de acasalamentos) *Seja $A = M \cup F$ um conjunto de animais, sendo M e F o conjunto de machos e matrizes, respectivamente. Seja $\Pi_{|M|\times|F|}$ a matriz de contribuição, onde cada π_{ij} representa a contribuição calculada que o par $(m_i, f_j) \in M \times F$ fornece ao rebanho esperado da próxima geração, ou o valor $\pi(m_i, f_j)$. Seja $t : M \to \mathbb{N}$ a função que, para cada touro, indica seu limite máximo de utilização no processo de cruzamento. A saída do* problema de maximização de seleção de acasalamento*, dada a entrada Π, é uma matriz binária $\mathscr{B}^*$, com dimensão $|M| \times |F|$ onde cada elemento b^*_{ij} de $\mathscr{B}^*$ é*

$$b^*_{ij} = \begin{cases} 1 & \textit{se o macho } i \textit{ e a matriz } j \textit{ devem acasalar} \\ 0 & \textit{caso contrário} \end{cases}$$

sujeito às seguintes restrições:

$\sum_{i=1}^{|M|} b^*_{ij} = 1$, *para cada matriz* j
$\sum_{j=1}^{|F|} b^*_{ij} \leq t_i$, *para cada macho* i *e limite de reprodução individual* t_i

e de modo que, para qualquer outra matriz binária B com as mesmas dimensões, temos que

$$\sum_{i=1}^{|M|}\sum_{j=1}^{|F|} \pi_{ij} b_{ij} \leq \sum_{i=1}^{|M|}\sum_{j=1}^{|F|} \pi_{ij} b^*_{ij}$$

Resumidamente, a entrada do problema de maximização é uma matriz de contribuição de acasalamentos Π com dimensão $|M| \times |F|$ e a saída é uma matriz binária $\mathscr{B}^*$ com dimensão $|M| \times |F|$. Cada elemento da matriz binária $\mathscr{B}^*$ deverá ser 1, indicando que o par deve acasalar, ou 0, indicando que o par em questão não faz parte da solução do problema. Em relação às restrições, para cada vaca j a soma dos elementos $\sum_{i\in M} b^*_{ij}$ deverá ser 1, ou seja, cada vaca deve ser coberta por exatamente um touro. Por outro lado, para cada touro i, a soma dos elementos $\sum_{j\in F} b^*_{ij}$ deverá ser igual ou menor ao seu limite de uso nas reproduções. Por exemplo, se um touro está limitado a ser usado 30 vezes, a soma deverá ser 30 ou menos.

O problema da maximização pode ser caracterizado como um problema de otimização em virtude de sua saída ser a melhor solução dentre um conjunto de soluções

possíveis, no que é denominado espaço de estados das soluções (CORMEN et al., 2002). Como a saída do problema é uma função inteira com domínio em F e contradomínio em M (materializada, no caso, por uma matriz), podemos modelar o problema como uma instância de Programação Inteira (HILLIER; LIEBERMAN, 2013). A Programação Inteira é um problema da classe dos NP-difíceis, ou seja, problemas cujo problema de decisão equivalente é NP-completo. No entanto, a análise do processo feito para encontrar uma solução pode dar pistas como a classe de complexidade do problema.

Um dos objetivos deste trabalho é investigar a classe de complexidade do problema da maximização de seleção de acasalamentos. Embora o tamanho do espaço de estados do problema inviabilize qualquer solução baseada em enumeração exaustiva das soluções possíveis, esse não é um critério para determinar a classe de complexidade do problema. A abordagem escolhida para a solução do problema, que permite uma boa investigação sobre as características das soluções ótimas encontradas, é o de combinar programação linear contínua e o método *branch-and-bound* para resolver um problema de programação inteira. As seções seguintes descrevem a modelagem do problema e de sua solução com essas técnicas.

4.2 Modelo do problema em programação linear/inteira

Um problema de otimização pode ser modelado como um problema de programação linear quando tanto a sua função objetivo – a função que representa a combinação dos valores que se quer maximizar ou minimizar – quanto as restrições do problema podem ser representadas por equações ou inequações lineares (CORMEN et al., 2002; GOLDBARG; LUNA, 2005). Quando as variáveis de um problema de programação linear somente podem assumir valores inteiros temos um problema de programação linear inteira ou, simplesmente, programação inteira. Se todas as variáveis possuem domínio no conjunto de números inteiros $\mathbb{Z}$, o problema é denominado *programação inteira pura*; no caso em que algumas variáveis devam ser inteiras e outras possam assumir valores contínuos (no conjunto dos números reais $\mathbb{R}$), o problema é qualificado como de *programação inteira mista* (HILLIER; LIEBERMAN, 2013). O caso do problema dos acasalamentos é específico já que as variáveis são binárias, isto é, assumem os valores 0 (zero) ou 1 (um). Sendo assim, temos um problema de *programação inteira binária*.

Apesar de bastante similares, esses problemas possuem diferenças importantes

com relação aos seus algoritmos de solução. Enquanto que um problema de programação linear com variáveis com domínio no conjunto dos números reais pode ser resolvido em tempo polinomial, um problema de programação inteira, mesmo aqueles em que as variáveis tomam somente valores binários, pertence à classe dos NP-difíceis. Conforme já apresentado no Capítulo 2, não se conhece algoritmo eficiente para resolver essa classe de problemas (GAREY; JOHNSON, 1979).

O problema da maximização dos valores de acasalamento pode ser descrito a partir dos elementos de entrada do problema, estabelecidos na Definição 4.4: sejam $\pi : M \times F \rightarrow \mathbb{R}$ a função que calcula a contribuição do par $(i, j) \in M \times F$, conforme a Definição 4.3, $t : M \rightarrow \mathbb{N}$ a função que, para cada macho, retorna o seu valor máximo de uso no processo de acasalamento e c_{ij} o conjunto de variáveis do problema, que tomam o valor 1 (um) quando o acasalamento do par $(i, j) \in M \times F$ pertence à solução do problema e o valor 0 (zero) caso contrário. Então o problema da maximização dos valores de acasalamento pode ser descrito como um problema de programação inteira onde a função objetivo é dada por

$$\text{MAX} \sum_{i=1}^{|M|} \sum_{j=1}^{|F|} \pi_{ij} c_{ij} \tag{1}$$

estando o valor da função objetivo sujeito às seguintes restrições:

$$\begin{aligned} &\forall j \in F \ \textstyle\sum_{i=1}^{|M|} c_{ij} = 1 \\ &\forall i \in M \ \textstyle\sum_{j=1}^{|F|} c_{ij} \leq t(i) \end{aligned} \tag{2}$$

Note-se que a função objetivo é uma função linear, visto que o valor da contribuição de cada par $(i, j) \in M \times F$ é um valor constante. As variáveis do problema são os valores c_{ij}, que podem tomar valores no conjunto $\{0, 1\}$.

Para ilustrar a formalização do problema, imagine que temos um total de 6 fêmeas e 3 machos para escolher o esquema de acasalamento, cujas contribuições individuais são apresentadas na Tabela 3 e cuja matriz de consanguinidade entre animais é apresentada na Tabela 4. A matriz de contribuição resultante, assumindo que o produtor não admite nenhum cruzamento entre indivíduos relacionados (ou seja, o limite é igual a zero), é apresentada na Tabela 5.

Tabela 3 – Exemplo do problema do acasalamento – contribuições individuais

f_1	f_2	f_3	f_4	f_5	f_6	m_1	m_2	m_3
126,25	121,19	93,78	85,48	79,04	71,33	74,34	72,87	70,47

Tabela 4 – Exemplo do problema do acasalamento – matriz de consanguinidade

	f_1	f_2	f_3	f_4	f_5	f_6
m_1	0	0,5	0	0	0,5	0
m_2	0,25	0	0	0,25	0	0
m_3	0	0	0	0	0	0

Tabela 5 – Exemplo do problema do acasalamento – matriz de contribuição

	f_1	f_2	f_3	f_4	f_5	f_6
m_1	100,30	$-\infty$	84,06	79,91	$-\infty$	72,83
m_2	$-\infty$	97,03	83,33	$-\infty$	75,95	72,10
m_3	98,36	95,83	82,13	77,97	74,75	70,90

Note-se que, na matriz de contribuição, os animais que têm relação de parentesco aparecem com o coeficiente $-\infty$. Naturalmente que não existe valor numérico $-\infty$: o símbolo indica um valor muito baixo, para minimizar as chances do par em questão aparecer na solução. Na prática, o valor usado é o menor número inteiro (ou o número negativo com maior valor absoluto) disponível na representação usada pela máquina. As linguagens de programação usualmente têm um valor constante associado (denominado INT_MIN, no caso da linguagem C ou $(.Machine\$integer.max) * -1$ na linguagem R).

A instância do modelo de programação inteira, para este exemplo específico, é apresentado a seguir. A função objetivo do modelo, definida pela expressão em 1, com $|F| = 6$ e $|M| = 3$ torna-se a expressão a seguir:

$$\begin{aligned} \text{MAX} \quad & \pi_{11}c_{11} + \pi_{12}c_{12} + \pi_{13}c_{13} + \pi_{14}c_{14} + \pi_{15}c_{15} + \pi_{16}c_{16} + \\ & \pi_{21}c_{21} + \pi_{22}c_{22} + \pi_{23}c_{23} + \pi_{24}c_{14} + \pi_{25}c_{25} + \pi_{26}c_{26} + \\ & \pi_{31}c_{31} + \pi_{32}c_{32} + \pi_{33}c_{33} + \pi_{34}c_{14} + \pi_{35}c_{35} + \pi_{36}c_{36} \end{aligned} \tag{3}$$

Substituindo os valores da Tabela 5 na expressão 3, tem-se a expressão

$$\begin{aligned} \text{MAX} \quad & 100,30c_{11} + (-2147483647)c_{12} + 84,06c_{13} + \\ & 79,91c_{14} + (-2147483647)c_{15} + 72,83c_{16} + \\ & (-2147483647)c_{21} + 97,03c_{22} + 83,33c_{23} + \\ & (-2147483647)c_{14} + 75,95c_{25} + 72,10c_{26} + \\ & 98,36c_{31} + 95,83c_{32} + 82,13c_{33} + \\ & 77,97c_{14} + 74,75c_{35} + 70,90c_{36} \end{aligned} \tag{4}$$

no caso acima, o valor -2147483647 é o menor valor possível de ser associado a uma variável inteira no caso da infraestrutura de hardware/software usada na execução deste

trabalho.

As restrições do problema, apresentadas no conjunto de expressões em 2, neste exemplo são instanciadas como

$$
\begin{aligned}
&c_{11}+c_{21}+c_{31}=1\\
&c_{12}+c_{22}+c_{32}=1\\
&c_{13}+c_{23}+c_{33}=1\\
&c_{14}+c_{24}+c_{34}=1\\
&c_{15}+c_{25}+c_{35}=1\\
&c_{16}+c_{26}+c_{36}=1\\
&c_{11}+c_{12}+c_{13}+c_{14}+c_{15}+c_{16}\leq t(1)\\
&c_{21}+c_{22}+c_{23}+c_{24}+c_{25}+c_{26}\leq t(2)\\
&c_{31}+c_{32}+c_{33}+c_{34}+c_{35}+c_{36}\leq t(3)
\end{aligned}
\tag{5}
$$

Assumindo os valores de contribuições apresentados na Tabela 5 e que os valores máximo de uso dos machos $m_1, m_2, m_3 \in M$ sejam, respectivamente, $t(1)=5$, $t(1)=4$ e $t(1)=3$, as restrições do problema tomam a seguinte forma:

$$
\begin{aligned}
&c_{11}+c_{21}+c_{31}=1\\
&c_{12}+c_{22}+c_{32}=1\\
&c_{13}+c_{23}+c_{33}=1\\
&c_{14}+c_{24}+c_{34}=1\\
&c_{15}+c_{25}+c_{35}=1\\
&c_{16}+c_{26}+c_{36}=1\\
&c_{11}+c_{12}+c_{13}+c_{14}+c_{15}+c_{16}\leq 5\\
&c_{21}+c_{22}+c_{23}+c_{24}+c_{25}+c_{26}\leq 4\\
&c_{31}+c_{32}+c_{33}+c_{34}+c_{35}+c_{36}\leq 3
\end{aligned}
\tag{6}
$$

O modelo construído, dados os conjuntos de machos M e de fêmeas F, gera uma função objetivo com $|F|.|M|$ termos e um conjunto de $|F|+|M|$ restrições.

A estratégia para resolver o problema de programação inteira acima descrito, por meio da combinação das técnicas de programação linear e *branch-and-bound* consiste na transformação de um problema de programação inteira em um problema de programação linear real. A programação linear, ao contrário da programação inteira, pode ser resolvida por algoritmos polinomiais. Especificamente, o método Simplex (HILLIER; LIEBERMAN, 2013) pode ser usado diretamente. Para que isso possa ser feito, é usada

uma estratégia conhecida como “relaxamento” do problema. O modelo construído e apresentado em 1 é caracterizado como inteiro pela exigência das variáveis c_{ij} tomarem valores no conjunto $\{0,1\}$. O problema é relaxado quando essa exigência é flexibilizada, permitindo que as variáveis tomem valores no conjunto $[0,1]$. Dessa forma, o novo modelo pode ser resolvido pelo algoritmo Simplex.

O espaço de soluções do problema de programação linear relaxado contém todas as soluções inteiras para o problema. O algoritmo *branch-and-bound* investiga as soluções inteiras dentro do espaço de soluções, eliminando aquelas que não podem retornar um valor maior do que a melhor solução encontrada até o momento. Essa estratégia permite que o espaço de soluções não precise ser todo investigado. Problemas NP-completos ainda geram uma árvore de tamanho proporcional a uma função exponencial no número de variáveis, mesmo que nem todo o espaço seja investigado. A análise da solução construída e dos resultados da técnica apontaram para a existência de um algoritmo de complexidade polinomial, como será discutido ao final do trabalho.

Para exemplificação, a Tabela 6 apresenta a solução do exemplo do problema do acasalamento.

Tabela 6 – Exemplo do problema do acasalamento – solução relaxada

	f_1	f_2	f_3	f_4	f_5	f_6
m_1	1	0	1	1	0	1
m_2	0	1	0	0	1	0
m_3	0	0	0	0	0	0

4.3 Codificação do modelo

Esta seção apresenta a codificação do modelo, ou seja, o conjunto de algoritmos desenvolvidos para a resolução do problema de maximização do valor esperado da seleção de acasalamentos. Os algoritmos desenvolvidos foram codificados nas linguagens de programação R e C++. Todos os códigos encontram-se no Apêndice A.

O primeiro algoritmo desenvolvido, para fins de teste dos resultados encontrados, foi o algoritmo de força bruta. Um algoritmo é dito de força bruta quando explora a totalidade do espaço de soluções possíveis. Naturalmente que o tempo de execução desse tipo de algoritmo é linear no tamanho do espaço de soluções, que usualmente é exponencial nas variáveis do problema. Como existem $|M|^{|F|}$ funções entre os conjuntos de reprodutores machos M e fêmeas F, o algoritmo resultante tem complexidade

exponencial e somente pode ser usado para tamanhos de problemas bastante restritos.

O algoritmo de força bruta foi construído com base em duas funções: *permutation* e *calc*. A função *permutation* é uma função recursiva que tem como entradas (i) um vetor *v* representando os touros, onde cada elemento no vetor representa um touro. Por exemplo, se for definido três touros, *v* possuirá os valores 1, 2 e 3, (ii) uma variável inteira *n* para controle das permutações de *p*, que é um vetor gerado dentro da função com os elementos de *v*, onde cada posição de *p* representa um fêmea, e (iii) o número de fêmeas nf. A função *permutation* consiste em permutar o vetor *p* com base nos elementos do vetor *v*, de modo que sejam feitos todos os arranjos possíveis com os elementos de *v* em *p*. A variável *n* inicia com o valor 1 e em cada chamada recursiva da função *permutation* vai sendo acrescentada uma unidade. Caso o teste da condição lógica `if (n > nf)` for verdadeiro, é feita a chamada da função *calc*. A saída da função *permutation* é armazenado na variável *v_max*, que é o valor ótimo encontrado em todas as configurações de *p*, ou seja, os acasalamentos.

A função *Calc* tem como entradas (i) um vetor *perm* que contém uma configuração de acasalamento proporcionada pela função *permutation*, onde cada posição representa uma fêmea e cada valor representa um macho, (ii) uma matriz *m* com os valores de contribuição dos acasalamentos e (iii) um vetor *v_max_use* com os valores máximos de uso de cada touro. A saída da função *Calc* é o valor da soma das contribuições dos acasalamentos de *perm* que são válidos. Além de calcular o valor da função objetivo, a função *Calc* valida cada configuração de acasalamento verificando se está de acordo com as restrições. A condição `if (v_max_use[perm[i]] > 0 && m[perm[i],i] != MININT)` que testa cada elemento de *perm* através de um laço de repetição, permite avaliar se os acasalamentos respeitam o uso máximo dos touros contidos *v_max_use* e se o acasalamento avaliado não está restringido com a constante *MININT* na matriz de contribuição de acasalamentos *m*. O código completo do algoritmo de força bruta pode ser apreciado no Apêndice A.

A seguir serão descritas as principais funções que integram o BRANGUSSELECTION. Por questões de espaço não serão apresentados todos os códigos nesta seção.

Para a implementação do cálculo de contribuição individual dos animais foi criada uma função denominada *f_individual_contribution* que possui como entradas uma tabela *db* (arquivo .csv) contendo a base de dados dos animais e um vetor *s_index* contendo os valores dos pesos de cada característica do índice de seleção que será utilizado para

gerar as recomendações dos acasalamentos. Neste trabalho é utilizado o índice de seleção econômico apresentado na Tabela 1, mas devido ao caráter genérico do algoritmo, é possível utilizar outros índices de seleção. Adicionalmente, é necessário fornecer como entradas o valor inicial *inicolDB* e final *endcolDB* das colunas das tabelas que pertencem às DEPG dos animais. Por exemplo, neste trabalho a base de dados é composta por uma tabela onde a primeira e segunda coluna indicam o nome e sexo do animal, respectivamente. A partir da terceira coluna até a décima coluna encontram-se os valores das DEPG dos animais. Sendo assim, é necessário indicar como parâmetro o valor inicial *inicolDB* como 3 e o valor final *endcolDB* como 10. A seguir, exemplifica-se a chamada da função: `dbexemplo <- f_individual_contribution(db = dbase, s_index = selection_index, inicolDB = 3, endcolDB = 10)`. Para que a saída do algoritmo seja correta, cada posição do vetor *s_index* deverá corresponder com a respectiva DEPG da base de dados de entrada, ou, em outras palavras, *s_index* deve estar organizado pela ordem das características apresentadas na tabela *db*. A saída do algoritmo *f_individual_contribution* é uma tabela contendo os dados dos animais com adição de uma nova coluna chamada *v* que contém os valores da contribuição individual dos animais para o futuro rebanho. A função *f_individual_contribution* faz uso de dois laços de iteração para efetuar o cálculo de contribuição. O primeiro laço de repetição, percorre toda a tabela *db* de forma vertical, ou seja, animal por animal. O segundo laço de repetição, que está aninhado ao primeiro, percorre a tabela *db* de forma horizontal, lendo os valores das DEPG individualmente. Os valores parciais que vão sendo calculados são somados na variável chamada *individual_contribution*. Finalmente, o valor total da contribuição individual do animal é armazenado em um vetor chamado *v*, que armazena todos os valores de contribuição dos animais da base de dados.

Para a implementação do cálculo da contribuição de acasalamento foi criado um algoritmo chamado *f_contribution_matrix* que possui como entradas a base de dados dos animais *db* já com os valores de contribuição individual dos animais computados; a matriz de consanguinidade *m_inbreed*; e o valor máximo de endogamia *max*. A chamada da função é exemplificada a seguir: `output <- f_contribution_matrix(db = dbase, m_inbreed = m_inbreeding, max = 0)`. A principal saída é uma matriz de contribuição de acasalamentos *M*. Adicionalmente, o algoritmo *f_contribution_matrix* fornece duas saídas mais: uma tabela com os dados dos touros e outra tabela com os dados das fêmeas, que posteriormente são necessárias para a exibição final da recomendação dos acasalamentos do BRANGUSSELECTION.

O algoritmo *f_contribution_matrix* além de efetuar o cálculo da contribuição de acasalamento, organiza os animais pelo sexo e os ordena pelo seu valor de contribuição individual de forma decrescente. Dessa forma, a matriz de contribuição resultante *M* fica ordenada pelo valor do índice dos animais. Essa ordenação foi efetivada pois pareceu uma heurística adequada para organizar o processo de escolha, visto que escolher os maiores índices possíveis primeiro é uma boa chance de conseguir um bom resultado. Essa estratégia não só diminuiu bastante o tempo total de execução do algoritmo (de quase duas horas para cerca de 15 minutos) como foi usada para construir outra estratégia ótima de solução do problema, conforme discussão no capítulo seguinte.

A função *f_contribution_matrix* faz uso de dois laços de iteração para efetuar o cálculo da contribuição de acasalamento e criar a matriz *M*. O primeiro laço de repetição percorre a tabela dos machos enquanto o segundo laço de repetição, que está aninhando ao primeiro, percorre a tabela das fêmeas. Para cada par dos possíveis progenitores é feito um teste de verificação do limite máximo de endogamia. Caso o par de animais esteja dentro do tolerável, é feito o cálculo da contribuição do acasalamento e seu valor armazenado em uma estrutura vetorial, onde cada posição *k* do vetor *v* corresponde a um par de animais, conforme ilustra o seguinte código: `v[k] <- (males[i,v_column] + females[j,v_column])/2}`[1]. Por outro lado, se o limite máximo de endogamia for ultrapassado, o cálculo da contribuição de acasalamento não é realizado e por conseguinte, é atribuído o valor da constante *MININT* a par de animais em questão. O código completo da função *f_contribution_matrix* pode ser consultado no Apêndice A.

Conforme as restrições do modelo de programação inteira mostradas na Seção 4.2, a seguir são apresentados os algoritmos para criar as matrizes de coeficientes das restrições das fêmeas e dos machos que são necessárias como entradas no método *lp* pertinente ao pacote *lpSolve* ou no caso do método *lpsymphony_solve_LP*, referente ao pacote *lpsymphony*, conforme explicação apresentada na Seção 3.4.

O seguinte código apresenta a função *f_coefficients_restrictions_females_c*, que cria a matriz de coeficientes das restrições da fêmeas.

```
library(Rcpp)
cppFunction(
  "NumericMatrix f_coefficients_restrictions_females_c(NumericMatrix m) {

    int nfemales = m.ncol();
    int nmales = m.nrow();
    NumericMatrix coef(nfemales, nfemales * nmales);
```

[1] A variável *v_column* armazena o valor referente ao número da coluna da tabela que possui os valores da contribuição individual dos animais.

```
    for (int female = 0; female < m.ncol(); female++){
      int k = 0;
      NumericVector v(nfemales * nmales);

      for (int i = 0; i < m.nrow(); i++) {
        for (int j = 0; j < m.ncol(); j++) {

          if (j == female){
            v[k] = 1;
          } else {
            v[k] = 0;
          }
          k = k+1;
        }
      }
    coef(female,_) = v;
    }
    return(coef);
  }
  "
)
```

Similarmente, a seguir é mostrado o código da função *f_coefficients_restrictions_males_c*, que cria a matriz de coeficientes das restrições dos machos.

```
library(Rcpp)
cppFunction(
  "NumericMatrix f_coefficients_restrictions_males_c(NumericMatrix m) {

    int nfemales = m.ncol();
    int nmales = m.nrow();
    NumericMatrix coef(nmales, nfemales * nmales);

    for (int male = 0; male < nmales; male++){
      int k = 0;
      NumericVector v(nfemales * nmales);

      for (int i = 0; i < nmales; i++) {
        for (int j = 0; j < nfemales; j++) {

          if (i == male){
            v[k] = 1;
          } else {
            v[k] = 0;
          }
          k = k+1;
        }
```

```
      }
    coef(male,_) = v;
    }
    return(coef);
  }
  "
)
```

Salienta-se que as funções *f_coefficients_restrictions_females_c* e *f_coefficients_restrictions_males_c* foram codificadas na linguagem de programação C++. Para a integração com o R, foi utilizado o pacote Rcpp, que possibilita a execução de blocos de códigos em C++ inseridos através de uma *string*. Como C++ é uma linguagem compilada, os laços de repetição executam mais rapidamente que os laços de repetição codificados diretamente em R, que é uma linguagem interpretada. Esta abordagem se apresentou vantajosa nos testes realizados principalmente quando foram utilizados muitos animais. Em ambas funções, a entrada é a matriz de contribuição de acasalamentos e a saída é uma matriz contendo os coeficientes das restrições dos animais conforme o modelo de programação inteira apresentado.

A seguir, será listado o algoritmo principal, que utiliza todas as funções descritas nesta seção, e as demais funções criadas que podem ser consultadas no Apêndice A deste trabalho. Os códigos serão apresentados de forma segmentada para facilitar as explicações e usam valores específicos, para ilustração.

```
#Importacao e preparacao da base de dados.
base <- read.csv(
"Documentos/scripts-R/Brangus_Selection/Base_Brangus.csv",
sep = ",", header = TRUE)
dbase <- base
dbase$Animal <- as.character(dbase$Animal)
dbase$Sexo <- as.factor(dbase$Sexo)

#Definicao do numero de machos e femeas.
nmales = 50 #machos
nfemales = 350 #femeas

#Amostra da base de dados.
dbase <- f_sample_database(dbase, nmales, nfemales)

#Definicao do inidice de selecao.
selection_index <- c(116.0879975, -9.1900268, 23.8195584,
6.0030468, -0.5495934, 0.1866388, 131.3764082, 65.8745087)

#Matriz de consanguinidade.
m_inbreeding <- f_inbreeding_matrix(nfemales, nmales)
```

```
#Calculo da contribuicao individual.
dbase <- f_individual_contribution(dbase, selection_index, 3, 10)

#Matriz de contribuicao de acasalamento.
output <- f_contribution_matrix(dbase, m_inbreeding, 0)
m <- output[[1]]
```

Na primeira parte é feita a importação da base de dados, define-se o número de machos e fêmeas que serão utilizados. Logo, é gerada uma amostra da base de dados com base ao número de animais informados. No caso de usar a base completa, é necessário informar o número total de machos e fêmeas, sendo dispensável a geração da amostra da base de dados. Também, determina-se o índice de seleção que será utilizado para a otimização dos acasalamentos, através de um vetor que contém os valores das ponderações. Na sequência, é criada a matriz de consanguinidade, que neste exemplo foi gerada aleatoriamente através da função *f_inbreeding_matrix*. Para a importação de uma matriz de consanguinidade, substitui-se a chamada da função por `read.csv()`, devendo ser especificado o nome do arquivo. Por último, é feito o cálculo da contribuição individual e contribuição dos acasalamentos. No exemplo, foi determinado zero para a tolerância máxima de endogamia.

```
#Coeficientes da funcao objetivo.
f.obj <- as.vector(t(m))

#Matriz dos coeficientes das restricoes.
crm <- f_coefficients_restrictions_males_c(m) #machos
crf <- f_coefficients_restrictions_females_c(m) #femeas
f.con <- rbind(crm,crf)

#Sinais de igualdade/desigualdade correspondente as restricoes.
#Para os machos o sinal deve ser menor ou igual (<=).
f.dir.males_lpsolve <- matrix("<=", nrow = 1, ncol = nmales)
#Para as femeas o sinal deve ser igual (=).
f.dir.females_lpsolve <- matrix("=", nrow = 1, ncol = nfemales)

f.dir_lpsolve <- cbind(f.dir.males_lpsolve, f.dir.females_lpsolve)

#Definicao do numero maximo de uso de touros.
f.rhs.males <- matrix(30, nrow = 1, ncol = nmales)

#Cada vaca so acasalar uma unica vez por solucao.
f.rhs.females <- matrix(1, nrow = 1, ncol = nfemales)

f.rhs <- cbind(f.rhs.males, f.rhs.females)
```

```
#Libera espaco na memoria da maquina.
rm(crm, crf, f.dir.males, f.dir.females, f.rhs.males, f.rhs.females)
gc()

# 1: lpSolve (Default)
# 2: lpsymphony
test <- f_solver(1)
test
```

Na segunda parte é criado um vetor com os coeficientes da função objetivo, uma matriz com os coeficientes das restrições dos machos e uma matriz com os coeficientes das restrições das fêmeas. São atribuídos a uma matriz os sinais de igualdade/desigualdade referente às restrições do modelo do problema em programação linear/inteira. Nesta etapa, define-se o número máximo de uso de cada touro. Neste exemplo são definidos 30 usos para todos os touros, mas é possível informar os valores especificamente para cada touro através da substituição do valor único por um vetor contendo os valores individuais. Finalmente, é chamada a função *f_solver* que proporciona a solução do problema.

A função *f_solver* é o solucionador do problema. Esta função faz uso de pacotes para o algoritmo Simplex e *branch-and-bound*; das saídas produzidas pelas demais funções apresentadas; e funções auxiliares que facilitam a apresentação final dos resultados. A entrada da função *f_solver* é o parâmetro *pack*, que indica qual pacote será utilizado, sendo 1 para o *lpSolve* e 2 para o *lpsymphony*. A saída da função *f_solver* é uma lista de cinco objetos a seguir: *Objective Value* que armazena uma variável *objval* do tipo *double* contendo um valor numérico de ponto flutuante referente ao valor da função objetivo calculada; *Binary Solution* que armazena uma matriz binária *solution* referente à solução ótima encontrada; *Male Count* que contém um vetor numérico *males_usage* com valores inteiros referente ao número de usos dos touros/sêmen; *Matings* que contém um *data.frame* nomeado *matings* com os acasalamentos recomendados; e *Sires* que armazena um vetor numérico *males_used* somente com os touros/sêmen que foram utilizados e a frequência de uso, em que o índice do vetor é o código do animal constante na base de dados utilizada. A seguir, será apresentada a função *f_solver* por partes para facilitar as descrições.

```
#Funcao f_solver.
f_solver <- function(pack){
  if (pack == 1) {
    library(lpSolve)
    Z_ipSolve <- lp("max", f.obj, f.con, f.dir_lpsolve,
    f.rhs, all.bin = TRUE)
    objval <- Z_ipSolve$objval
```

As entradas do método `lp` correspondem aos coeficientes da função objetivo, coeficientes das restrições, sinais de igualdade/desigualdade e coeficientes do lado direito das restrições. O parâmetro `"max = TRUE"` indica que é um problema de maximização e o parâmetro `types = "B"` indica que as variáveis são binárias. A saída da função `lp` é armazenada num objeto nomeado `Z_ipSolve` da classe *list*. Conforme o manual do R, uma *list* é uma coleção de objetos que podem ser de diferentes classes.

```
#Cria a matriz com as solucoes (acasalamentos recomendados).
solution_ipSolve <- matrix(Z_ipSolve$solution, nrow = nmales,
ncol = nfemales, byrow = TRUE)
solution <- solution_ipSolve

#Contagem do uso de cada touro para verificar se
#as restricoes de uso foram atendidas.
males_usage_ipSolve <- f_male_count(solution_ipSolve)
males_usage <- males_usage_ipSolve
```

Para facilitar o mapeamento dos touros que compõem a solução final, foi criada a função `f_male_count`, que verifica a quantidade de vezes que cada macho foi utilizado na recomendação. Independente se o animal foi utilizado ou não, irá constar neste vetor. A saída é um vetor composto de números inteiros onde cada posição do índice do vetor representa um touro, na ordem em que se encontram na base de dados.

```
#Mostra a recomendacao de acasalamentos.
#Ordem das colunas: femeas, machos.
dbmales <- output[[2]]
dbfemales <- output[[3]]
matings_ipSolve <- f_mating(solution_ipSolve,
dbfemales, dbmales)
matings <- matings_ipSolve

#Mostra os nomes dos machos com as respectivas
#quantidades de uso.
males_used_ipSolve <- table(matings_ipSolve$Macho)
males_used <- males_used_ipSolve
```

A função *table* do pacote *base* do R, utilizado no código `table(acasalamentos_ipSolve$Macho)` mostra os nomes/identificação dos machos com as respectivas quantidades de uso, excluindo os machos inutilizados na recomendação apresentada pelo algoritmo. A saída de maior interesse para o produtor que oferece o algoritmo criado neste trabalho é a recomendação completa de acasalamentos que proporcionará a melhor prole do rebanho conforme as restrições impostas. Para apresentar a listagem dos acasalamentos foi criada a função *f_mating*, que possui como entradas a matriz binária da solução, a tabela dos machos e tabela das fêmeas. A função

f_mating cria uma tabela com três colunas, sendo esta a saída da função, na seguinte ordem: identificação do acasalamento, nome da fêmea e nome do touro.

O restante da função *f_solver* é apresentado a seguir. As funcionalidades nesta parte são as mesmas, a diferença é a utilização do pacote *lpsymphony*. No desfecho é possível observar que o resultado final é armazenado na variável *output*, que é o retorno da função.

```
  } else if (pack == 2) {
    library(BiocManager)
    library(lpsymphony)
    Z_lpsymphony <- lpsymphony::lpsymphony_solve_LP(f.obj, f.con,
    f.dir, f.rhs, max = TRUE, types = "B")
    objval <- Z_lpsymphony$objval

    #Cria a matriz com as solucoes (acasalamentos recomendados).
    solution_lpsymphony <- matrix(Z_lpsymphony$solution,
    nrow = nmales, ncol = nfemales, byrow = TRUE)
    solution <- solution_lpsymphony

    #Contagem do uso de cada touro para verificar se
    #as restricoes de uso foram atendidas.
    males_usage_lpsymphony <- f_male_count(solution)
    males_usage <- males_usage_lpsymphony

    #Mostra a recomendacao de acasalamentos.
    #Ordem das colunas: femeas, machos.
    dbmales <- output[[2]]
    dbfemales <- output[[3]]
    matings_lpsymphony <- f_mating(solution, dbfemales, dbmales)
    matings <- matings_lpsymphony

    #Mostra os nomes dos machos com as respectivas
    #quantidades de uso.
    males_used_lpsymphony <- table(matings_lpsymphony$Macho)
    males_used <- males_used_lpsymphony

  } else {

    cat("Incorrect option. Please verify your entry.")

  }

  output <- list("Objective Value" = objval,
  "Binary Solution" = solution, "Male Count" = males_usage,
  "Matings" = matings, "Sires" = males_used)

  return(output)
}
```

O BRANGUSSELECTION foi desenvolvido com base a uma estratégia para resolução do problema que consiste em três etapas: (i) modelagem do problema como um problema de programação inteira, (ii) solução do modelo de programação inteira relaxado (remoção das restrições de integralidade das variáveis) através do algoritmo Simplex, (iii) aplicação do algoritmo *branch-and-bound* para encontrar a melhor solução inteira dentro do espaço de soluções fornecido pelo algoritmo Simplex.

O *branch-and-bound* é um método exato e sua estratégia consiste em dividir para conquistar. Embora o *branch-and-bound* assegure encontrar a solução ótima, não necessita explorar todo o espaço de soluções para isso, como no caso de algoritmos que usam a estratégia de força bruta. O mecanismo que permite ao *branch-and-bound* eliminar grandes quantidades de soluções ineficientes são os limites estimados. Basicamente, o algoritmo *branch-and-bound* possui dois passos: o *branching* e o *bounding*. O *branching* consiste em particionar o espaço de soluções S em dois ou mais conjuntos $S_1, S_2, ..., S_n$. Dessa forma, o problema original é particionado em subproblemas, sendo criada uma árvore de enumeração cujo nó raiz é o espaço de soluções completo, contendo as soluções inteiras e não inteiras, e os nós abaixo são os espaços de soluções que foram particionados, ou seja, os subproblemas. A união dos nós filhos, que são aqueles que estão abaixo de um determinado nó, devem representar o espaço de solução do nó pai, ou seja, a união de $S_1, S_2, ..., S_n$ cobre S. O *bounding* consiste em estimar o limite superior e o limite inferior de acordo com a função objetivo do problema, ou seja, se o problema for de minimização, os limites serão estimados para o valor mínimo da função objetivo; se o problema for de maximização, os limites serão estimados para o valor máximo da função objetivo. Em um problema de maximização, como é o caso do problema de maximização de seleção de acasalamentos, o descarte de nós que possuem soluções inválidas é feito por meio de comparações entre os subconjuntos, por exemplo, se um nó S_1 possui um limite superior menor do que o limite inferior de um outro nó S_2, o primeiro nó S_1 é podado, ou seja, é descartado já que não é possível melhorar o valor da função objetivo. Este procedimento continua consecutivamente nos demais nós. Depois da primeira poda, é feito o processo de ramificação (*branching*) de forma mais restrita nos nós que não foram podados, e conseguintemente, é repetido o processo (*bounding*) nos novos nós. Os procedimentos *branching* e *bounding* continuam até que não seja mais possível particionar os subconjuntos de soluções, encontrando dessa forma, a melhor solução inteira possível (solução ótima inteira). A essência do *branch-and-bound* é explorar inteligentemente o espaço de soluções fornecido pelo

algoritmo Simplex, utilizando métodos que evitam a necessidade de analisar todos os sub-problemas.

5 RESULTADOS E DISCUSSÃO

5.1 Eficiência do algoritmo BRANGUSSELECTION

O objetivo dos testes realizados e apresentados nesta seção é verificar como a função do tempo de execução se comporta na medida que o tamanho da entrada – ou o número de machos e fêmeas – cresce. A intenção do experimento é aumentar gradualmente o tamanho da entrada e medir o tempo de execução do algoritmo, apresentando os resultados graficamente. O procedimento pode dar uma ideia sobre a complexidade média do tempo de execução do algoritmo implementado. No caso da função ter a forma de um polinômio, é um indicativo importante de que o problema não é intratável computacionalmente.

Para mensurar o tempo de execução do algoritmo nos experimentos realizados foi utilizada a função *system.time* do pacote *base* do R. Conforme a documentação oficial do R, a função *system.time* internamente faz uma chamada à função *proc.time*, avalia a expressão passada como argumento, que neste caso é o algoritmo desenvolvido, e faz novamente uma chamada à função *proc.time*, retornando a diferença em segundos entre as duas chamadas *proc.time*. A saída da função *proc.time* é um vetor com os tempos de execução denominados *user*, *system* e *elapsed* que o processo em execução no R utilizou. Neste trabalho foi considerado o tempo *elapsed* que indica o tempo real usado pela CPU na execução do algoritmo. Apesar de ser possível cronometrar fazendo as chamadas *proc.time* de forma manual antes e depois da execução do algoritmo, optou-se pela utilização da função *system.time*, que essencialmente funciona como um cronômetro, facilitando a avaliação do tempo computacional dispendido em uma determinada expressão válida passada como argumento.

A base de dados utilizada para realizar os testes dos algoritmos contém dados de 1366 bovinos da raça *Brangus*, sendo 511 machos e 855 fêmeas. Os dados correspondem à *Diferença Esperada na Progênie aprimorada pela Genômica* (DEPG) das características dos animais que compõem o índice de seleção econômico. O índice de seleção utilizado neste trabalho com os respectivos pesos e objetivos de seleção associados podem ser novamente consultados na Tabela 1 (pág. 32).

Para a automatização dos testes foi criado o algoritmo *Test* que pode ser consultado no Apêndice A. O algoritmo de testes possui como parâmetros o número de repetições/iterações *rexc* de cada instância que será testada; um vetor *num_males* com

o número de machos que serão utilizados; um vetor *num_females* com o número de fêmeas que serão utilizadas; e um vetor *v_use_max_males* com o valor de uso máximo dos machos nos acasalamentos. Assim, a Figura 5 exemplifica uma configuração inicial válida:

Figura 5 – Exemplo de configuração inicial do algoritmo *Test*

```
rexc <- 10
num_males <- c(2, 3, 4)
num_females <- c(3, 5, 6)
v_use_max_males <- c(2, 3, 3)
```

Fonte: Autor (2021)

Cada posição dos vetores representa uma instância específica. No exemplo, o algoritmo *Test* irá executar o teste com a primeira instância com 2 machos, 3 fêmeas e uso máximo de touros em 2; a segunda instância com 3 machos, 5 fêmeas e uso máximo de touros em 3; e por último, a terceira instância com 4 machos, 6 fêmeas e uso máximo de touros em 3. Para cada instância serão feitas 10 execuções, conforme definido na variável *rexc*. Salienta-se que para cada execução é gerada uma amostra da base de dados, que é utilizada para gerar a matriz de consanguinidade, efetuar os cálculos da contribuição individual e de acasalamento e criar da matriz de contribuição de acasalamento.

O algoritmo *Test* tem como saída uma tabela com os seguintes dados: os tempos de execução, em segundos; tempo médio, em segundos; variância; desvio padrão; número de machos; número de fêmeas; e total de animais utilizados. Ademais, são criados um arquivo de texto (.csv) com os dados do teste e um gráfico apresentando a relação de tempo de execução com o número de animais. A Figura 6 apresenta um exemplo de saída do algoritmo *Test* onde foi testado o algoritmo BRANGUSSELECTION.

Figura 6 – Exemplo da saída do algoritmo *Test*

```
Console  Terminal  Jobs
R 4.1.2 · ~/
> Experiment_Results_Dataframe
      V1     V2     V3     V4     V5     V6     V7     V8     V9    V10    Mean Variance S.Deviation Males Females T. Animals
1 24.698 22.211 27.017 21.812 22.247 24.036 22.863 23.656 23.634 23.264 23.5438   2.2952      1.5150    50     855        905
2 30.963 31.022 30.920 31.273 31.090 31.361 31.283 30.916 30.377 30.504 30.9709   0.1038      0.3221   100     855        955
3 63.402 63.923 63.629 63.493 63.338 64.019 62.225 64.949 63.465 63.438 63.5881   0.4615      0.6793   150     855       1005
```

Fonte: Autor (2021)

O primeiro teste foi feito com o algoritmo de força bruta. O algoritmo de força bruta se caracteriza por explorar todas as combinações possíveis de acordo com as entradas (conjuntos de machos e fêmeas), assegurando que a melhor solução seja

encontrada. Caso exista uma solução que respeite todas as restrições impostas, a solução é um ótimo global do espaço de estados. O objetivo deste experimento foi verificar a eficiência do algoritmo de força bruta. Foi testado o tempo de execução em relação ao número de animais selecionados para acasalamento para verificar qual é o limite aproximado de animais a partir do qual o algoritmo deixa de ser utilizável, devido ao seu comportamento exponencial. Os resultados e os parâmetros utilizados nos testes do algoritmo de força bruta se encontram na Tabela 7. As colunas correspondem ao número da instância, número de machos, número de fêmeas, uso máximo de cada touro, tempo médio de execução, variância e desvio padrão, respectivamente. Foram testadas 10 instâncias, em que cada uma foi executada 10 vezes, ou seja, no total foram feitas 100 execuções. Cada execução utilizou uma amostra diferente da base de dados.

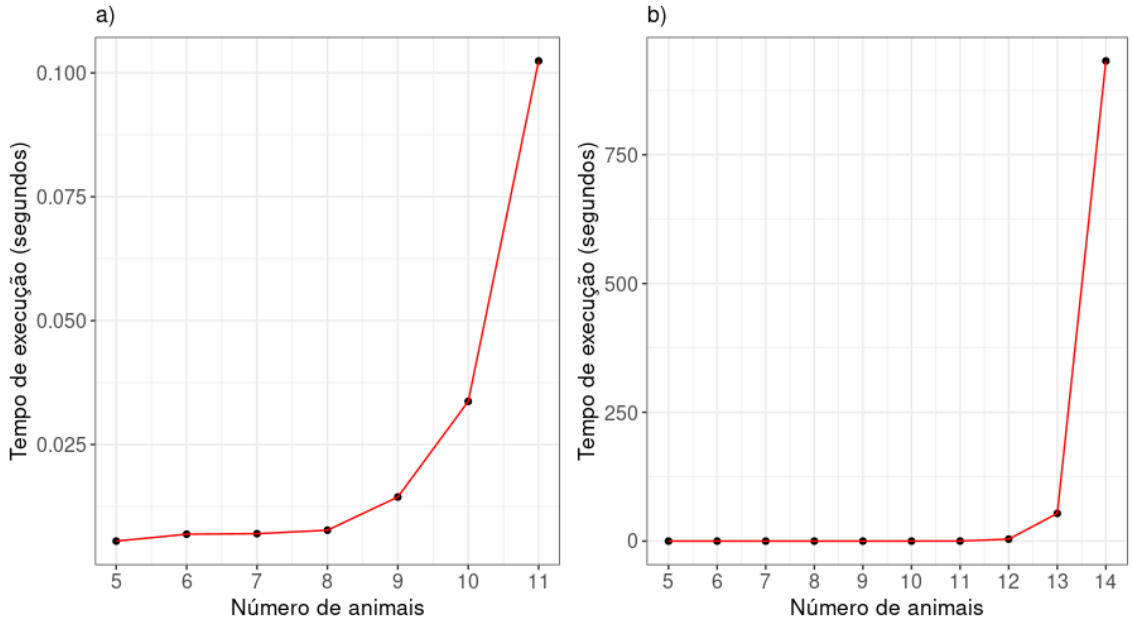
Tabela 7 – Resultados e parâmetros do teste com o algoritmo de força bruta.

n	$\|M\|$	$\|F\|$	max	tempo médio	σ^2	σ
1	2	3	2	0,0055 s	<0,0001	<0,0001
2	2	4	3	0,0069 s	<0,0001	<0,0001
3	2	5	3	0,0070 s	<0,0001	<0,0001
4	2	6	3	0,0077 s	<0,0001	<0,0001
5	3	6	3	0,0144 s	<0,0001	<0,0001
6	3	7	3	0,0337 s	0,0001	0,01
7	3	8	3	0,1024 s	0,0001	0,01
8	4	8	3	3,8705 s	0,0086	0,092
9	4	9	3	53,8114 s	1,1142	1,055
10	4	10	3	932,0849 s	4272,271	65,362

Conforme os resultados apresentados na Tabela 7, na 9 instância o algoritmo de força bruta utiliza em média 54 segundos aproximadamente para encontrar a solução ótima para 13 animais. Na 10 instância, acrescentando apenas 1 unidade no conjunto de fêmeas, o algoritmo de força bruta utilizou em média 932 segundos (16 minutos aproximadamente) para encontrar a solução ótima, aumentando cerca de 1626% do tempo médio utilizado na 9 instância.

A Figura 7(a) apresenta os resultados da Tabela 7 da 1 instância até a 7 instância para facilitar a observação da intensificação do incremento de tempo de execução que inicia a partir da utilização de 8 animais. A Figura 7(b) apresenta os resultados completos Tabela 7 onde é possível observar o comportamento exponencial do algoritmo. Conclui-se que esta abordagem é inviável de ser aplicada de forma prática no sentido de ser útil para o produtor. Ainda assim, com pequenas entradas o algoritmo de força bruta poderá ser utilizado.

Figura 7 – Resultados dos testes realizados com o algoritmo de força bruta.



Fonte: Autor

O segundo teste foi realizado com o BRANGUSSELECTION. O algoritmo foi executado 10 vezes para cada uma das 9 instâncias, totalizando 90 execuções. Cada instância possui uma determinada configuração de entrada, que neste caso constituem o número de machos, o número de fêmeas e o número máximo de uso de cada touro. Para cada execução foi gerada uma amostra distinta da base de dados. Os resultados do teste e as configurações de entrada de cada instância podem ser corroborados na Tabela 8. As colunas indicam o número da instância, número de machos, número de fêmeas, uso máximo de cada touro, tempo médio de execução, variância e desvio padrão, respectivamente.

O delineamento do teste realizado com o BRANGUSSELECTION consistiu em fixar um número de fêmeas aumentado o número de touros de forma que fossem explorados todos. Neste caso, fixou-se 855 vacas, que é o total de fêmeas da base de dados, e acrescentou-se 50 touros em cada instância, iniciando-se com 100 touros até atingir 500 touros, lembrando que a base de dados utilizada possui 511 touros. O valor do uso máximo de cada touro foi definido como o menor número possível para cobrir o total de vacas.

Conforme os resultados obtidos, o desvio padrão foi considerado baixo em relação à média, uma vez que foi calculado o coeficiente de variação (σ / tempo médio * 100) para cada instância, nos quais seus valores mantiveram-se próximo de 0%, num intervalo

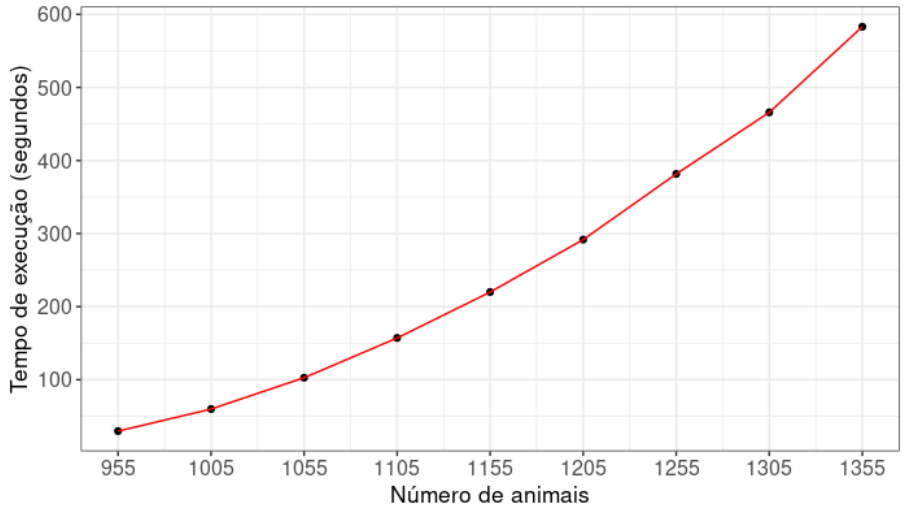
de 0,30% e 1,02%. Os dados indicam que as execuções do teste do BRANGUSSELECTION foram consistentes, próximas do tempo médio de execução, não sendo registrados tempos discrepantes no pior e melhor caso. Na instância 9, observa-se que foram utilizados praticamente todos os animais da base de dados, no qual o algoritmo conseguiu encontrar a solução ótima em aproximadamente 10 minutos.

n	$\|M\|$	$\|F\|$	max	tempo médio	σ^2	σ
1	100	855	9	29,6 s	0,08	0,29
2	150	855	6	59,9 s	0,19	0,43
3	200	855	5	102,7 s	0,44	0,67
4	250	855	4	157,1 s	1,43	1,19
5	300	855	3	220,0 s	5,13	2,26
6	350	855	3	291,7 s	7,19	2,68
7	400	855	3	381,6 s	4,90	2,21
8	450	855	3	465,9 s	19,49	4,41
9	500	855	3	583,11 s	3,19	1,78

Tabela 8 – Resultados e parâmetros do Teste do BRANGUSSELECTION com incremento dos machos.

A figura Figura 8 apresenta graficamente os dados da Tabela 8, evidenciando a relação de tempo de execução com o número total de animais (machos e fêmeas) utilizados em cada instância.

Figura 8 – Gráfico dos resultados referentes à Tabela 8.



Fonte: Autor

Para verificar como o tempo de execução do BRANGUSSELECTION varia ao incrementar o número de fêmeas, foi fixado um número de machos e aumentou-se o número de fêmeas gradativamente, configurando-se o uso máximo de cada touro como o

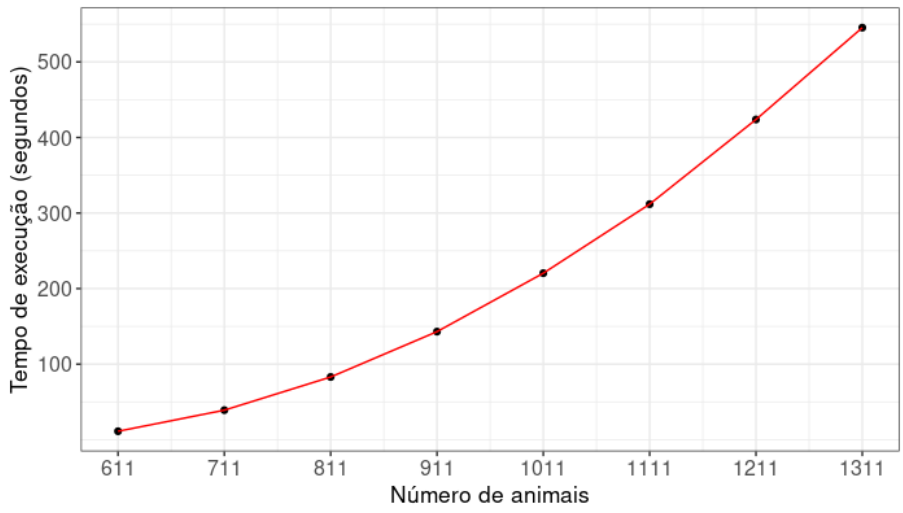
valor mínimo para cobrir todas as fêmeas. A Tabela 9 apresenta os resultados do teste. As colunas correspondem o número da instância, número de machos, número de fêmeas, uso máximo de cada touro, tempo médio de execução, variância e desvio padrão, nessa ordem. O delineamento do teste consistiu em fixar em todas as instâncias o número de touros em 511, que é o total de touros presentes na base de dados. O número de fêmeas foi incrementado consecutivamente em 100, iniciando com 100 fêmeas na primeira instância e finalizando com 800 fêmeas na 8 instância.

n	$\|M\|$	$\|F\|$	max	tempo médio	σ^2	σ
1	511	100	1	11,1 s	0,03	0,19
2	511	200	1	39,1 s	0,08	0,29
3	511	300	1	83,0 s	1,13	1,06
4	511	400	1	143,0 s	1,76	1,32
5	511	500	1	220,4 s	1,82	1,35
6	511	600	2	311,7 s	1,39	1,18
7	511	700	2	423,8 s	2,99	1,73
8	511	800	2	545,3 s	13,49	3,67

Tabela 9 – Resultados e parâmetros do Teste do BRANGUSSELECTION com incremento das fêmeas.

Na figura Figura 8 é apresentado o gráfico construído com os dados da Tabela 9, no qual é possível observar a variação do tempo de execução em relação ao número de animais (machos e fêmeas) usados em cada instância.

Figura 9 – Gráfico dos resultados referentes à Tabela 9.



Fonte: Autor

Observa-se similaridade nos gráficos da Figura 8 e Figura 9 onde os tempos de execução são semelhantes para aproximadamente o mesma quantidade de animais.

Para corroborar esta afirmativa, os resultados dos testes de tempo de execução do BRANGUSSELECTION foram submetidos a análise de variância, sendo aplicados os testes F e Tukey a 5% de significância. A verificação dos pressupostos de homocedasticidade e normalidade dos resíduos efetuou-se com a aplicação dos testes de Levene e Shapiro-Wilk, respectivamente. Ambos pressupostos foram satisfeitos. Os resultados da análise de variância e teste de Tukey apontam que não há diferença estatisticamente significativa entre os dois testes de tempo de execução ($P > 0,05$).

Conclui-se que o BRANGUSSELECTION não é influenciado no tempo de execução pela quantidade de machos em relação à quantidade de fêmeas ou vice-versa que integram a entrada de uma instância. É possível confirmar que tanto os dados da tabela quanto a forma do gráfico apontam para uma função polinomial.

A complexidade do problema da programação linear é $O(nm)$ em que n é o número de variáveis e m é o número de restrições do problema (DAGUSPTA; PAPADIMITRIOU; VAZIRANI, 2006). No caso do BRANGUSSELECTION, como o número de variáveis do problema é $|M| \cdot |F|$ e o número de restrições é $|M| + |F|$ (cf. Seção 4.4), a complexidade é da ordem de $O(|M| \cdot |F| \cdot (|M| + |F|))$, ou seja, é uma função que cresce com o cubo do número de animais, o que pode indicar que o problema pode ter algoritmo ótimo mais eficiente que o resolva, dispensando abordagens com meta-heurísticas que não garantem uma solução ótima ao final de sua execução.

A execução do algoritmo *branch-and-bound*, após a solução do problema relaxado de programação linear, gera um árvore de tamanho exponencial se o problema for, de fato, computacionalmente intratável. Como não é o caso do tempo de execução nesse exercício que, como já mencionado, foi bastante diminuído pela ordenação dos animais conforme sua contribuição individual, a indicação é que o problema não é, como afirmado em alguns trabalhos anteriores, NP-difícil.

5.2 Análise dos resultados do algoritmo BRANGUSSELECTION

Esta seção apresenta a solução encontrada pelo BRANGUSSELECTION e são apresentados cinco exemplos com respectivas análises de seus resultados. Para cada exemplificação, foram utilizados 30 animais (10 touros e 20 vacas), selecionados aleatoriamente da base de dados através da função *f_sample_database*. A matriz de coancestralidade (ou consanguinidade) dos exemplos foi gerada através da função *f_inbreeding_matrix*. Por questões de espaço, não foram incluídas na apresentação a

matriz de contribuição de acasalamento, uma vez que o BRANGUSSELECTION trabalha com números em notação científica nesta matriz. De todas formas, o valor da contribuição do acasalamento pode ser calculado pelo valor médio da contribuição individual do par de animais (cf. definição do Problema 4.3) e que estão disponíveis nas tabelas das bases de dados utilizadas em cada exemplo.

A Tabela 10 apresenta a amostra da base de dados utilizada no primeiro exemplo. As colunas são referentes à posição do animal nas matrizes de consanguinidade e binária, identificação do animal, sexo e valor da contribuição individual.

i	Animal	Sexo	CI	i	Animal	Sexo	CI
1	L235	F	63.61712058	16	L561	F	-14.909469273
2	K606	F	19.798823345	17	L203	F	-19.34316693
3	L896	F	19.05490097	18	K485	F	-22.682576133
4	L954	F	15.812436027	19	L663	F	-34.759984607
5	K421	F	16.294636705	20	L845	F	-42.448213065
6	M71	F	10.1970494	1	L413	M	31.290124497
7	I131	F	9.427484965	2	M2	M	25.035727364
8	L332	F	7.170736257	3	M31	M	19.154223981
9	K339	F	2.278146471	4	L142	M	18.867927321
10	L719	F	1.384712689	5	M316	M	8.933804286
11	K369	F	-3.211896346	6	L752	M	2.210252196
12	K567	F	-4.930507873	7	L373	M	-0.467981516
13	L196	F	-4.75420316	8	M11	M	-3.599822516
14	M9	F	-5.786597483	9	L594	M	-7.774515846
15	M7	F	-14.476966459	10	L422	M	-9.276240506

Tabela 10 – Amostra da base de dados do exemplo 1.

A Figura 10 apresenta a matriz de consanguinidade e a matriz binária correspondentes ao exemplo 1. A matriz de consanguinidade está identificada como *m_inbreeding*, na qual as linhas da matriz representam os machos e as colunas representam as fêmeas, organizados pelo seu valor de contribuição individual, que pode ser verificado na Tabela 10. Para o exemplo 1 foi determinado o uso máximo de cada touro em 4 e o índice máximo de consanguinidade aceitável de 3,125%, ou seja, os animais poderão acasalar caso tenham até 1/32 de parentesco. A matriz binária gerada no exemplo 1 está identificada como *test$'Binary Solution'* cujas dimensões e índices correspondem à matriz de consanguinidade. A interpretação da matriz binária está em conformidade com a definição do Problema 4.4, em que 1 indica que o par deve acasalar e 0 indica que o par não deve acasalar. Pela comparação das matrizes, é possível observar que o BRANGUSSELECTION utilizou somente acasalamentos com um índice de consanguinidade 0 ou 3,125%, atendendo dessa forma as restrições de consanguinidade.

A Tabela 11 exibe a recomendação de acasalamentos incluindo o valor da função objetivo, que determina o valor médio esperado da próxima geração dos animais do

Figura 10 – Matriz de consanguinidade e matriz binária do exemplo 1.

```
> m_inbreeding
        [,1]  [,2]   [,3]   [,4]   [,5]   [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]  [,13] [,14]  [,15]  [,16]  [,17] [,18]  [,19]  [,20]
 [1,]  0.000 0.000  0.000  0.000 12.500  0.000  0.00 3.125  0.00 0.000 25.00 0.000  0.000 0.000 12.500  3.125 12.500 0.000  3.125 50.000
 [2,]  3.125 0.000  3.125  0.000  0.000  3.125  0.00 0.000  0.00 0.000  0.00 0.000 12.500 0.000  3.125  3.125  3.125 3.125 12.500  0.000
 [3,] 12.500 3.125  3.125 50.000  0.000 25.000  0.00 3.125  0.00 0.000 25.00 3.125  3.125 3.125  0.000  0.000  0.000 0.000  0.000  3.125
 [4,]  3.125 0.000  6.250 25.000  0.000  0.000  6.25 0.000 12.50 3.125  0.00 0.000  0.000 6.250  3.125  6.250  0.000 6.250  3.125  0.000
 [5,]  0.000 3.125  0.000  0.000  3.125  0.000  0.00 0.000  0.00 3.125  0.00 0.000  3.125 6.250  0.000  0.000  0.000 3.125  0.000  0.000
 [6,]  3.125 0.000  0.000  3.125 25.000  3.125  6.25 3.125  0.00 0.000  0.00 3.125  0.000 0.000  3.125  0.000 50.000 3.125 12.500  0.000
 [7,] 12.500 6.250  0.000  0.000  3.125  0.000  0.00 6.250  6.25 0.000  6.25 6.250  0.000 3.125  0.000 50.000  3.125 0.000 25.000 25.000
 [8,]  0.000 0.000  0.000  3.125  0.000  0.000  0.00 0.000  0.00 3.125 25.00 0.000  0.000 0.000  0.000  0.000  6.250 0.000  3.125 25.000
 [9,]  0.000 3.125 50.000  0.000  3.125  0.000  0.00 0.000  0.00 3.125  6.25 3.125  0.000 3.125  3.125  3.125  0.000 0.000  3.125  3.125
[10,] 12.500 0.000  0.000  0.000  0.000  0.000 12.50 0.000  0.00 3.125 50.00 3.125  0.000 0.000  6.250  3.125  0.000 3.125 25.000  0.000
> test$`Binary Solution`
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]    1    0    0    1    0    1    0    0    0     0     0     0     0     0     0     0     0     0     1     0
 [2,]    0    1    0    0    0    0    0    0    0     0     1     0     0     0     0     0     1     1     0     0
 [3,]    0    0    1    0    1    0    0    0    0     0     0     0     0     1     0     0     0     0     0     1
 [4,]    0    0    0    0    0    0    0    1    0     1     0     1     0     0     1     0     0     0     0     0
 [5,]    0    0    0    0    0    0    1    0    1     0     0     0     1     0     0     1     0     0     0     0
 [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [7,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [8,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
[10,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
```

Fonte: Autor

exemplo 1. No primeiro exemplo, a solução final consiste na utilização de 5 touros para cobrir as 20 vacas, alcançando um valor de 205,4298 na função objetivo. As informações da tabela são disponibilizadas em formato de planilha eletrônica.

Touro	**Vacas**			
L142	L332	L719	L196	M7
L413	L235	K421	M71	L663
M2	K606	K369	L203	K485
M31	L896	L954	M9	L845
M316	I131	K339	K567	L561
Valor função objetivo: 205,4298				

Tabela 11 – Recomendação de acasalamentos do exemplo 1.

A Tabela 12 apresenta a amostra da base de dados utilizada no segundo exemplo. As colunas são referentes à posição/índice do animal na matriz de consanguinidade e na matriz binária, identificação do animal, sexo e valor da contribuição individual, respectivamente. No exemplo 2, optou-se por utilizar somente animais sem nenhum grau de parentesco, sendo assim, definiu-se o índice máximo de consanguinidade aceitável igual a 0. O uso máximo de vezes que os touros podem acasalar manteve-se em 4.

A Figura 11 apresenta a matriz de consanguinidade e a matriz binária pertencentes ao exemplo 2. É possível notar que os elementos indicados com 1 na matriz binária estão de acordo com os elementos indicados com 0 na matriz de consanguinidade, o que significa que o BRANGUSSELECTION garantiu o uso adequado dos animais apropriados para acasalamento. Note-se que a fêmea 20 (L996) é a que menos opções de acasalamento possui, podendo acasalar somente com o touro 2 (M240) ou 4 (L656). Para consolidar a

i	Animal	Sexo	CI	i	Animal	Sexo	CI
1	L745	F	93.789983803	16	K292	F	-24.238776194
2	L903	F	39.253687008	17	K541	F	-28.487746868
3	M255	F	34.934506162	18	L965	F	-36.739831607
4	L784	F	27.816290624	19	M140	F	-41.958001283
5	L200	F	29.196232504	20	L996	F	-48.793731265
6	M302	F	29.030346597	1	M144	M	41.007034976
7	H883	F	19.483325545	2	M240	M	26.97310794
8	L407	F	17.152962098	3	M273	M	23.97702565
9	L835	F	4.11655629	4	L656	M	22.588103471
10	M5	F	-0.406630742	5	M294	M	14.454669857
11	M9	F	-5.786597483	6	L945	M	10.462133581
12	M60	F	-8.809263058	7	M19	M	6.681225003
13	L595	F	-20.101873585	8	L436	M	2.215535586
14	M16	F	-21.336943945	9	M96	M	0.838857566
15	M74	F	-22.871329159	10	L876	M	-16.016741206

Tabela 12 – Amostra da base de dados do exemplo 2.

solução ótima, o algoritmo efetivou o acasalamento com o touro 4.

Figura 11 – Matriz de consanguinidade e matriz binária do exemplo 2.

```
> m_inbreeding
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7] [,8]  [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,] 0.000 3.125 0.000 0.000 0.000 50.000 0.000 0.000 0.000 3.125 3.125 12.500 3.125 3.125  12.5 0.000 0.000 25.000 12.500 3.125
 [2,] 0.000 0.000 0.000 0.000 0.000  3.125 12.500 0.000 6.250 3.125 0.000 0.000 0.000 3.125   0.0 0.000 3.125 0.000 0.000 0.000
 [3,] 0.000 3.125 3.125 0.000 3.125  0.000  0.000 0.000 25.000 3.125 12.500 0.000 50.000 3.125   0.0 0.000 3.125 3.125 3.125 3.125
 [4,] 50.000 50.000 0.000 6.250 3.125 6.250 0.000 0.000 3.125 0.000 25.000 0.000 50.000 0.000   0.0 0.000 0.000 0.000 0.000 0.000
 [5,] 12.500 0.000 25.000 3.125 0.000 6.250 0.000 3.125 0.000 25.000 3.125 0.000 25.000 3.125   0.0 0.000 25.000 0.000 3.125 3.125
 [6,] 3.125 3.125 25.000 0.000 0.000 3.125 3.125 0.000 3.125 0.000 0.000 0.000 3.125 3.125   0.0 3.125 0.000 0.000 0.000 12.500
 [7,] 0.000 12.500 3.125 50.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 6.250 0.000   0.0 12.500 0.000 50.000 0.000 3.125
 [8,] 0.000 0.000 3.125 3.125 3.125 3.125 3.125 0.000 6.250 0.000 50.000 0.000 0.000 0.000   0.0 0.000 0.000 12.500 3.125 3.125
 [9,] 12.500 3.125 0.000 3.125 25.000 0.000 50.000 0.000 12.500 3.125 3.125 3.125 0.000 0.000   0.0 50.000 6.250 0.000 50.000 6.250
[10,] 6.250 0.000 0.000 0.000 0.000 0.000 0.000 3.125 0.000 12.500 6.250 50.000 25.000 0.000   0.0 0.000 3.125 0.000 25.000 3.125
> test$`Binary Solution`
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]    0    0    1    0    1    0    1    0    1     0     0     0     0     0     0     0     0     0     0     0
 [2,]    0    1    0    0    0    0    0    0    0     0     1     0     1     0     0     0     0     0     1     0
 [3,]    1    0    0    1    0    1    0    1    0     0     0     0     0     0     0     0     0     0     0     0
 [4,]    0    0    0    0    0    0    0    0    0     1     0     0     0     1     0     0     1     0     0     1
 [5,]    0    0    0    0    0    0    0    0    0     0     0     1     0     0     1     1     0     1     0     0
 [6,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [7,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [8,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
[10,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
```

Fonte: Autor

A Tabela 13 demonstra a solução ótima encontrada pelo algoritmo. Conforme os dados da tabela, foram utilizados 5 touros para cobrir as 20 vacas selecionadas, em que cada touro foi usado até seu limite permitido. A função objetivo atingiu o valor 275,6215, sendo superior ao exemplo 1, onde o índice máximo de consanguinidade tolerável foi estipulado com mais amplitude (3,125%), entre tanto, nota-se que os machos do exemplo 2 são superiores nos valores de contribuição individual em relação aos machos do primeiro exemplo. Destaca-se que a melhor fêmea possui um valor superiormente elevado em comparação à melhor fêmea do exemplo 1 (93,789 contra 63,617).

O exemplo 4 consiste em demonstrar se BRANGUSSELECTION atende às restrições de uso máximo de cada touro de forma individual, para isso, foi determinado

Touro	Vacas			
L656	M5	M16	K541	L996
M144	M255	M302	H883	L835
M240	L903	M9	L595	M140
M273	L745	L200	L784	L407
M294	M60	M74	K292	L965
Valor função objetivo: 275,6215				

Tabela 13 – Recomendação de acasalamentos do exemplo 2.

que o melhor touro seja utilizado no máximo 1 vez, o segundo melhor touro, 2 vezes, o terceiro melhor touro, 3 vezes e os restantes dos touros, 4 vezes. O índice máximo de consanguinidade tolerável foi fixado em 3,125%. A Tabela 14 apresenta os animais utilizados para a execução do algoritmo neste exemplo.

i	Animal	Sexo	CI	i	Animal	Sexo	CI
1	K370	F	51.378562293	16	L414	F	-21.73293831
2	K435	F	34.621909594	17	L251	F	-22.8519957
3	M216	F	33.74616516	18	K519	F	-29.291101792
4	L847	F	20.796551396	19	K495	F	-33.453815515
5	K724	F	14.243165301	20	M35	F	-41.730256193
6	K330	F	13.428951936	1	L580	M	23.051552457
7	K412	F	10.446187479	2	L314	M	20.556685032
8	L481	F	7.5029096	3	M192	M	11.435717151
9	L347	F	6.090362465	4	L687	M	8.424750269
10	M217	F	1.844569364	5	L690	M	4.5061653
11	K462	F	-5.612319119	6	L938	M	2.729881368
12	L804	F	-6.309372951	7	L290	M	-6.371581394
13	L989	F	-11.67846381	8	L876	M	-16.016741206
14	L966	F	-14.702599367	9	L978	M	-33.395480466
15	L694	F	-18.678309742	10	L378	M	-44.458431907

Tabela 14 – Amostra da base de dados do exemplo 3.

Ao analisar a matriz binária e matriz de consanguinidade é possível comprovar que o BRANGUSSELECTION atendeu todas as restrições impostas obtendo a solução ótima. O melhor touro foi utilizado apenas 1 vez, o segundo melhor touro foi utilizado 2 vezes e o terceiro melhor touro foi utilizado 3 vezes, conforme foi imposto. Os demais touros foram utilizados corretamente conforme seus limites de uso. A Figura 12 exibe a matriz de consanguinidade e a matriz binária concernentes ao presente exemplo.

A Tabela 15 apresenta a recomendação de acasalamento entregue pelo BRANGUSSELECTION utilizando os animais do exemplo. De acordo com os dados da tabela foram utilizados 7 touros para cobrir as 20 vacas. O valor da função objetivo correspondeu a 68,21513, que comparando-se com os valores atingidos nos exemplos 1 (205,4298) e 2 (275,6215), fica evidente a desvantagem da qualidade da prole dos animais do exemplo 3. Também, nota-se a necessidade de uso de 7 touros, ao invés de 5 touros

Figura 12 – Matriz de consanguinidade e matriz binária do exemplo 3.

```
> m_inbreeding
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]  [,11] [,12]  [,13]  [,14]  [,15]  [,16]  [,17]  [,18]  [,19]  [,20]
 [1,] 0.000  3.125  3.125 25.000 50.000 25.000  3.125  0.000  0.000  0.000  3.125 0.000  0.000 12.500  0.000  0.000  0.000 50.000  0.000 50.000
 [2,] 12.500 0.000  3.125  0.000  6.250  0.000  0.000 12.500  0.000 50.000  6.250 0.000  3.125 25.000  0.000  0.000  3.125 12.500  0.000  0.000
 [3,] 0.000  3.125  3.125  0.000  0.000 50.000  0.000  0.000 50.000  0.000 25.000 0.000  3.125  3.125  3.125  0.000  0.000 50.000  3.125 25.000
 [4,] 0.000 25.000 25.000  0.000 12.500  3.125  0.000  3.125  0.000  0.000  0.000 6.250  0.000 25.000  0.000  3.125 12.500  0.000 50.000  0.000
 [5,] 0.000 25.000  0.000  3.125  3.125 50.000  3.125 50.000  0.000  0.000  3.125 3.125  0.000 25.000  0.000  0.000 50.000  0.000  6.250  0.000
 [6,] 0.000 12.500  0.000  0.000  0.000  0.000  0.000  0.000  3.125  0.000  3.125 0.000  6.250  3.125 50.000 25.000  0.000  3.125 50.000  0.000
 [7,] 0.000  0.000 12.500 12.500 25.000  0.000  3.125 25.000  0.000  0.000  0.000 0.000  3.125  0.000  3.125  0.000  0.000 50.000 12.500  6.250
 [8,] 0.000 12.500  6.250  0.000  3.125  0.000  3.125  3.125  0.000  3.125  0.000 3.125  3.125  0.000  0.000 25.000  0.000  0.000  3.125  0.000
 [9,] 3.125 25.000  3.125  0.000  0.000  0.000  0.000  3.125  3.125  0.000  0.000 3.125 12.500  0.000  0.000  0.000 50.000  0.000  3.125  3.125
[10,] 0.000  0.000  3.125  3.125  0.000  3.125 50.000  0.000  0.000  0.000  6.250 0.000 12.500 12.500 25.000 25.000  3.125  0.000  0.000  3.125
> test$`Binary Solution`
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
 [1,]    0    0    1    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [2,]    0    0    0    0    0    1    0    0    0     0     0     0     0     0     0     0     0     0     0     1
 [3,]    1    1    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     1     0
 [4,]    0    0    0    0    0    0    0    1    0     0     0     0     0     0     1     1     0     1     0     0
 [5,]    0    0    0    1    1    0    1    0    0     0     1     0     0     0     0     0     0     0     0     0
 [6,]    0    0    0    0    0    0    0    0    1     1     0     1     0     1     0     0     0     0     0     0
 [7,]    0    0    0    0    0    0    0    0    0     0     0     0     1     0     0     0     1     0     0     0
 [8,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
 [9,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
[10,]    0    0    0    0    0    0    0    0    0     0     0     0     0     0     0     0     0     0     0     0
```

Fonte: Autor

Touro	**Vacas**			
L290	L989	L251		
L314	K330	M35		
L580	M216			
L687	L481	L694	L414	K519
L690	L847	K724	K412	K462
L938	L347	M217	L804	L966
M192	K370	K435	K495	
Valor função objetivo: 68,21513				

Tabela 15 – Recomendação de acasalamentos do exemplo 3.

como relatado nos exemplos supracitados, para poder efetivar os acasalamentos com todas as fêmeas. Isto se deve às restrições fixadas nos 3 melhores touros.

O exemplo 4 objetivou verificar se o BRANGUSSELECTION encontra a solução ótima ao definir um valor de uso mínimo para todos os touros, ilustrando uma situação onde um produtor deseja utilizar todos os touros selecionados por ele. Para a execução deste exemplo, foi utilizada a mesma amostra da base de dados e matriz de consanguinidade do exemplo 1, porém, aumentando-se a tolerância máxima do índice de consanguinidade para 12,5%, ou seja, permitindo acasalamentos em que os animais possuem até um 1/8 de parentesco, isto é, entre meio-irmãos, por exemplo. O uso máximo dos touros manteve-se em 4 e conforme o objetivo deste exemplo, foi estabelecido que todos os touros sejam usados pelo menos uma vez. Na Tabela 16 encontram-se os acasalamentos recomendados pertinentes ao exemplo 4. O valor da função objetivo foi 163,7071, sendo inferior ao exemplo 1, que alcançou o valor 205,4298. Conforme é possível verificar, todos os touros foram utilizados, atendendo às restrições de uso mínimo. Nota-se que apesar de ser ampliado o índice de consanguinidade, de 3,125%

Touro	Vacas			
L142	L203	K485		
L373	K369			
L413	L235	K606	M9	L663
L422	M71			
L594	I131			
L752	K339			
M11	L332			
M2	L896	K421	L954	L845
M31	L196	K567	M7	L561
M316	L719			
Valor função objetivo: 163,7071				

Tabela 16 – Recomendação de acasalamentos do exemplo 4.

para 12,5%, o fator que incidiu no resultado final foi a qualidade dos touros e a quantidade de vezes que foram usados. Ao impor o uso do pior touro, que possui um valor de contribuição individual de -16,016, pelo menos uma vez, reflete-se no valor econômico da futura geração dos animais, em base ao índice de seleção utilizado neste trabalho.

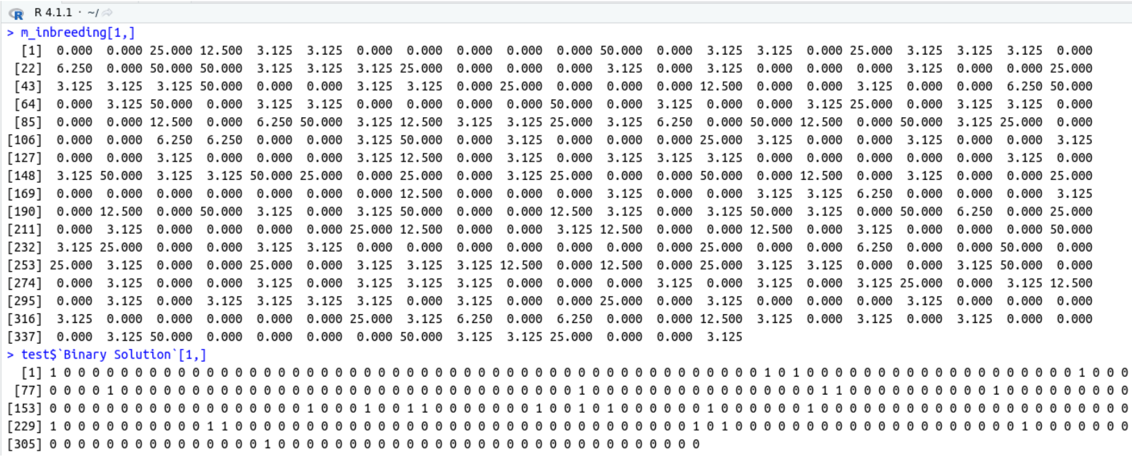
O exemplo 5 objetivou criar um cenário onde um produtor seleciona 15 touros/sêmen da base de dados para serem usados no seu rebanho na estação de monta em 350 vacas. O índice máximo de endogamia permitido foi ajustado para 0, ou seja, não poderão ocorrer acasalamentos entre animais com nenhum tipo de parentesco. O uso máximo e mínimo dos touros foi fixado em 25 e 5, respectivamente. Pare executar este exemplo foi gerada uma amostra da base de dados com 365 animais.

Devido ao grande número de animais utilizados, não serão apresentadas a matriz de consanguinidade e binária completas, sendo assim, a Figura 13 apresenta parte da matriz de consanguinidade e matriz binária. Especificamente, são mostrados os valores unicamente para o touro L210. Conforme é possível observar, os acasalamentos ocorreram 25 vezes com as fêmeas com índice de consanguinidade 0, atendendo assim as restrições do produtor.

A solução ótima foi encontrada satisfatoriamente com o valor da função objetivo, ou índice de prole, atingindo 1960,049. A Figura 14 apresenta uma amostra da listagem de acasalamentos gerada na saída do BRANGUSSELECTION também disponibilizada em formato de planilha eletrônica. Esta listagem designa o pareamento de animais recomendado para que seja obtido o maior índice possível nos filhos, que no caso do índice de seleção utilizado neste trabalho, a otimização proporcionará a maximização dos ganhos financeiros ao sistema de produção.

As saídas do algoritmo BRANGUSSELECTION são consistentes com os resultados teóricos apresentados e desenvolvidos em (FERREIRA; YOKOO; MOTTA, 2021).

Figura 13 – Matriz de consanguinidade e matriz binária do exemplo 5.

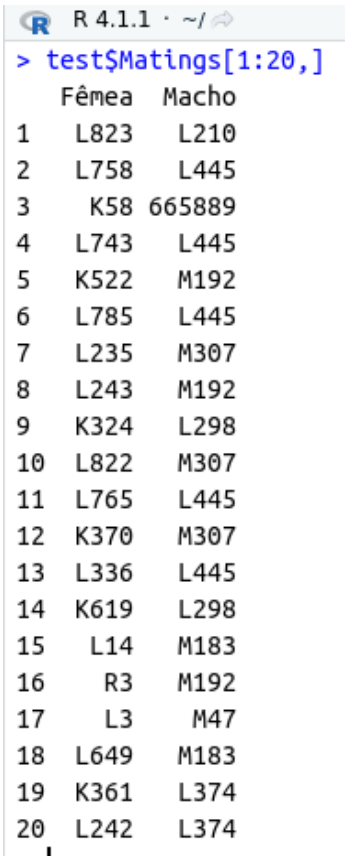


```
R 4.1.1 · ~/
> m_inbreeding[1,]
  [1]  0.000  0.000 25.000 12.500  3.125  3.125  0.000  0.000  0.000  0.000  0.000 50.000  0.000  3.125  3.125  0.000 25.000  3.125  3.125  3.125  0.000
 [22]  6.250  0.000 50.000 50.000  3.125  3.125  3.125 25.000  0.000  0.000  0.000  3.125  0.000  3.125  0.000  0.000  0.000  3.125  0.000  0.000 25.000
 [43]  3.125  3.125  3.125 50.000  0.000  0.000  3.125  3.125  0.000 25.000  0.000  0.000  0.000 12.500  0.000  0.000  3.125  0.000  0.000  6.250 50.000
 [64]  0.000  3.125 50.000  0.000  3.125  3.125  0.000  0.000  0.000  0.000 50.000  0.000  3.125  0.000  0.000  3.125 25.000  0.000  3.125  3.125  0.000
 [85]  0.000  0.000 12.500  0.000  6.250 50.000  3.125 12.500  3.125  3.125 25.000  3.125  6.250  0.000 50.000 12.500  0.000 50.000  3.125 25.000  0.000
[106]  0.000  0.000  6.250  6.250  0.000  0.000  3.125 50.000  0.000  3.125  0.000  0.000  0.000 25.000  3.125  0.000  0.000  3.125  0.000  0.000  3.125
[127]  0.000  0.000  3.125  0.000  0.000  0.000  3.125 12.500  0.000  3.125  0.000  3.125  3.125  3.125  0.000  0.000  0.000  0.000  0.000  3.125  0.000
[148]  3.125 50.000  3.125  3.125 50.000 25.000  0.000 25.000  0.000  3.125 25.000  0.000  0.000 50.000  0.000 12.500  0.000  3.125  0.000  0.000 25.000
[169]  0.000  0.000  0.000  0.000  0.000  0.000  0.000 12.500  0.000  0.000  0.000  3.125  0.000  0.000  3.125  3.125  6.250  0.000  0.000  0.000  3.125
[190]  0.000 12.500  0.000 50.000  3.125  0.000  3.125 50.000  0.000  0.000 12.500  3.125  0.000  3.125 50.000  3.125  0.000 50.000  6.250  0.000 25.000
[211]  0.000  3.125  0.000  0.000  0.000  0.000 25.000 12.500  0.000  0.000  3.125 12.500  0.000  0.000 12.500  0.000  3.125  0.000  0.000  0.000 50.000
[232]  3.125 25.000  0.000  0.000  3.125  3.125  0.000  0.000  0.000  0.000  0.000  0.000  0.000 25.000  0.000  0.000  6.250  0.000  0.000 50.000  0.000
[253] 25.000  3.125  0.000  0.000 25.000  0.000  3.125  3.125  3.125 12.500  0.000 12.500  0.000 25.000  3.125  3.125  0.000  0.000  3.125 50.000  0.000
[274]  0.000  3.125  0.000  0.000  3.125  0.000  3.125  3.125  3.125  0.000  0.000  0.000  3.125  0.000  3.125  0.000  3.125 25.000  0.000  3.125 12.500
[295]  0.000  3.125  0.000  3.125  3.125  3.125  3.125  0.000  3.125  0.000  0.000 25.000  0.000  3.125  0.000  0.000  0.000  3.125  0.000  0.000  0.000
[316]  3.125  0.000  0.000  0.000  0.000  0.000 25.000  3.125  6.250  0.000  6.250  0.000  0.000 12.500  3.125  0.000  3.125  0.000  3.125  0.000  0.000
[337]  0.000  3.125 50.000  0.000  0.000  0.000  0.000 50.000  3.125  3.125 25.000  0.000  0.000  3.125
> test$`Binary Solution`[1,]
  [1] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [77] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
[153] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[229] 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
[305] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Fonte: Autor

Nesse trabalho são estabelecidos e provados quais são os critérios para o problema ter uma solução. As provas apresentadas também delineiam qual é a melhor solução possível do conjunto de animais passados como parâmetro do algoritmo: os melhores touros são acasalados até o seu limite de uso possível, desde que não haja restrições de consanguinidade. Os resultados do algoritmo BRANGUSSELECTION apresentam exatamente esse resultado, mostrando que sua implementação está correta. Consequentemente, o algoritmo BRANGUSSELECTION implementa uma estratégia de acasalamento entre semelhantes: as melhores vacas cruzarão com os melhores touros, a menos que sejam aparentadas com eles acima do limite definido.

Figura 14 – Amostra da listagem dos acasalamentos do exemplo 5.



```
R 4.1.1 · ~/
> test$Matings[1:20,]
   Fêmea  Macho
1   L823   L210
2   L758   L445
3    K58 665889
4   L743   L445
5   K522   M192
6   L785   L445
7   L235   M307
8   L243   M192
9   K324   L298
10  L822   M307
11  L765   L445
12  K370   M307
13  L336   L445
14  K619   L298
15   L14   M183
16    R3   M192
17    L3    M47
18  L649   M183
19  K361   L374
20  L242   L374
```

Fonte: Autor

6 CONCLUSÃO

Esta dissertação apresentou o algoritmo BRANGUSSELECTION, com o objetivo de encontrar uma solução ótima de acasalamentos de bovinos da raça Brangus. O algoritmo faz a otimização com base em um índice de seleção, que neste trabalho foi utilizado o índice de seleção econômico, que objetiva o maior retorno financeiro em um sistema de produção. Entretanto, o algoritmo BRANGUSSELECTION foi criado para trabalhar com qualquer tipo de índice de seleção, podendo ser alteradas tanto as variáveis quanto suas respectivas ponderações. Também não é dependente de uma raça específica de bovinos, ou mesmo de gado bovino, podendo ser considerado um algoritmo genérico para qualquer espécie ou raça.

O algoritmo recebe como entrada, além do índice de seleção, informações provenientes de duas bases de dados: os valores individuais dos animais para cada caraterística de interesse (ou dos valores individuais de contribuição, já computados) e a matriz de coancestralidade (ou de consanguinidade) referente aos animais selecionados para reprodução. Adicionalmente, recebe como parâmetros o número máximo de vezes que cada macho pode ser usado no processo de acasalamento à frente e o índice máximo de consanguinidade tolerável, que pode ser qualquer valor maior ou igual zero.

O algoritmo de otimização da seleção de acasalamentos BRANGUSSELECTION resolve o problema por meio de três passos sucessivos: (i) modelagem do problema como um problema de programação inteira, (ii) solução do modelo de programação inteira relaxado para admitir soluções com valores reais e (iii) aplicação do algoritmo *branch-and-bound* para encontrar a melhor solução inteira dentro do espaço de soluções fornecido pelo algoritmo Simplex, que resolve o problema de programação linear relaxada. Um dos objetivos do uso desta estratégia foi investigar o tempo de execução do algoritmo. A programação inteira é um problema NP-completo, mesmo para valores binários. Se o problema subjacente é computacionalmente intratável, a busca de soluções inteiras dentro do espaço de estados da solução linear relaxada gera um processo de tamanho exponencial.

A partir de uma heurística que começou a ser desenvolvida paralelamente a este trabalho, os animais foram ordenados na matriz de contribuição em ordem decrescente de contribuição individual. Essa ordenação resultou em uma diminuição importante do tempo de execução do algoritmo já implementado, dando pistas de que o problema poderia ser resolvido em tempo polinomial. Os testes realizados neste trabalho

apontam para uma função polinomial cúbica na quantidade de animais, correspondente à complexidade do método Simplex. A execução do BRANGUSSELECTION em uma base com cerca de 1400 animais em um tempo ainda da ordem de minutos comprova a suposição. Ainda que a partir desse fato não se possa afirmar categoricamente que o problema é tratável, a partir dos testes feitos – que indicam uma complexidade média e podem esconder alguns casos em que o tempo necessário é maior, o exercício realizado constitui um bom indicativo. A ordenação dos animais também contribuiu para a diminuição do tempo de execução do algoritmo, reforçando a suspeita de solução ótima polinomial para todos os casos.

O algoritmo BRANGUSSELECTION está correntemente codificado na linguagem de programação R com uso de pacotes para otimização de laços iterativos por meio da implementação de código C++, de forma a melhorar o desempenho do algoritmo. Pacotes específicos para os algoritmos simplex e *branch-and-bound* também são usados. O índice de seleção econômico desenvolvido pela EMBRAPA Pecuária Sul foi usado para os testes, apresentando os resultados da aplicação do BRANGUSSELECTION em uma base real. Os resultados obtidos podem ser comparados com outros índices, para análises referentes ao processo de seleção. Note-se que, como a melhor solução possível da futura geração do rebanho está diretamente ligada ao índice usado, resultados do processo de acasalamento com outros índices não são comparáveis: somente podem ser comparados em relação ao processo de seleção, para escolha do melhor índice. De toda forma, o algoritmo proposto neste trabalho provê a solução ótima para qualquer índice usado. Se a mesma base for usada para índices diferentes, pode-se usá-lo para comparar os índices entre si.

Como trabalhos futuros, pode-se construir um pacote R que opere recebendo parâmetros estruturados e que seja disponibilizado. O algoritmo também pode ser disponibilizada por meio de uma API (*application programming interface*) para que outras aplicações façam uso da funcionalidade. Estratégias de comparação de índices também podem ser construídas a partir do algoritmo BRANGUSSELECTION. Finalmente, a investigação sobre outras estratégias de acasalamento, como o de acasalamentos compensatórios, cujo objetivo é, além de obter a solução máxima da contribuição, é produzir um rebanho mais homogêneo, ou seja, menos animais excepcionais e menos animais deficientes em relação aos objetivos de seleção definidos no processo de melhoramento genético. O desenvolvimento de interfaces web para o BRANGUSSELECTION em que o usuário possa interagir com facilidade e ter acesso em qualquer dispositivo, bem como consultar posteriormente os resultados obtidos, também

é uma possibilidade.

O algoritmo será disponibilizado para uso da EMBRAPA Pecuária Sul. Espera-se que este trabalho contribua com o melhoramento do potencial genético dos rebanhos do país, mantendo a competitividade no mercado internacional e auxiliando no suprimento de carne de forma mais produtiva.

REFERÊNCIAS

ABREU, U. G. P. d.; SONOHATA, M. M.; LOPES, P. S. Definição de pesos econômicos e de índices de seleção para sistemas de produção. In: ROSA, A. d. N. et al. (Ed.). **Melhoramento Genético Aplicado em Gado de Corte. Programa Geneplus-Embrapa**. Brasilia, DF: Embrapa Gado de Corte, 2013. v. 1, cap. 11, p. 256.

AGUIAR, M. S. d. Análise formal da complexidade de algoritmos genéticos. Universidade Federal do Rio Grande do Sul, 1998.

ARORA, S.; BARAK, B. **Computational Complexity: a Modern Approach**. Cambridge, UK: Cambridge University Press, 2009. 640 p. ISBN 978-0-521-42426-4.

BAGHEL, M.; AGRAWAL, S.; SILAKARI, S. Survey of metaheuristic algorithms for combinatorial optimization. **International Journal of Computer Applications**, Foundation of Computer Science, v. 58, n. 19, 2012.

BECCENERI, J. C. Meta-heurísticas e otimização combinatória: Aplicações em problemas ambientais. **INPE, Sao José dos Campos**, 2008.

BERKELAAR, M. et al. **lpSolve: Interface to 'Lp_solve' v. 5.5 to Solve Linear/Integer Programs**. 2020. R package version 5.6.15. Disponível em: https://CRAN.R-project.org/package=lpSolve.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

BOURDON, R. Shortcomings of current genetic evaluation systems. **Journal of animal science**, Oxford University Press, v. 76, n. 9, p. 2308–2323, 1998.

BROOKSHEAR, J. G. **Ciência da Computação-: Uma Visão Abrangente**. Porto Alegre: Bookman Editora, 2013.

CARDOSO, F. Ferramentas e estratégias para o melhoramento genético de bovinos de corte. **Embrapa Pecuária Sul-Documentos (INFOTECA-E)**, Bagé: Embrapa Pecuária Sul, 2009., 2009.

CARVALHEIRO, R.; QUEIROZ, S. A. d.; KINGHORN, B. Optimum contribution selection using differential evolution. **Revista Brasileira de Zootecnia**, SciELO Brasil, v. 39, n. 7, p. 1429–1436, 2010.

CARVALHO, T. et al. Um sistema de informação para melhoramento genético de caprinos e ovinos. In: SBC. **Anais Principais do XII Simpósio Brasileiro de Sistemas de Informação**. Florianópolis, 2016. p. 100–107.

CORMEN, T. H. et al. Algoritmos: teoria e prática. **Editora Campus**, v. 2, p. 296, 2002.

CUNHA, C. B. da; BONASSER, U. de O.; ABRAHÃO, F. T. M. Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. In: **XVI Congresso da Anpet**. [S.l.: s.n.], 2002.

DAGUSPTA, S.; PAPADIMITRIOU, C. H.; VAZIRANI, U. V. **Algorithms**. 1. ed. [S.l.]: McGraw-Hill Higher Education, 2006. 320 p. ISBN 978-0073523408.

EDDELBUETTEL, D.; BALAMUTA, J. J. Extending extitR with extitC++: A Brief Introduction to extitRcpp. **The American Statistician**, v. 72, n. 1, p. 28–36, 2018. Disponível em: https://doi.org/10.1080/00031305.2017.1375990.

ELER, J. P. **Teorias e métodos em melhoramento genético animal: bases do melhoramento genético animal**. Pirassununga – São Paulo: Faculdade de Zootecnia e Engenharia de Alimentos da USP, 2017. 239 p.

ELER, J. P. **Teorias e métodos em melhoramento genético animal: seleção**. Pirassununga – São Paulo: Faculdade de Zootecnia e Engenharia de Alimentos da USP, 2017. 177 p.

ELER, J. P. **Teorias e métodos em melhoramento genético animal: sistemas de acasalamento**. Pirassununga – São Paulo: Faculdade de Zootecnia e Engenharia de Alimentos da USP, 2017. 129 p.

FAO. Food and agriculture organization of the united nations (2009) how to feed the world 2050. In: DISCUSSION PAPER PREPARED FOR EXPERT FORUM: 12–13 OCTOBER. Rome, 2009.

FAO, IFAD, UNICEF, WFP and WHO. **The state of food security and nutrition in the world 2020: transforming food systems for affordable healthy diets**. Rome: Food & Agriculture Org., 2021. v. 2021.

FERREIRA, A. P. L.; YOKOO, M. J.-I.; MOTTA, B. T. On the problem of optimal mating in animal breeding. In: **Submetido**. [S.l.: s.n.], 2021.

FONTOURA, D. d. C. N. d. Uma solução de recomendações de acasalamentos baseada em algoritmos genéticos. Universidade Federal do Pampa, 2019.

FORMIGONI, I. B. **Estimação de valores econômicos para características componentes de índices de seleção em bovinos de corte.** Tese (Doutorado) — Universidade de São Paulo, 2002.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)**. First edition. [S.l.]: W. H. Freeman, 1979. ISBN 0716710455.

GIBSON, J.; WILTON, J. Defining multiple-trait objectives for sustainable genetic improvement. **Journal of animal science**, Oxford University Press, v. 76, n. 9, p. 2303–2307, 1998.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear: modelos e algoritmos**. Rio de Janeiro: Elsevier, 2005.

HILLIER, F. S.; LIEBERMAN, G. J. **Introdução à pesquisa operacional**. Porto Alegre: McGraw Hill Brasil, 2013.

IHAKA, R.; GENTLEMAN, R. R: a language for data analysis and graphics. **Journal of computational and graphical statistics**, Taylor & Francis Group, v. 5, n. 3, p. 299–314, 1996.

KIM, V. **lpsymphony: Symphony integer linear programming solver in R**. 2020. Http://R-Forge.R-project.org/projects/rsymphony, https://projects.coin-or.org/SYMPHONY, http://www.coin-or.org/download/source/SYMPHONY/.

KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. Keele, 2004.

LAKATOS, E. M.; MARCONI, M. d. A. Fundamentos de metodologia científica. 5. reimp. **São Paulo: Atlas**, v. 310, 2007.

LANDIM, V. et al. Diagnóstico da situação da resistência do carrapato boophilus microplus a carrapaticidas em bovinos de corte e leite na região de uberaba. **FAZU em Revista**, n. 03, 2006.

MAZZONETTO, A.; HÖLBIG, C. A. Computação de alto desempenho em r: paralelização e técnicas de otimização. 2014.

MELIÁN, B.; PÉREZ, J. A. M.; VEGA, J. M. M. Metaheurísticas: Una visión global. **Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial**, Asociación Española para la Inteligencia Artificial, v. 7, n. 19, 2003.

NETO, A. D. B. et al. Estrutura populacional e otimização de esquemas de acasalamento em ovinos com uso de algoritmos evolucionários. Universidade Federal de Sergipe, 2014.

NIETO, L. M.; ALENCAR, M. M. d.; ROSA, A. d. N. Critérios de seleção. In: ROSA, A. d. N. et al. (Ed.). **Melhoramento Genético Aplicado em Gado de Corte. Programa Geneplus-Embrapa**. Brasilia, DF: Embrapa Gado de Corte, 2013. v. 1, cap. 10, p. 256.

OLIVEIRA, I. C. et al. Complexidade computacional e o problema p vs np. [sn], 2010.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. **Combinatorial optimization: algorithms and complexity**. Mineola, NY: Dover Publications, 1998.

ROSA, A. d. N.; MENEZES, G. R. d. O.; EGITO, A. A. d. Recursos genéticos e estratégias de melhoramento. In: ROSA, A. d. N. et al. (Ed.). **Melhoramento Genético Aplicado em Gado de Corte. Programa Geneplus-Embrapa**. Brasilia, DF: Embrapa Gado de Corte, 2013. v. 1, cap. 2, p. 256.

SILVA, E. V. C. e et al. Ambiência e comportamento no manejo reprodutivo. In: ROSA, A. d. N. et al. (Ed.). **Melhoramento Genético Aplicado em Gado de Corte. Programa Geneplus-Embrapa**. Brasilia, DF: Embrapa Gado de Corte, 2013. v. 1, cap. 3, p. 256.

SIMÕES, M. R. S. et al. Breeding objectives of brangus cattle in brazil. **Journal of Animal Breeding and Genetics**, v. 137, n. 2, p. 177–188, 2020. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/jbg.12415.

SIPSER, M. **Introdução à teoria da computação**. São Paulo: Thomson Learning, 2007.

TILLÉ, Y.; MATEI, A. **sampling: Survey Sampling**. 2021. R package version 2.9. Disponível em: https://CRAN.R-project.org/package=sampling.

TOSCANI, L. V.; VELOSO, P. A. **Complexidade de algoritmos: análise, projeto e métodos**. Porto Alegre: Sagra-Luzzatto, 2012.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. Rio de Janeiro: Elsevier, 2009.

WICKHAM, H. **ggplot2: Elegant Graphics for Data Analysis**. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. Disponível em: https://ggplot2.tidyverse.org.

WOLFOVÁ, M. et al. Breeding objectives for beef cattle used in different production systems: 2. model application to production systems with the charolais breed. **Livestock Production Science**, Elsevier, v. 95, n. 3, p. 217–230, 2005.

YANG, X.-S. **Nature-inspired metaheuristic algorithms**. Cambridge, UK: Luniver press, 2010.

APÊNDICE A – CÓDIGO-FONTE DOS PROGRAMAS DESENVOLVIDOS

```
#Algoritmo Forca Bruta. Parte 1.
#Funcao para permutar o vetor v.

permuta <- function(v, n, nf, nm, vmaxuse){
#Entradas: vetor dos machos, variavel de controle
#de permutacao de v, numero de femeas, numero de machos
#e vetor com o uso maximo de cada touro.

  if (n > nf) {
    res[k] <<- calc(p, m, nm, vmaxuse)
    k <<- k +1
  } else {
    for (i in 1:length(v)) {
      p[n] <<- v[i]
      permuta(v, n+1, nf, nm, vmaxuse)
    }
  }
  return(res[which.max(res)])
}

#Algoritmo Forca Bruta. Parte 2.
#Funcao para calcular o valor da funcao objetivo.


calc <- function(perm, m, nmal, v_max_use) {
#Entradas: vetor permutado, matriz de contribuicao,
#numero de machos e vetor com o uso maximo de cada touro.

  sum <- 0
  for (i in 1:length(perm)) {
    if (v_max_use[perm[i]] > 0) {
      v_max_use[perm[i]] <- v_max_use[perm[i]] - 1
      sum <- sum + m[perm[i],i]
    } else {
      sum <- sum + MININT
    }
  }

  if (sum > best_value){
    best_value <<- sum
    best_solution <<- perm
    #output_bf <- list(best_value, best_solution)
    #return(output_bf)
    return(best_value)
  } else {
    return(MININT)
  }
```

```
}


#Funcao para calcular a contribuicao individual de cada animal

f_individual_contribution = function(db, s_index, inicolDB, endcolDB){
#Entradas: Base de dados, indice de selecao, valor inicial e
#final das colunas da base de dados pertencentes as DEP.

  v = 0 #Vetor de armazenamento dos valores da contribuicao de cada animal
  for (i in 1:nrow(db)){
    individual_contribution = 0 #Variavel da contribuicao individual
    k = 0 #Variavel de controle do indice do s_index
    for (j in inicolDB:endcolDB) {
      k = k+1

      #Calculo da contribuicao individual
      individual_contribution <- (db[i,j] * s_index[k])
      + individual_contribution

    }
    v[i] <-individual_contribution
  }
  db <- cbind(db,v)
  return (db) #Saida: base de dados com os valores calculados
}


#Funcao para calcular a contribuicao dos pares e criar
#a matriz de contribuicao

f_contribution_matrix = function(db, m_inbreed, max){
#Entradas: Base de dados, matriz de consanguinidade
#e valor maximo de endogamia

  females <- subset(db, Sexo == "Femea")
  males <- subset(db, Sexo == "Macho")

  #Ordenacao da base de dados pelo valor da contribuicao individual
  females <- females[order(females$v, decreasing = TRUE),]
  males <- males[order(males$v, decreasing = TRUE),]

  #Criacao de id para cada animal
  n_females <- nrow(females)
  n_males <- nrow(males)
  id <- 1:n_females
  females <- cbind(females, id)
  id <- 1:n_males
  males <- cbind(males, id)
```

```
  v = 0 #Vetor para armazenar a contribuicao dos pares
  k = 0 #Variavel de controle de v

  #Criacao de constante negativa
  MININT = as.integer((.Machine$integer.max)*-1)
  v_column<- which(colnames(db)=="v") #Verifica o numero da coluna v

  for (i in 1:n_males){
    for (j in 1:n_females) {
      k = k+1
      if (m_inbreed[i,j] <= max) { #Teste do valor maximo de endogamia

        #Calculo e atribuicao da contribuicao do par

        #A variavel v_column representa o valor da coluna
        #da contribuicao individual do animal (v)
        v[k] <- (males[i,v_column] + females[j,v_column])/2
      } else {
        #Descarta a contribuicao do par
        v[k] <- MININT
      }

    }
  }

  m <- matrix(v, n_males, n_females, byrow = TRUE)

  output <- list(m, males, females)

  return(output)
  #Saidas: Lista com matriz de contribuicao, dataframe
  #dos machos e das femeas

}


#Funcao para criar a matriz de coeficientes das restricoes
#das femeas utilizando o pacote Rcpp codificado em C++

library(Rcpp)
cppFunction(
  "NumericMatrix f_coefficients_restrictions_females_c(NumericMatrix m) {

    int nfemales = m.ncol();
    int nmales = m.nrow();
    NumericMatrix coef(nfemales, nfemales * nmales);

    for (int female = 0; female < m.ncol(); female++){
      int k = 0;
      NumericVector v(nfemales * nmales);
```

```
      for (int i = 0; i < m.nrow(); i++) {
        for (int j = 0; j < m.ncol(); j++) {

          if (j == female){
            v[k] = 1;
          } else {
            v[k] = 0;
          }

          k = k+1;

        }
      }

    coef(female,_) = v;
    }

    return(coef);
  }

  "
)


#Funcao para criar a matriz de coeficientes das restricoes
#dos machos utilizando o pacote Rcpp codificado em C++

library(Rcpp)
cppFunction(
  "NumericMatrix f_coefficients_restrictions_males_c(NumericMatrix m) {

    int nfemales = m.ncol();
    int nmales = m.nrow();
    NumericMatrix coef(nmales, nfemales * nmales);

    for (int male = 0; male < nmales; male++){
      int k = 0;
      NumericVector v(nfemales * nmales);

      for (int i = 0; i < nmales; i++) {
        for (int j = 0; j < nfemales; j++) {

          if (i == male){
            v[k] = 1;
          } else {
            v[k] = 0;
          }

          k = k+1;
```

```
        }
      }

    coef(male,_) = v;
    }

    return(coef);
  }

  "
)

#Funcao para selecionar os primeiros animais (ordenados pelo IECC da base)
#de acordo com o numero definido nas variaveis nmales e nfemales

f_first_database = function(db, n_males, n_females){
  males <- subset(dbase, Sexo == "Macho")
  males <- males[1:n_males,]
  females <- subset(dbase, Sexo == "Femea")
  females <- females[1:n_females,]
  dbase <- rbind(females, males)
  return(dbase)
}

#Funcao para criar uma amostra da base de dados com um determinado
#numero de machos e femeas

f_sample_database = function(db, n_males, n_females){
#Entrada: base de dados, numero de machos e femeas

  library(sampling)
  sample_db = strata(db, c("Sexo"), size = c(n_females, n_males),
  method = "srswor") # srswor = sem reposicao e srswr = com reposicao
  sample_db <- db[sample_db$ID_unit,]
  return(sample_db) # Saida: amostra da base de dados
}


#Funcao para contabilizar o uso de cada touro

f_male_count <- function(solution){ #Entrada: Matriz binaria
  v = 0
  for (i in 1:dim(m)[1]) {
    counter = 0
    for (j in 1:dim(m)[2]){
      if (solution[i,j] == 1){
        counter = counter + 1
      }
      v[i] = counter
```

```
    }
  }
  return(v)
  #Saida: vetor com o numero de uso de cada macho.
  #Cada posicao do vetor corresponde a um macho.
}

#Funcao para mostrar a recomendacao de acasalamentos

f_mating <- function(solution, dbfemales, dbmales){
#Entrada: Matriz binaria, dataframe das femeas e dos machos.

  nfemales = dim(solution)[2]
  nmales = dim(solution)[1]

  f = 0
  m = 0

  for (i in 1:nfemales) {
    for (j in 1:nmales) {
      if (solution[j, i] == 1) {
        f[i] <- dbfemales$Animal[i]
        m[i] <- dbmales$Animal[j]
      }
    }
  }
  s <- data.frame(f,m)
  colnames(s) <- c("Femea", "Macho")
  return(s)
  #Saida: Dataframe com as recomendacoes de acasalamentos.
}

#Funcao para atribuir aleatoriamente valores maximos
#de uso de cada touro

f_max_use_males <- function(nmin, nmax){
# Entrada: valor minimo e maximo do intervalo.
  values = sample(nmin:nmax, size = nmales, replace = TRUE)
  m <- matrix(values, nrow = 1, ncol = nmales)
  #conversao do vetor values em uma matriz m para compatibilidade
  #com o pacote lpSolve

  return(m)
}

# Funcao para criar uma matriz de consanguinidade aleatoria
f_inbreeding_matrix <- function(nfemales, nmales,
vprob = c(0.0625, 0.0625, 0.0625, 0.0625, 0.25, 0.50)){
#Entradas: Numero de femeas, numero de machos,
#vetor de probabilidades
```

```
  values<- sample(c(0.5 * 100, 0.25 * 100, 0.125 * 100,
  0.0625 * 100, 0.03125 * 100, 0), nfemales * nmales,
  replace = TRUE, vprob)

  m <- matrix(values, nrow = nmales, ncol = nfemales)
  return(m)
}


#Exemplo de utilizacao do modelo

#Importacao e preparacao da base de dados
base <- read.csv("Documentos/scripts-R/Brangus_Selection/Base_Brangus.csv",
sep = ",", header = TRUE)

dbase <- base
dbase$Animal <- as.character(dbase$Animal)
dbase$Sexo <- as.factor(dbase$Sexo)

#Definicao do inidice de selecao.
selection_index <- c(116.0879975, -9.1900268, 23.8195584, 6.0030468,
-0.5495934, 0.1866388, 131.3764082, 65.8745087)

#Chamada da funcao para criar a matriz de consanguinidade.
m_inbreeding <- f_inbreeding_matrix(nfemales,nmales, 0.1, 3)

#Chamada da funcao para calcular a contribuicao individual.
dbase <- f_individual_contribution(dbase, selection_index, 3, 10)

#Chamada da funcao para calcular a matriz de contribuicao
output <- f_contribution_matrix(dbase, m_inbreeding, 4)
m <- output[[1]]

#Coeficientes da funcao objetivo
f.obj <- as.vector(t(m))

#Matriz dos coeficientes das restricoes
crm <- f_coefficients_restrictions_males_c(m) #machos
crf <- f_coefficients_restrictions_females_c(m) #femeas
f.con <- rbind(crm,crf)

#Atribuicao dos sinais de igualdade/desigualdade
#correspondentes as restricoes

#Para os machos o sinal deve ser menor ou igual (cada macho pode
#acasalar <= numero de acasalamentos definido pelo produtor).
f.dir.males_lpsolve <- matrix("<=", nrow = 1, ncol = nmales)

#Para as femeas o sinal deve ser = (cada femea deve acasalar = 1)
```

```
f.dir.females_lpsolve <- matrix("=", nrow = 1, ncol = nfemales)

f.dir_lpsolve <- cbind(f.dir.males_lpsolve, f.dir.females_lpsolve)

#Definicao do numero maximo de vezes que um touro pode acasalar.
#Neste caso todos acasalam ate 30 vezes.
f.rhs.males <- matrix(30, nrow = 1, ncol = nmales)

#Cada vaca so pode acasalar uma unica vez por solucao.
f.rhs.females <- matrix(1, nrow = 1, ncol = nfemales)

f.rhs <- cbind(f.rhs.males, f.rhs.females)

#Limpa memoria.
rm(crm, crf, f.dir.males, f.dir.females, f.rhs.males, f.rhs.females)
gc()

#Carrega pacote
library(lpSolve)

#Executa a funcao lp com todos os parametros configurados.
Z_ipSolve <- lp("max", f.obj, f.con, f.dir_lpsolve, f.rhs, all.bin = TRUE)
Z_ipSolve
Z_ipSolve$solution

#Exibicao do uso de cada touro
male_usage_ipSolve <- f_male_count(solution_ipSolve)
male_usage_ipSolve

#Mostra recomendacao de acasalamentos. A primeira coluna correspondem
#as femeas e a segunda aos machos.
dbmales <- output[[2]]
dbfemales <- output[[3]]
acasalamentos_ipSolve <- f_mating(solution_ipSolve, dbfemales, dbmales)
acasalamentos_ipSolve #Mostra todos os acasalamentos

#Mostra os nomes dos machos com as respectivas quantidades de uso.
table(acasalamentos_ipSolve$Macho)

Teste do algoritmo de forca bruta.

Results <- 0
times_exc_BF <- 0
Z <- 0

rexc <- 10
num_males <-   c(2, 2, 2, 2, 3, 3, 3, 4, 4)
num_females <- c(3, 4, 5, 6, 6, 7, 8, 8, 9)
v_use_max_males <- c(2, 3, 3, 3, 3, 3, 3, 3, 3)
exc <- length(num_males) # Verifica a quantidade de instancias do teste.
```

```
for (e in 1:exc){

  for (r in 1:rexc){

    time_exc <- system.time({
      MININT <- as.integer((.Machine$integer.max)*-1)
      selection_index <- c(116.0879975, -9.1900268, 23.8195584,
      6.0030468, -0.5495934, 0.1866388, 131.3764082, 65.8745087)
      # Numero de machos.
      nmales <- num_males[e]
      # Numero de femeas.
      nfemales <- num_females[e]
      # Maximo de uso de cada touro.
      vmaxuse <- rep(v_use_max_males[e], nmales)
      # Armazena configuracoes dos acasalamentos.
      p <- 0
      # Armazena valor da funcao objetivo.
      best_value <- MININT
      # Armazena a melhor configuracao de acasalamentos.
      best_solution <- 0
      # Armazena saidas da funcao calc.
      results <- 0
      # Controla o indice do vetor results.
      k <- 0
      # Reset da base de dados.
      dbase <- base
      # Gera amostra da BD.
      dbase <- f_sample_database(dbase, nmales, nfemales)
      # Matriz de consanguinidade.
      m_inbreeding <- f_inbreeding_matrix(nfemales, nmales)
      # Calculo da contribuicao individual.
      dbase <- f_individual_contribution(dbase, selection_index, 3, 10)
      # Calculo da matriz de contribuicao.
      output <- f_contribution_matrix(dbase, m_inbreeding, 4)
      # Gera matriz de contribuicao.
      m <- output[[1]]

      #Entradas: vetor machos, controle de permutacao, numero de femeas.
      Z <- permutation(1:nmales, 1, nfemales)

      #Saidas do console.
      cat("\n ############################")
      cat("\n EXECUCAO: ", e)
      cat("\n Repeticao: ", r, "de ", rexc)
      cat("\n Numero de machos: ", nmales)
      cat("\n Numero de femeas: ", nfemales)
      cat("\n Total de animais: ", nmales + nfemales)
      cat("\n Valor otimo (Z): ", Z)
```

```
      cat("\n Base de dados: \n")
      print(dbase)
      cat("\n Matriz consanguinidade: \n")
      print(m_inbreeding)
      cat("\n Matriz contribuicao acasalamentos: \n")
      print(m)
      cat("\n ############################\n")

    })
    cat("\n TEMPO DE EXECUCAO: ", time_exc[3], "\n")
    cat("\n _____________________________________ \n")
    cat("\n _____________________________________ \n")
    times_exc_BF[r] <- time_exc[3]
  }

  mean_time_rep <- mean(times_exc_BF) # Media.

  variance_time_rep <-var(times_exc_BF) # Variancia.

  sd_time_rep <- sd(times_exc_BF) # Desvio padrao.

  results_this_rep <- list(times_exc_BF,
  mean_time_rep, variance_time_rep, sd_time_rep)

  names(results_this_rep) <- c("Times of executions",
  "Mean", "Variance", "Standard Deviation")

  # Gera uma lista com todos os resultados.
  Results[e] <- list(results_this_rep)

  # Saidas do console.
  cat("\n Tempos de execucao: \n")
  print(times_exc_BF)
  cat("\n Media: \n")
  print(mean_time_rep)
  cat("\n Variancia: \n")
  print(variance_time_rep)
  cat("\n Desvio padrao: \n")
  print(sd_time_rep)

}

Results

# Criacao do dataframe Experiment_Results_Dataframe
Experiment_Results <- 0
for (i in 1:length(Results)) {
  unlisted_time <- unlist(Results[[i]][1])
  unlisted_mean <- unlist(Results[[i]][2])
```

```
  unlisted_var <- unlist(Results[[i]][3])
  unlisted_sd <- unlist(Results[[i]][4])
  Experiment_Results[i] <- data.frame(c(round(unlisted_time, 4),
  round(unlisted_mean,4), round(unlisted_var,4), round(unlisted_sd,4)))
}
Experiment_Results


Experiment_Results_Dataframe  <-  as.data.frame(matrix(unlist(Experiment_Results),
nrow=length(Experiment_Results), byrow = T))

total_animals <- num_males + num_females

Experiment_Results_Dataframe <- cbind(Experiment_Results_Dataframe,
num_males, num_females, total_animals)

colnames(Experiment_Results_Dataframe)[rexc + 1] <- c("Mean")
colnames(Experiment_Results_Dataframe)[rexc + 2] <- c("Variance")
colnames(Experiment_Results_Dataframe)[rexc + 3] <- c("S.Deviation")
colnames(Experiment_Results_Dataframe)[rexc + 4] <- c("Males")
colnames(Experiment_Results_Dataframe)[rexc + 5] <- c("Females")
colnames(Experiment_Results_Dataframe)[rexc + 6] <- c("T. Animals")

Experiment_Results_Dataframe

# Arquivo
write.csv(Experiment_Results_Dataframe,
"Experiment_Results_Dataframe3.csv")

# Grafico
dataframe_grafico <- as.data.frame(cbind(total_animals,
Experiment_Results_Dataframe$Mean))

colnames(dataframe_grafico) <- c("animals", "means")

library(ggplot2)
g1<-ggplot(data = dataframe_grafico, aes(x = animals, y = means)) +
  labs(x = "Numero de animais", y = "Tempo de execucao (segundos)") +
  geom_point() + geom_line(size=0.5, color="red") +
  theme_bw() + scale_x_continuous(breaks = dataframe_grafico$animals)

g1

#Teste do algoritmo BrangusSelection

Results <- 0
times_exc_BS <- 0
Z <- 0

rexc <- 10
```

```
num_males <-   c(511,511,511,511,511,511,511,511)
num_females <- c(100,200,300,400,500,600,700,800)
v_use_max_males <- c(1,1,1,1,2,2,2,2)

exc <- length(num_males) # Verifica a quantidade iteracoes do teste.

for (e in 1:exc){

  for (r in 1:rexc){

    time_exc <- system.time({
      MININT <- as.integer((.Machine$integer.max)*-1)
      selection_index <- c(116.0879975, -9.1900268, 23.8195584,
      6.0030468, -0.5495934, 0.1866388, 131.3764082, 65.8745087)
      nmales <- num_males[e]
      nfemales <- num_females[e]
      vmaxuse <- v_use_max_males[e]

      # Reset da base de dados.
      dbase <- base
      # Gera amostra aleatoria da base de dados.
      dbase <- f_sample_database(dbase, nmales, nfemales)
      # Matriz de consanguinidade.
      m_inbreeding <- f_inbreeding_matrix(nfemales, nmales)
      # Calculo da contribuicao individual.
      dbase <- f_individual_contribution(dbase, selection_index, 3, 10)
      # Calculo da matriz de contribuicao.
      output <- f_contribution_matrix(dbase, m_inbreeding, 4)
      m <- output[[1]] # Gera matriz de contribuicao m.

      f.obj <- as.vector(t(m))
      crm <- f_coefficients_restrictions_males_c(m)
      crf <- f_coefficients_restrictions_females_c(m)
      f.con <- rbind(crm,crf)

      f.dir.males <- matrix("<=", nrow = 1, ncol = nmales)
      f.dir.females <- matrix("==", nrow = 1, ncol = nfemales)
      f.dir <- cbind(f.dir.males, f.dir.females)

      f.rhs.males <- matrix(vmaxuse, nrow = 1, ncol = nmales)
      f.rhs.females <- matrix(1, nrow = 1, ncol = nfemales)
      f.rhs <- cbind(f.rhs.males, f.rhs.females)

      rm(crm, crf, f.dir.males, f.dir.females, f.rhs.males, f.rhs.females)
      gc()

      f_solver <- function(pack){
        if (pack == 1) {
          library(lpSolve)
```

```
  Z_ipSolve <- lp("max", f.obj, f.con, f.dir_lpsolve,
  f.rhs, all.bin = TRUE)

  objval <- Z_ipSolve$objval

  solution_ipSolve <- matrix(Z_ipSolve$solution,
  nrow = nmales, ncol = nfemales, byrow = TRUE)

  solution <- solution_ipSolve

  males_usage_ipSolve <- f_male_count(solution_ipSolve)
  males_usage <- males_usage_ipSolve

  dbmales <- output[[2]]
  dbfemales <- output[[3]]

  matings_ipSolve <- f_mating(solution_ipSolve,
  dbfemales, dbmales)

  matings <- matings_ipSolve

  males_used_ipSolve <- table(matings_ipSolve$Macho)
  males_used <- males_used_ipSolve

} else if (pack == 2) {

  library(BiocManager)
  library(lpsymphony)

  Z_lpsymphony <- lpsymphony::lpsymphony_solve_LP(f.obj,
  f.con, f.dir, f.rhs, max = TRUE, types = "B")

  objval <- Z_lpsymphony$objval

  solution_lpsymphony <- matrix(Z_lpsymphony$solution,
  nrow = nmales, ncol = nfemales, byrow = TRUE)

  solution <- solution_lpsymphony

  males_usage_lpsymphony <- f_male_count(solution)
  males_usage <- males_usage_lpsymphony

  dbmales <- output[[2]]
  dbfemales <- output[[3]]
  matings_lpsymphony <- f_mating(solution, dbfemales, dbmales)
  matings <- matings_lpsymphony

  males_used_lpsymphony <- table(matings_lpsymphony$Macho)
```

```
        males_used <- males_used_lpsymphony

      } else {
        cat("Incorrect option. Please verify your entry.")
      }

      output <- list("Objective Value" = objval,
      "Binary Solution" = solution, "Male Count" = males_usage,
      "Matings" = matings, "Sires" = males_used)

      return(output)
    }

    # 1: lpSolve (Default)
    # 2: lpsymphony
    Z <- f_solver(1)


    #Saidas do console.
    cat("\n ############################")
    cat("\n EXECUCAO: ", e)
    cat("\n Repeticao: ", r, "de ", rexc)
    cat("\n Numero de machos: ", nmales)
    cat("\n Numero de femeas: ", nfemales)
    cat("\n Total de animais: ", nmales + nfemales)
    cat("\n Valor otimo (Z): ", Z$`Objective Value`)
    cat("\n Base de dados: \n")
    print(dbase)
    cat("\n Matriz consanguinidade: \n")
    print(m_inbreeding)
    cat("\n Matriz contribuicao acasalamentos: \n")
    print(m)
    cat("\n ############################\n")

  })
  cat("\n TEMPO DE EXECUCAO: ", time_exc[3], "\n")
  cat("\n _____________________________________________ \n")
  cat("\n _____________________________________________ \n")
  times_exc_BS[r] <- time_exc[3]
}

mean_time_rep <- mean(times_exc_BS) # Media.
variance_time_rep <-var(times_exc_BS) # Variancia.
sd_time_rep <- sd(times_exc_BS) # Desvio padrao.

results_this_rep <- list(times_exc_BS, mean_time_rep,
variance_time_rep, sd_time_rep)

names(results_this_rep) <- c("Times of executions",
```

```
  "Mean", "Variance", "Standard Deviation")

  # Gera uma lista com todos os resultados.
  Results[e] <- list(results_this_rep)

  # Saidas do console.
  cat("\n Tempos de execucao: \n")
  print(times_exc_BS)
  cat("\n Media: \n")
  print(mean_time_rep)
  cat("\n Variancia: \n")
  print(variance_time_rep)
  cat("\n Desvio padrao: \n")
  print(sd_time_rep)
}

Results

# Criacao do dataframe Experiment_Results_Dataframe

Experiment_Results <- 0

for (i in 1:length(Results)) {

  unlisted_time <- unlist(Results[[i]][1])
  unlisted_mean <- unlist(Results[[i]][2])
  unlisted_var <- unlist(Results[[i]][3])
  unlisted_sd <- unlist(Results[[i]][4])

  Experiment_Results[i] <- data.frame(c(round(unlisted_time, 4),
  round(unlisted_mean,4), round(unlisted_var,4), round(unlisted_sd,4)))
}

Experiment_Results

Experiment_Results_Dataframe  <-  as.data.frame(matrix(unlist(Experiment_Results),
nrow=length(Experiment_Results), byrow = T))

total_animals <- num_males + num_females

Experiment_Results_Dataframe <- cbind(Experiment_Results_Dataframe,
num_males, num_females, total_animals)

colnames(Experiment_Results_Dataframe)[rexc + 1] <- c("Mean")
colnames(Experiment_Results_Dataframe)[rexc + 2] <- c("Variance")
colnames(Experiment_Results_Dataframe)[rexc + 3] <- c("S.Deviation")
colnames(Experiment_Results_Dataframe)[rexc + 4] <- c("Males")
colnames(Experiment_Results_Dataframe)[rexc + 5] <- c("Females")
colnames(Experiment_Results_Dataframe)[rexc + 6] <- c("T. Animals")
```

```
Experiment_Results_Dataframe

# Arquivo
write.csv(Experiment_Results_Dataframe, "Experiment_Results.csv")

#Grafico
df_g <- as.data.frame(cbind(Experiment_Results_Dataframe$`T. Animals`,
Experiment_Results_Dataframe$Mean))

colnames(df_g) <- c("animals", "means")

library(ggplot2)
g<-ggplot(data = df_g, aes(x = animals, y = means)) +
  labs(x = "Numero de animais", y = "Tempo de execucao (segundos)") +
  geom_point() + geom_line(size=0.5, color="red") + theme_bw() +
  scale_x_continuous(breaks = df_g$animals)
  + scale_y_continuous(breaks = c(100,200,300,400,500,600))

print(g + theme(plot.title = element_text(hjust = 0),
axis.title = element_text(size = 13), axis.text = element_text(size = 12)))
```