UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

TRISS-AG: um algoritmo genético para ajuste do método TRISS de previsão de sobrevivência de pacientes de trauma

André Terra Ennes

Dissertação de Mestrado do Programa de Mestrado Profissional em
Matemática, Estatística e Computação Aplicadas à Indústria (MECAI)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

André Terra Ennes

TRISS-AG: um algoritmo genético para ajuste do método TRISS de previsão de sobrevivência de pacientes de trauma

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre – Mestrado Profissional em Matemática,
Estatística e Computação Aplicadas à Indústria.
VERSÃO REVISADA

Área de Concentração: Matemática, Estatística e
Computação

Orientador: Prof. Dr. José Alberto Cuminato

USP – São Carlos
Outubro de 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

E59t Ennes, André Terra
 TRISS-AG: um algoritmo genético para ajuste do
 método TRISS de previsão de sobrevivência de
 pacientes de trauma / André Terra Ennes; orientador
 José Alberto Cuminato. -- São Carlos, 2021.
 107 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
 em Mestrado Profissional em Matemática, Estatística
 e Computação Aplicadas à Indústria) -- Instituto de
 Ciências Matemáticas e de Computação, Universidade
 de São Paulo, 2021.

 1. Algoritmos genéticos. 2. Computação evolutiva.
 3. Índices de gravidade do trauma. 4. Trauma and
 Injury Severity Score. 5. TRISS. I. Cuminato, José
 Alberto, orient. II. Título.

André Terra Ennes

**TRISS-AG: a genetic algorithm to adjust the TRISS
prediction method of trauma patients survival**

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Professional Master's Program in Mathematics Statistics and Computing Applied to Industry, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Mathematics, Statistics and Computing

Advisor: Prof. Dr. José Alberto Cuminato

**USP – São Carlos
October 2021**

*Dedico este trabalho aos meus pais, **in memoriam**, Alberto Ennes e Martilha Terra,
que renunciaram a muito em suas vidas para dedicá-las aos filhos
e que, mesmo com uma origem simples e de pouco estudo,
nos deixaram um legado de muita sabedoria, honra e amor.
Que Deus os tenha a Seu lado e me permita ser para eles motivo de orgulho.*

AGRADECIMENTOS

Meus sinceros agradecimentos ao Prof. Dr. José Alberto Cuminato, pelo apoio e pela confiança no desenvolvimento deste trabalho. Agradeço novamente ao mesmo e também ao Prof. Dr. Afonso Paiva Neto, ao Prof. Dr. Rodrigo Fernandes de Mello e aos demais professores do ICMC, pela excelência no ensino e pela contribuição valiosa para o meu aprendizado.

Agradeço também ao Prof. Dr. Eduardo do Valle Simões pela enorme colaboração no desenvolvimento deste trabalho. Apesar de não constar formalmente como orientador ou coorientador perante o programa, sua participação como mentor foi fundamental durante toda a pesquisa, sendo sua inclusive a concepção da ideia. Muito obrigado pelo apoio sempre muito próximo, disponível e amigável.

Estendo meus agradecimentos ao Breno Cunha Queiroz, aluno do ICMC, pela valiosa ajuda na melhoria do desempenho e na solução de problemas nos códigos-fonte.

Manifesto, finalmente, meus agradecimentos especiais à Sucocítrico Cutrale Ltda. e ao Sr. José Luis Cervato, diretor financeiro da empresa e meu superior durante o período em que fui colaborador na mesma, por me permitir e me incentivar a cursar o mestrado e perseguir este importante passo da minha formação.

*“O grande livro da natureza está escrito
na linguagem da matemática.”
(Galileu Galilei)*

RESUMO

ENNES, A. T. **TRISS-AG: um algoritmo genético para ajuste do método TRISS de previsão de sobrevivência de pacientes de trauma.** 2021. 107 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

O método TRISS é amplamente utilizado para previsão de sobrevivência de pacientes de trauma, tendo papel chave tanto para guiar decisões clínicas no momento do atendimento como posteriormente para avaliar o desempenho do serviço de saúde. Embora seja o mais usado no mundo todo, ele tem sido criticado por não refletir a realidade do atendimento médico de países em desenvolvimento, pois foi construído por meio de uma regressão logística usando dados de pacientes norte-americanos há mais de três décadas. Estudos tem sido feitos buscando adaptá-lo usando dados locais e técnicas alternativas, incluindo inteligência computacional. Propõe-se neste trabalho uma abordagem para ajustar o TRISS usando algoritmos genéticos, testando se é possível utilizar tal técnica e superar o desempenho do TRISS original quando aplicado a dados que representem tal realidade. Inicialmente foram sintetizados cinco bancos de dados simulando pacientes atendidos por hospitais de qualidade progressivamente inferior. O desempenho do TRISS em prever a sobrevivência destes pacientes foi medido utilizando curvas ROC, mostrando-se progressivamente menor, confirmando que o TRISS apresenta problemas em realidades de atendimento piores. Foi, então, desenvolvido o TRISS-AG, um algoritmo genético para ajustar a equação do TRISS. O desempenho do TRISS-AG e do TRISS aplicados aos bancos de dados foi comparado com base na quantidade de acertos na previsão de sobrevivência e o TRISS-AG apresentou desempenho melhor para todos os bancos de dados, com maior vantagem para os bancos de dados representado os piores cenários de atendimento. Os resultados confirmaram que é possível usar um algoritmo genético para ajustar o TRISS e que o algoritmo pode ter desempenho superior ao do método TRISS original.

Palavras-chave: Algoritmos genéticos, Computação evolutiva, Índices de gravidade do trauma, *Trauma and Injury Severity Score*, TRISS.

ABSTRACT

ENNES, A. T. **TRISS-AG: a genetic algorithm to adjust the TRISS prediction method of trauma patients survival**. 2021. 107 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

The TRISS method is widely used to predict the survival of trauma patients, having a key role both in guiding clinical decisions at the time of care and later to assess the performance of the health service. Although it is the most used worldwide, it has been criticized for not reflecting the reality of medical care in developing countries, as it was built by means of a logistic regression using data from American patients more than three decades ago. Studies have been done seeking to adapt it using local data and alternative techniques, including computational intelligence. This work proposes an approach to adjust TRISS using genetic algorithms, testing whether it is possible to use such a technique and outperform of the original TRISS when applied to data that represent such a reality. Initially, five databases were synthesized simulating patients treated by hospitals of progressively lower quality. The performance of TRISS in predicting the survival of these patients was measured using ROC curves, showing to be progressively lower, confirming that TRISS presents problems in worse care realities. Then, TRISS-AG was developed, a genetic algorithm to adjust the TRISS equation. The performance of TRISS-AG and TRISS applied to the databases was compared based on the number of correct survival predictions and TRISS-AG performed better for all databases, with a greater advantage for the databases representing the worst service scenarios. The results confirmed that it is possible to use a genetic algorithm to adjust TRISS and that such algorithm can perform better than the original TRISS method.

Keywords: Genetic algorithms, Evolutionary computation, Trauma severity scores, Trauma and Injury Severity Score, TRISS.

LISTA DE ILUSTRAÇÕES

Figura 1 – Composição do TRISS	36
Figura 2 – <i>Blend Crossover (BLX)</i>	41
Figura 3 – Curvas ROC aplicando TRISS nos bancos de dados	53
Figura 4 – Evolução do <i>fitness</i> máximo do TRISS-AG por variação de operadores para o BD-A	55
Figura 5 – Evolução do <i>fitness</i> máximo do TRISS-AG por variação de operadores para o BD-B10	56
Figura 6 – Evolução do <i>fitness</i> máximo do TRISS-AG por variação de operadores para o BD-B20	56
Figura 7 – Evolução do <i>fitness</i> máximo do TRISS-AG por variação de operadores para o BD-B30	57
Figura 8 – Evolução do <i>fitness</i> máximo do TRISS-AG por variação de operadores para o BD-B50	57
Figura 9 – Evolução do <i>fitness</i> durante a execução do TRISS-AG para o BD-A	59
Figura 10 – Evolução do <i>fitness</i> durante a execução do TRISS-AG para o BD-B10	60
Figura 11 – Evolução do <i>fitness</i> durante a execução do TRISS-AG para o BD-B20	60
Figura 12 – Evolução do <i>fitness</i> durante a execução do TRISS-AG para o BD-B30	61
Figura 13 – Evolução do <i>fitness</i> durante a execução do TRISS-AG para o BD-B50	61
Figura 14 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-A	62
Figura 15 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B10	63
Figura 16 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B20	63
Figura 17 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B30	64
Figura 18 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B50	64

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Algoritmo Genetico TRISS.R	79
Código-fonte 2 – Executar.R	88
Código-fonte 3 – Experimento.R	90
Código-fonte 4 – Fitness.R	97
Código-fonte 5 – Gerar Dados.R	98
Código-fonte 6 – Operators.R	100
Código-fonte 7 – Plot.R	102
Código-fonte 8 – Scores.R	105

LISTA DE TABELAS

Tabela 1 – Campos do banco de dados de pacientes	27
Tabela 2 – Nomes dos bancos de dados de pacientes	28
Tabela 3 – <i>Abbreviated Injury Scale</i> (AIS)	32
Tabela 4 – <i>Glasgow Coma Scale</i> (GCS)	34
Tabela 5 – <i>Revised Trauma Score</i> (GCS)	35
Tabela 6 – Coeficientes da equação do TRISS	37
Tabela 7 – Limites dos campos do banco de dados de pacientes	44
Tabela 8 – Cálculo de sobrevivência para cada banco de dados de pacientes	44
Tabela 9 – Genes do cromossomo de cada indivíduo no AG	45
Tabela 10 – EQM aplicando TRISS nos dos bancos de dados	52
Tabela 11 – Valores de AUC aplicando TRISS nos bancos de dados	54
Tabela 12 – Coeficientes das novas equações do TRISS calculados pelo TRISS-AG	58
Tabela 13 – Quantidade de acertos do TRISS e do TRISS-AG para os bancos de dados	65

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AIS	<i>Abbreviated Injury Scale</i>
AUC	<i>Area Under Curve</i>
AX	<i>Average Crossover</i>
BLX	<i>Blend Crossover</i>
EQM	Erro Quadrático Médio
EUA	Estados Unidos da América
FR	Frequência Respiratória
GCS	<i>Glasgow Coma Scale</i>
IA	Inteligência Artificial
ISS	<i>Injury Severity Score</i>
MTOS	<i>Major Trauma Outcome Study</i>
PAS	Pressão Arterial Sistólica
RCGA	<i>Real-Coded Genetic Algorithms</i>
ROC	<i>Receiver Operating Characteristic</i>
RST	<i>Random Sampling Technique</i>
RTS	<i>Revised Trauma Score</i>
TRISS	<i>Trauma and Injury Severity Score</i>
TS	<i>Trauma Score</i>

SUMÁRIO

1	INTRODUÇÃO	25
2	REVISÃO BIBLIOGRÁFICA	31
2.1	Métodos de Classificação de Trauma	31
2.2	<i>Abbreviated Injury Scale</i> (AIS)	32
2.3	<i>Injury Severity Score</i> (ISS)	32
2.4	<i>Glasgow Coma Scale</i> (GCS)	33
2.5	<i>Revised Trauma Score</i> (RTS)	34
2.6	<i>Trauma and Injury Severity Score</i> (TRISS)	36
2.7	Algoritmos Genéticos	39
3	ALGORITMO E IMPLEMENTAÇÃO	43
3.1	Geração do Banco de Dados	43
3.2	Implementação do TRISS-AG	45
4	RESULTADOS	51
4.1	Aplicação do TRISS aos Banco de Dados	51
4.2	Aplicação do TRISS-AG e comparação com o TRISS	54
5	CONCLUSÃO	67
	REFERÊNCIAS	71
	GLOSSÁRIO	77
	APÊNDICE A CÓDIGO-FONTE	79

INTRODUÇÃO

Trauma é uma das principais causas de morte e invalidez (WHO, 2009). O custo para a sociedade com as consequências de ferimentos intencionais ou acidentais corresponde a 16% do custo global com doenças, sendo este custo especialmente notável em países com menor renda. Diminuir este peso sobre a sociedade está entre os maiores desafios para a saúde pública neste século (WHO, 2004). Respostas a este problema precisam envolver iniciativas organizadas tanto de prevenção como de melhoria da qualidade do tratamento dado às vítimas. Na esfera do tratamento se encontram os serviços de saúde responsáveis pelo atendimento inicial às vítimas de trauma e seu tratamento posterior. A qualidade deste serviço de saúde pode ser caracterizada como o grau em que o tratamento oferecido aumenta a probabilidade de sobrevivência da vítima, dentro dos limites do conhecimento médico atual (WHO, 2009). Melhorias na prevenção do trauma e no controle da qualidade dos sistemas de atendimento ao traumatizado demonstraram efeitos mensuráveis em termos de vidas salvas e melhoria dos resultados. (DOMINGUES *et al.*, 2015). Melhorar esta qualidade de atendimento requer a otimização de habilidades profissionais e do uso de materiais e equipamentos, usando como critério o monitoramento dos resultados de melhoria da condição de saúde dos pacientes. A avaliação do atendimento precisa ser uma parte integral de qualquer sistema de cuidado a paciente severamente feridos. Esta avaliação precisa permitir a comparação com padrões nacionais e internacionais (WHO, 2009).

Um papel chave na melhoria da qualidade dos serviços de saúde tem sido desempenhado nas últimas décadas pelos sistemas de classificação de trauma. Estes sistemas são modelos matemáticos utilizados para avaliar a gravidade de um dado paciente e para prever a probabilidade de sobrevivência de uma dada população em termos estatísticos. O cálculo destes índices, ou *scores*, considera parâmetros de sinais vitais e avaliações feitas por médicos sobre os ferimentos nos primeiros momentos do atendimento. Os índices são usados para guiar decisões clínicas sobre os procedimentos imediatos a serem tomados, mas também tem utilidade posteriormente, na avaliação do desempenho do serviço de saúde (FANI-SALEK; TOTTEN; TEREZAKIS, 1999). A documentação sistemática dos atendimentos permite a geração de índices com base em grandes

volumes de dados coletados. Os índices gerados são medidas de desempenho que permitem a comparação dos resultados produzidos pelo cuidado com os pacientes entre diferentes serviços de saúde em uma região, país ou mesmo entre países. Esta comparação, ou *benchmarking*, é necessária para orientar iniciativas de melhoria de qualidade, com foco principal na redução de mortes que poderiam ser evitadas (WHO, 2009).

Nas últimas décadas, alguns modelos de predição de sobrevivência de trauma foram desenvolvidos e tem sido amplamente utilizados. Um dos mais comumente usados é o *Trauma and Injury Severity Score* (TRISS). Esta metodologia oferece uma abordagem padrão para avaliação de resultados de atendimento a pacientes de trauma (FANI-SALEK; TOTTEN; TEREZAKIS, 1999). O método TRISS foi desenvolvido a partir de dados do *Major Trauma Outcome Study* (MTOS), um estudo conduzido nos Estados Unidos da América (EUA) nos anos 80 envolvendo pacientes dos EUA e do Canadá. O método combina características anatômicas, fisiológicas e a idade do paciente no momento do atendimento para quantificar a sua probabilidade de sobrevivência. Na construção do método foi utilizado um modelo de regressão logística que resultou em coeficientes de uma equação linear aplicados às características do paciente (CHAMPION *et al.*, 1990).

Embora amplamente aceito, o método TRISS tem sido criticado em vários estudos por não refletir corretamente a realidade de países em desenvolvimento (DOMINGUES *et al.*, 2017). Como ele foi derivado de dados de pacientes dos EUA e Canadá, os coeficientes do método foram ajustados para a realidade destes países desenvolvidos com sistemas de saúde maduros. Seus resultados podem ser vistos como metas a serem alcançadas, mas que não retratam corretamente as dificuldades enfrentadas por países mais pobres devido a suas condições econômicas, sociais, demográficas, dos serviços de saúde locais e consequentes taxas de morbidade e mortalidade próprias. Estudos tem sido feitos recentemente buscando ajustar os coeficientes do método TRISS a bancos de dados de países em desenvolvimento, como o Brasil, alcançando resultados positivos (DOMINGUES *et al.*, 2018) (DOMINGUES *et al.*, 2017). Alternativas ao modelo de regressão logística utilizado pelo método TRISS tem sido também estudadas, como o uso de redes neurais e outras técnicas de inteligência computacional para ajustar os coeficientes do método, com resultados também positivos (CHRISTIE *et al.*, 2018) (THEODORAKI; KOUKOUVINOS; PARPOULA, 2010).

Adaptar com sucesso métodos de monitoramento de qualidade à realidade enfrentada pelos serviços de saúde brasileiros permite disponibilizar a estes serviços e às autoridades de saúde pública do país ferramentas mais precisas para avaliação de desempenho e estimativa de sobrevivência, aumentando a eficácia da aplicação de tais métodos e oferecendo uma contribuição relevante para programas de melhoria de qualidade de atendimento no país que podem, em última instância, resultar em uma melhor qualidade no tratamento de pacientes, no uso mais eficiente de recursos e na redução do número de mortes evitáveis durante o atendimento médico (DOMINGUES *et al.*, 2017).

Neste contexto, o presente trabalho tem como objetivo propor uma abordagem para ajustar os coeficientes do método TRISS usando a técnica de inteligência computacional dos algoritmos genéticos, procurando mostrar que por esta abordagem é possível obter melhor desempenho na previsão de sobrevivência em comparação ao método TRISS original, especialmente quando o banco de dados utilizado representa uma realidade diferente da norte-americana na qual o método TRISS foi desenvolvido, como seria o caso de um serviço de saúde pública brasileiro.

Devido à dificuldade em se obter dados reais de atendimento de pacientes por questões de privacidade e disponibilidade dos mesmos, optou-se nesta pesquisa por sintetizar dados de teste usando o próprio método TRISS como preditor de sobrevivência. Esta abordagem, embora viabilize a execução do estudo, implica na necessidade de se validar os resultados utilizando dados reais em estudos futuros.

No desenvolvimento do trabalho será, então, gerado um banco de dados que representa um conjunto de 20 mil pacientes aleatórios que poderiam chegar a um determinado hospital de uma determinada região para atendimento de trauma, coletados durante um período de vários anos. Cada paciente hipotético deste banco de dados terá um identificador numérico sequencial (1, 2, ..., 20.000), campos contendo características do seu estado de saúde no momento do atendimento (necessários para estimativa de sua sobrevivência pelo método TRISS) e um campo indicando sua sobrevivência como resultado do tratamento, como detalhado na [Tabela 1](#).

Tabela 1 – Campos do banco de dados de pacientes

Campo	Descrição
(Número da linha)	Identificador numérico sequencial do paciente
FR	Frequência respiratória
PAS	Pressão arterial sistólica
GLASGOW_RESULTADO	Índice na escala da Glasgow: avaliação do nível de consciência do paciente, como descrito na Seção 2.4 (Glasgow Coma Scale (GCS))
ISS_RESULTADO	Índice ISS: avaliação da severidade do trauma, levando em conta sua gravidade e parte do corpo atingida, como descrito na Seção 2.3 (Injury Severity Score (ISS))
IDADE	Idade do paciente
SOBREVIVEU	Indicador lógico apontando se o paciente sobreviveu (1 ou 0, correspondentes a “sim” ou “não”)

Cada campo referente ao estado de saúde terá seu valor gerado aleatoriamente dentro dos limites mínimo e máximo que seriam possíveis em uma medição real. O método TRISS será usado para preencher o campo indicador de sobrevivência levando em conta a probabilidade de sobrevivência por ele estimada. Sua equação será alimentada com os dados de cada paciente, resultando na sua probabilidade de sobrevivência P_s ($0 \leq P_s \leq 1$). Será, então, sorteado um valor

aleatório x ($0 \leq x \leq 1$) e comparado com a probabilidade P_s : caso x esteja dentro da probabilidade P_s ($x \leq P_s$), o resultado indicará sobrevivência. Para reduzir o ruído nos dados, este sorteio será repetido por 10 vezes para cada paciente, contando-se a quantidade de vezes em que o resultado tenha sido sua sobrevivência. Caso esta quantidade seja maior que cinco, considera-se que o paciente tenha sobrevivido, caso contrário não. Como o método TRISS foi desenvolvido a partir de dados dos EUA e do Canadá, o banco de dados será, dessa forma, representativo da realidade desta região e da qualidade de atendimento médico disponível nos hospitais desta região.

Serão geradas também outras quatro variações do mesmo banco de dados, com os mesmos pacientes, mas com alterações na forma de calcular a sobrevivência para representar realidades diversas em que a qualidade do atendimento médico disponível seja inferior, como seria o caso de hospitais brasileiros de diferentes níveis de performance. A forma de calcular a sobrevivência para cada variação do banco de dados será modificada reduzindo-se a probabilidade de sobrevivência estimada pelo método TRISS em 10%, 20%, 30% e 50% respectivamente. Assim, quando for sorteado um valor aleatório x ($0 \leq x \leq 1$), ele será comparado com a probabilidade P_s reduzida, considerando-se que o paciente tenha sobrevivido se $x \leq [P_s \cdot (1 - 10\%)]$, $x \leq [P_s \cdot (1 - 20\%)]$, $x \leq [P_s \cdot (1 - 30\%)]$ ou $x \leq [P_s \cdot (1 - 50\%)]$ respectivamente para cada variação do banco de dados.

Haverá, então, cinco bancos de dados gerados, representando cinco realidades diversas de qualidade de atendimento, que serão nomeados na forma descrita na [Tabela 2](#).

Tabela 2 – Nomes dos bancos de dados de pacientes

Nome	Descrição
BD-A	Banco de dados de um hospital norte-americano, com probabilidade de sobrevivência estimada pelo método TRISS original
BD-B10	Banco de dados de um hospital brasileiro com boa qualidade de atendimento, com a probabilidade de sobrevivência estimada pelo método TRISS reduzida em 10%
BD-B20	Banco de dados de um hospital brasileiro de menor qualidade, com probabilidade de sobrevivência estimada pelo método TRISS reduzida em 20%
BD-B30	Banco de dados de um hospital brasileiro de qualidade ainda menor, com probabilidade de sobrevivência estimada pelo método TRISS reduzida em 30%
BD-B50	Banco de dados de um hospital brasileiro de qualidade inferior, com probabilidade de sobrevivência estimada pelo método TRISS reduzida em 50%

A seguir, será feita uma avaliação do erro entre a estimativa de sobrevivência pelo método TRISS e o índice de sobrevivência apresentado pelos cinco bancos de dados. Os valores de erro para os bancos de dados serão então comparados.

A primeira hipótese que se pretende testar na presente pesquisa é a de que o método TRISS apresentará maior taxa de erro para os bancos de dados que representam serviços de saúde de menor qualidade, mostrando que o método criado com dados norte-americanos apresenta problemas ao ser usado para realidades diversas, como a brasileira. Será, então, desenvolvido um algoritmo genético que ajuste os coeficientes da equação do TRISS quando treinado com os dados de pacientes de um determinado banco de dados. Este algoritmo será executado com os cinco bancos de dados, gerando novas equações como a do TRISS, porém com coeficientes ajustados a estes bancos de dados. Será feita novamente a avaliação do erro entre a estimativa de sobrevivência feita por estas novas equações e o índice de sobrevivência apresentado pelos cinco bancos de dados, respectivamente. Os valores de erro serão comparados com os valores de erro apresentados pela equação original do TRISS para os mesmos bancos de dados.

A segunda hipótese que se pretende testar nesta pesquisa é a de que o algoritmo genético apresentará menor erro na estimativa de sobrevivência dos pacientes quando comparado à equação original do TRISS, mostrando que é possível usar um algoritmo genético para ajustar as equações do TRISS e que elas poderão prever melhor a sobrevivência dos pacientes ao levarem em conta a realidade dos serviços de saúde que originaram os dados, especialmente quando são diferentes da realidade norte-americana em que se baseou o método original, como é o caso dos serviços de saúde brasileiros.

O trabalho será estruturado de forma a apresentar no [Capítulo 2 \(Revisão Bibliográfica\)](#) a revisão bibliográfica sobre o método TRISS e outros índices relacionados, bem como as técnicas matemáticas e computacionais usadas para o ajuste; explicar no [Capítulo 3 \(Algoritmo e Implementação\)](#) o desenvolvimento do *software* e os experimentos realizados no trabalho proposto; descrever os resultados alcançados no [Capítulo 4 \(Resultados\)](#) e discuti-los na conclusão, no [Capítulo 5 \(Conclusão\)](#). O código fonte desenvolvido será reproduzido no [Apêndice A](#).

Todos os códigos-fonte foram armazenados na ferramenta de controle de versões GitHub e o repositório está disponível em <https://github.com/andre-ennes/Algoritmo-Genetico-TRISS>.

REVISÃO BIBLIOGRÁFICA

É apresentada nesta seção a revisão bibliográfica com os conceitos, métodos de classificação de trauma e técnicas computacionais necessários ao desenvolvimento do presente trabalho.

2.1 Métodos de Classificação de Trauma

Os sistemas, ou métodos, de classificação de trauma são modelos matemáticos úteis para prever a probabilidade estatística de sobrevivência de um indivíduo com base em dados de uma população previamente estudada, embora a maioria não seja eficaz em prever a qualidade de vida de indivíduos sobreviventes. Médicos e profissionais de saúde usam os índices, ou *scores*, para orientar decisões de tratamento. Gestores de serviços de saúde usam os índices agregados de vários pacientes atendidos em um período como medidas de desempenho e comparação, ou *benchmark*, com outros serviços e padrões nacionais e internacionais para orientar iniciativas de melhoria de qualidade destes serviços (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

Métodos de classificação de trauma podem ser classificados em fisiológicos, anatômicos ou combinados. Os métodos de classificação fisiológicos medem as mudanças físicas induzidas pelo trauma. Eles tendem a se concentrar em anormalidades neurológicas, hematológicas e respiratórias e são fortes preditores de mortalidade. A maioria dos métodos de classificação fisiológica não considera o local da lesão no corpo do paciente. Sistemas de classificação anatômicos caracterizam o grau do dano anatômico ao paciente, ponderado pela importância do local da lesão em seu corpo. Eles claramente delineiam danos anatômicos, mas não consideram distúrbios não visíveis a sistema de órgãos internos. Os sistemas combinados tentam superar as limitações dos sistemas anatômicos ou fisiológicos puros, incorporando elementos de ambos. Sistemas combinados são superiores na previsão de sobrevivência. No entanto, como são mais abrangentes e, os sistemas combinados são também mais complicados de se calcular, sendo por isso mais usados em serviços de saúde após os passos iniciais de atendimento, estabilização e

tratamento de pacientes (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

Nas próximas sessões são detalhados os métodos de classificação de trauma utilizados no desenvolvimento desta pesquisa. Eles estão entre os mais amplamente usados mundialmente (DOMINGUES *et al.*, 2017).

2.2 *Abbreviated Injury Scale (AIS)*

O *Abbreviated Injury Scale* (AIS) é um sistema de classificação anatômico que foi desenvolvido em 1971 para avaliar originalmente traumatismo contuso (não penetrante), embora tenha sido estendido posteriormente para avaliar também traumatismo penetrante. O AIS consiste em atribuir a cada dano individual sofrido pelo paciente um valor inteiro, ou *score*, de 1 até 5, conforme descrito na Tabela 3. Uma pontuação de 6 é reservada para lesões fatais. Uma referência para atribuição dos pesos encontra-se no Manual do Dicionário AIS, que é um compêndio de mais de 1200 lesões e seus índices AIS correspondentes divididos em seções anatômicas: cabeça; face; pescoço; tórax; abdome e conteúdo pélvico; coluna; membros superiores; membros inferiores; superfície externa; outros. O AIS é complicado de se usar durante o tratamento inicial de uma vítima de trauma, pois cada lesão individual deve ser pesquisada no Manual do Dicionário de AIS (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

Tabela 3 – *Abbreviated Injury Scale* (AIS)

Tipo de Trauma	Score AIS
Leve	1
Moderado	2
Severo, sem risco de vida	3
Severo, com risco de vida, sobrevivência provável	4
Crítico, sobrevivência incerta	5
Sobrevivência não possível	6

Fonte: Fani-Salek, Totten e Terezakis (1999).

Desde a sua introdução, o AIS foi revisado seis vezes e a versão atual inclui uma gama mais ampla de lesões e se tornou mais específica do que as versões anteriores. Foram adicionados valores para lesões penetrantes. O AIS correlaciona-se bem com o grau de lesão, no entanto, ela sofre como uma ferramenta prognóstica porque não leva em conta desequilíbrios fisiológicos ou estado de saúde crônico (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

2.3 *Injury Severity Score (ISS)*

O *Injury Severity Score* (ISS) é o sistema de classificação anatômica mais utilizado no mundo. Foi introduzido pela primeira vez em 1974 como um método de comparar resultados de pacientes com múltiplas lesões. Ele é baseado no AIS e, portanto, também não é uma pontuação

rápida para se calcular. Para se chegar ao índice ISS, cada lesão deve ser codificada pelo sistema AIS. No entanto, os cálculos subsequentes são simples. O índice ISS apresenta uma boa correção com o índice de mortalidade, sendo um bom preditor do resultado (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

O índice do ISS é um *score* obtido por um cálculo matemático baseado nos *scores* AIS mais elevados de cada uma das três diferentes regiões do corpo mais gravemente traumatizadas. Para o ISS, as regiões anatômicas estão agrupadas em: cabeça e pescoço; face; tórax, abdome e conteúdos pélvicos; extremidades e cintura pélvica; superfície externa. O ISS é calculado após a classificação dos índices mais graves de cada uma das seis regiões, escolhendo-se os três valores das AIS mais altos em segmentos corpóreos diferentes e realizando-se a soma dos quadrados desses índices, como descrito na Equação 2.1. O índice ISS varia de 1 a 75 e a pontuação máxima possível é de $5^2 + 5^2 + 5^2 = 75$, pois todas as lesões fatais com score AIS de 6 resultam automaticamente na pontuação de 75. Pacientes com índices ISS de 16 ou mais devem ser tratados em um grande centro de atendimento de trauma (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

$$ISS = \begin{cases} 75 & \text{se } AIS_1 = 6 \text{ ou } AIS_2 = 6 \text{ ou } AIS_3 = 6 \\ AIS_1^2 + AIS_2^2 + AIS_3^2 & \text{caso contrário} \end{cases} \quad (2.1)$$

O ISS é abrangente e tem bom poder de previsão. Sua desvantagem está em que, na fase inicial do atendimento da vítima de trauma, a natureza de algumas lesões ainda não foi totalmente determinada. Além disso, o ISS não leva em conta a idade do paciente ou condições médicas crônicas. Múltiplas lesões na mesma área do corpo não são mais pesadas que uma única lesão nessa área; por exemplo, um paciente com uma combinação de lesões maciças do fígado, baço e intestino é pontuado da mesma forma como se aquele paciente tivesse qualquer um desses ferimentos sozinho (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

2.4 Glasgow Coma Scale (GCS)

A *Glasgow Coma Scale* (GCS), ou Escala de Coma de Glasgow, foi desenvolvida no início da década de 1970 para padronizar a avaliação da alteração do nível de consciência de pacientes acometidos por trauma cranioencefálico. A avaliação do nível de consciência é baseada em três respostas comportamentais: abertura ocular, resposta verbal e resposta motora. Isto torna a GCS fácil de usar mesmo antes do atendimento hospitalar ao paciente. Ela é útil para indicar a profundidade do estado de coma e predizer sua duração (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

A pontuação GCS varia de 3 (coma profundo ou morte) até 15 (nenhum déficit neurológico). Ela é calculada somando-se a pontuação individual das três respostas comportamentais, conforme descrito na Tabela 4.

Tabela 4 – *Glasgow Coma Scale* (GCS)

Resposta	Score
1. Abertura Ocular	
Espontânea	4
Ao comando verbal	3
À dor	2
Nenhuma	1
2. Resposta Verbal	
Orientado	5
Confuso	4
Palavras inapropriadas	3
Sons incompreensíveis	2
Nenhuma	1
3. Resposta Motora	
Obedece a comandos	6
Localiza a dor	5
Retirada (à dor)	4
Flexão (à dor)	3
Extensão (à dor)	2
Nenhuma	1

Fonte: Fani-Salek, Totten e Terezakis (1999).

O GCS é simples de aplicar e é usado pela maioria dos sistemas de serviços médicos de emergência para determinar a urgência do atendimento. Além disso, vários outros sistemas de pontuação de trauma incorporaram o GCS ou uma transformação matemática do mesmo em sua composição (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

Uma limitação significativa do GCS está no fato de ser uma escala ordinal discreta, não contínua. Neste tipo de escala, a diferença entre os valores unitários não é consistente e apenas permite comparações entre dois índices como um sendo melhor ou pior que outro. É difícil comparar, por exemplo, um grupo com uma pontuação de 10 com um grupo com uma pontuação de 12 (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

2.5 *Revised Trauma Score* (RTS)

O *Revised Trauma Score* (RTS) é um índice classificado como fisiológico, pois leva em consideração os parâmetros das funções vitais do paciente. Foi originalmente criado em 1981 com o nome *Trauma Score* (TS) a partir do GCS, com a adição de quatro elementos: Frequência Respiratória (FR), Pressão Arterial Sistólica (PAS), grau de expansão respiratória e enchimento capilar. Os dois últimos se mostraram difíceis de se avaliar em campo e foram removidos quando o índice foi revisado e se tornou o RTS (CHAMPION *et al.*, 1989) (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

O RTS, então, incorpora três componentes que devem ser medidos no momento da admissão hospitalar da vítima: GCS, FR e PAS. Para o cálculo do índice, inicialmente os três componentes são convertidos para um *score* entre 0 e 4 conforme a faixa de valores de cada um. As faixas estão descritas na [Tabela 5](#). Os *scores* obtidos são, então, multiplicados pelo peso atribuído a cada um na definição do RTS e depois somados, conforme a [Equação 2.2](#). Os pesos foram obtidos a partir de uma regressão linear com os dados do MTOS ([DOMINGUES et al., 2015](#)) ([BOYD; TOLSON; COPES, 1987](#)) ([CHAMPION et al., 1989](#)) ([CHAMPION et al., 1990](#)).

$$RTS = 0,9368 \cdot GCS_{score} + 0,2908 \cdot FR_{score} + 0,7326 \cdot PAS_{score} \quad (2.2)$$

O resultado do RTS é um índice contínuo entre 0 e aproximadamente 8 (mais precisamente 7,8408). Valores menores representam traumas piores e valores maiores são relacionados a prognósticos melhores e consequentemente maior probabilidade de sobrevivência ([CHAMPION et al., 1990](#)) ([FANI-SALEK; TOTTEN; TEREZAKIS, 1999](#)).

Tabela 5 – Revised Trauma Score (GCS)

Componente / Faixa	Peso / Score
Glasgow Coma Scale	0,9368
13-15	4
9-12	3
6-8	2
4-5	1
3	0
Frequência Respiratória	0,2908
10-29	4
> 29	3
6-9	2
1-5	1
0	0
Pressão Arterial Sistólica	0,7326
> 89	4
76-89	3
50-75	2
1-49	1
0	0

Fonte: [Fani-Salek, Totten e Terezakis \(1999\)](#).

O RTS é um forte preditor de sobrevivência. O RTS efetivamente incorpora a gravidade da lesão na cabeça como preditor de mortalidade e é fácil de usar em campo, mesmo antes do pleno conhecimento dos órgãos lesados dos pacientes, embora seja mais complexo do que calcular somente o GCS ([FANI-SALEK; TOTTEN; TEREZAKIS, 1999](#)).

2.6 Trauma and Injury Severity Score (TRISS)

O método TRISS foi desenvolvido pelo Colégio Americano de Cirurgiões a partir de um estudo iniciado em 1982 e publicado em 1990, denominado *Major Trauma Outcome Study* (MTOS). O estudo incluiu dados demográficos, etiológicos, de severidade de trauma e de resultado do tratamento de 80.544 pacientes de 139 serviços de saúde dos EUA e do Canadá coletados entre 1982 e 1987 (CHAMPION *et al.*, 1990).

O método tem como objetivo quantificar a probabilidade de sobrevivência de uma vítima ou população após um trauma grave, além de permitir a comparação da mortalidade de pacientes de diferentes centros hospitalares e sistemas de atendimento com os índices esperados estimados pelo TRISS (DOMINGUES *et al.*, 2015). O TRISS é atualmente um dos índices mais utilizados mundialmente para tais fins (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

Para o cálculo do TRISS são usados os elementos fisiológicos incorporados pelo RTS e os elementos anatômicos do ISS, além de considerar a idade da vítima e o tipo de trauma (contuso ou penetrante). Indiretamente, o TRISS também incorpora o GCS e o AIS, pois o RTS utiliza o GCS em sua composição, bem como o ISS utiliza o AIS, como descrito na Seção 2.5 e na Seção 2.3. O TRISS é, portanto, um método de classificação combinado (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

A composição do TRISS, incluindo os outros índices anatômicos e fisiológicos que ele incorpora, é ilustrada na Figura 1.

Figura 1 – Composição do TRISS

Na construção do método TRISS foi utilizado um modelo de regressão logística aplicada aos dados do MTOS que derivou os coeficientes de uma equação linear aplicados aos índices RTS, ISS e a um parâmetro relativo à idade do paciente. O resultado do índice TRISS é um número entre 0 e 1 que representa a probabilidade de sobrevivência do paciente. Desta forma, usando a metodologia TRISS, a probabilidade P_s de sobrevivência é estimada pela Equação 2.3

(CHAMPION *et al.*, 1990) (BOYD; TOLSON; COPES, 1987) (FANI-SALEK; TOTTEN; TEREZAKIS, 1999).

$$P_s = \frac{1}{1 + e^{-b}} \quad (2.3)$$

Onde e é a constante dos logaritmos naturais e b é dado pela Equação 2.4.

$$b = b_0 + b_1 \cdot RTS + b_2 \cdot ISS + b_3 \cdot AGE \quad (2.4)$$

Onde os coeficientes b_0 , b_1 , b_2 e b_3 são os obtidos pelo MTOS. Eles são diferentes para traumas contusos e perfurantes e estão listados na Tabela 6. É exatamente sobre esses coeficientes que o presente trabalho se concentra, efetuando o ajuste dos mesmos por meio de um algoritmo genético.

Tabela 6 – Coeficientes da equação do TRISS

Tipo de Trauma	b_0	b_1 (RTS)	b_2 (ISS)	b_3 (AGE)
Contuso	-1,2470	0,9544	-0,0768	-1,9052
Penetrante	-0,6029	1,1430	-0,1516	-2,6676

Fonte: Champion *et al.* (1990).

Na mesma Equação 2.4, AGE é o parâmetro que varia conforme a idade do paciente. Ele é definido pela Equação 2.5.

$$AGE = \begin{cases} 0 & \text{se idade do paciente} < 55 \\ 1 & \text{se idade do paciente} \geq 55 \end{cases} \quad (2.5)$$

Este parâmetro incorpora ao cálculo do TRISS um risco aumentado de morte relacionado ao comprometimento cardíaco associado à idade. Entre os pacientes do MTOS, uma idade igual ou superior a 55 anos foi identificada como tendo um impacto significativo na mortalidade em comparação a pacientes nas mesmas condições de severidade de trauma, mas com idade inferior a 55 anos (BOYD; TOLSON; COPES, 1987).

Para exemplificar o cálculo do TRISS, é apresentado na Equação 2.6 o cálculo da probabilidade de sobrevivência de um paciente hipotético de 58 anos. Este paciente teria chegado ao atendimento de emergência com $FR = 22$, $PAS = 85$ e teria avaliados seu $GCS = 12$, devido a uma baixa redução em seu nível de consciência, e seu $ISS = 16$, devido a um único trauma

contuso de natureza severa com risco de vida ($AIS = 4$).

$$FR = 22$$

$$PAS = 85$$

$$GCS = 12$$

$$ISS = 16$$

$$GCS_{score} = 3 \quad (\text{faixa 9-12})$$

$$FR_{score} = 4 \quad (\text{faixa 10-29})$$

$$PAS_{score} = 3 \quad (\text{faixa 76-89})$$

$$RTS = 0,9368 \cdot GCS_{score} + 0,2908 \cdot FR_{score} + 0,7326 \cdot PAS_{score}$$

$$RTS = 0,9368 \cdot 3 + 0,2908 \cdot 4 + 0,7326 \cdot 3$$

$$RTS = 2,8104 + 1,1632 + 2,1978$$

$$RTS = 6,1714$$

$$AGE = 1 \quad (\text{idade} \geq 55) \tag{2.6}$$

$$b = b_0 + b_1 \cdot RTS + b_2 \cdot ISS + b_3 \cdot AGE$$

$$b = -1,2470 + 0,9544 \cdot 6,1714 + (-0,0768) \cdot 16 + (-1,9052) \cdot 1$$

$$b = -1,2470 + 5,88998416 - 1,2288 - 1,9052$$

$$b = 1,50898416$$

$$P_s = \frac{1}{1 + e^{-b}}$$

$$P_s = \frac{1}{1 + e^{-1,50898416}}$$

$$P_s = \frac{1}{1 + 0,221134501171911}$$

$$P_s = 0,818910610616856$$

$$P_s \approx 81,9\%$$

Logo, no caso hipotético acima, a probabilidade de sobrevivência do paciente dada pelo método TRISS seria de aproximadamente 0,8189 ou 81,9%.

2.7 Algoritmos Genéticos

Um Algoritmo Genético (AG) é uma abordagem heurística para solução de problemas de otimização inspirada nos processos biológicos de seleção natural da teoria da evolução das espécies de Darwin. Os AGs são parte de uma família maior, a dos algoritmos evolutivos, uma das áreas de estudo da computação bioinspirada. Em um AG busca-se evoluir uma população de possíveis soluções para um problema de otimização na direção de soluções melhores. Cada solução candidata é chamada de indivíduo e possui um conjunto de propriedades representadas em estruturas de dados chamadas de cromossomos (ou genótipo), que podem ser alteradas durante o processo de evolução. Um indivíduo é, então, uma possível solução dentro do espaço de pesquisa de um determinado problema (WHITLEY, 1994).

Um AG consiste em um processo iterativo que se inicia com uma população de indivíduos gerada aleatoriamente (MITCHELL; FORREST, 1994). Cada indivíduo da população é avaliado em cada etapa da iteração, também chamada de geração, e recebe uma medida de *fitness* que representa quão bem o indivíduo está “adaptado” ao ambiente imposto pelo problema, ou, em outras palavras, quão bom é seu desempenho na solução do problema quando comparado aos resultados esperados (TOMASSINI, 1995). As gerações seguintes são formadas pela evolução das anteriores de modo que, com o tempo, a população passa a ser composta por indivíduos melhores (mais adaptados, ou com melhor *fitness*). Cada geração subsequente é criada pela reprodução de um grupo selecionado de indivíduos da geração anterior. Os indivíduos são selecionados por critérios baseados em seu *fitness* de forma que indivíduos melhores tenham uma maior probabilidade de serem selecionados para se reproduzir. Durante a reprodução aplicam-se, tradicionalmente, dois operadores genéticos conhecidos na biologia como *crossover* (ou cruzamento) e mutação (WATSON; POLLACK, 2000). O *crossover* combina partes dos cromossomos de dois indivíduos pais para formar o cromossomo de um novo indivíduo filho com objetivo de passar para a próxima geração características potencialmente positivas combinadas de ambos. A mutação produz pequenas variações (ou erros intencionais) no cromossomo do indivíduo filho de forma a aumentar a diversidade genética da população de indivíduos como um todo. Os pais se reproduzem, então, criando um indivíduo que consiste normalmente em uma combinação do material genético dos pais (cruzamento) junto com uma pequena quantidade de erros de cópia (mutação). A reprodução pode ser sexuada, quando ocorre o cruzamento entre dois pais, ou assexuada, caso em que apenas uma mutação é usada para produzir filhos de um único pai (BROOKS, 2000).

A seleção é a etapa de um AG em que os indivíduos são escolhidos para reprodução. Ela pode ser feita por diferentes métodos baseados no seu *fitness*, sendo os chamados *roulette*, *ranking* e *tournament* alguns dos mais conhecidos (SIMÕES, 2000). No método *roulette*, a probabilidade de se escolher um indivíduo é proporcional ao seu *fitness*. Este método de seleção pode ser comparado ao giro de uma roleta (por isso seu nome) dividida em apostas correspondentes ao tamanho da população, sendo que o tamanho de cada aposta é proporcional ao *fitness* do indivíduo

correspondente (TOMASSINI, 1995). Se um indivíduo tem um *fitness* alto, ele tem uma alta probabilidade de ser selecionado e produzir descendentes. Um problema com essa abordagem é que, depois que a população converge e os indivíduos melhoram, as diferenças de *fitness* entre os indivíduos se tornam pequenas, o que torna a seleção ineficaz (TOMASSINI, 1996). Outros métodos que não alocam a probabilidade proporcionalmente ao *fitness* tentam resolver esse problema (SIMÕES, 2000). No método *ranking* de seleção, os indivíduos são ordenados por *fitness* e o melhor indivíduo é selecionado para reprodução um múltiplo predeterminado do número de vezes em que o pior é selecionado (TODD; MILLER, 1997). No método *tournament*, ou por torneio, vários indivíduos (o tamanho do torneio) são selecionados aleatoriamente com probabilidade uniforme e apenas o vencedor do torneio (aquele com melhor *fitness*) é selecionado para se reproduzir. A pressão de seleção, uma medida da probabilidade de participação de um indivíduo no torneio, pode ser ajustada alterando o tamanho do torneio. Em um torneio maior, indivíduos fracos têm uma chance menor de serem selecionados, pois há uma probabilidade maior de que um indivíduo mais forte também esteja nesse torneio. Este método não é global, e torneios locais podem ser realizados simultaneamente em populações espacialmente isoladas (HARVEY, 1993).

Após a seleção de indivíduos para reprodução, o próximo passo de um AG é a criação da próxima geração da população a partir dos indivíduos selecionados. Em um AG clássico, o *crossover*, ou cruzamento, é o operador fundamental de reprodução enquanto o operador de mutação é considerado suplementar. O *crossover* combina partes dos cromossomos dos pais que são comprovadamente boas, pois os pais foram vencedores de uma seleção e, portanto, tem mais chances de gerar indivíduos mais aptos. A versão original do operador de *crossover* é o cruzamento de um ponto, na qual, uma vez que duas cadeias de genes são selecionadas para produzir descendentes, uma posição p é selecionada aleatoriamente entre um e o tamanho das cadeias (*length*) menos um ($1 \leq p \leq [length - 1]$). Ambas as strings (A e B) são divididas nessa posição, gerando quatro trechos: *trecho1A*, *trecho1B*, *trecho2A* e *trecho2B*. Duas novas cadeias de genes são produzidas trocando-se os trechos originais, combinando assim: $filho1 = trecho1A + trecho2B$ e $filho2 = trecho2A + trecho1B$ (onde $+$ significa concatenação de sequências de genes). Esses dois novos filhos entrarão na nova população da próxima geração no lugar de seus pais. Essa estratégia foi inspirada na biologia e é muito simples e eficiente. A ideia foi expandida para o cruzamento multipontos. Nesta variação aplica-se a mesma estratégia, mas divide-se as sequências originais em mais de um ponto de corte e troca-se os trechos entre esses pontos (SIMÕES, 2000). Outra estratégia é o *uniform crossover*, ou cruzamento uniforme, que também gera dois filhos a partir da sequência de genes de dois pais. Para cada gene do primeiro filho, o gene de um dos pais é copiado aleatoriamente e o segundo filho fica com o gene do outro pai, repetindo-se a escolha para todos os genes até que se completem os cromossomos dos filhos, que entram para a população da nova geração no lugar dos pais. Essa estratégia tem menos probabilidade de preservar bons conjuntos de genes, mas fornece uma distribuição mais uniforme dos genes individuais (WATSON; POLLACK, 2000). *Average Crossover* (AX), ou

cruzamento pela média, é uma estratégia de cruzamento que tira a média de dois pais para criar um único filho. Ela seleciona dois pais e, para cada gene do filho, calcula a média dos genes correspondentes dos pais, até completar o cromossomo do filho (UMBARKAR; SHETH, 2015).

Muitos outros operadores de *crossover* têm sido propostos, alguns específicos para os chamados *Real-Coded Genetic Algorithms* (RCGA), ou algoritmos genéticos de codificação real, algoritmos em que os genes de cada indivíduo são codificados na forma de números reais. O *Blend Crossover* (BLX) é um operador deste tipo que demonstra uma boa capacidade de busca por solução para funções de *fitness* separáveis (TAKAHASHI; KITA, 2001). Os algoritmos RCGA têm a característica de usar números reais para codificar genes dos cromossomos dos indivíduos, o que torna possível utilizar variáveis de grande domínio, o que seria difícil usando codificação binária. Outra vantagem dos RCGA é a capacidade de explorar a gradualidade das variáveis representando os genes, permitindo ajustes melhores e mais rápidos. Adicionalmente, o uso da codificação real é muito próximo da formulação natural de muitos problemas, tornando o código do AG mais simples e sua execução mais rápida (HERRERA; LOZANO; VERDEGAY, 1998). No cruzamento BLX, um filho x^c é gerado a partir de dois pais x^1 e x^2 . O valor de cada gene x_i^c do filho é escolhido aleatoriamente, usando distribuição probabilística uniforme, no intervalo $[x_i^1 - \alpha(x_i^2 - x_i^1), x_i^2 + \alpha(x_i^2 - x_i^1)]$, onde $x_i^1 \leq x_i^2$ e $\alpha > 0$, como ilustrado na Figura 2. O parâmetro α ajusta a variabilidade do operador, sendo indicado como uma boa escolha o valor de $\alpha = 0,5$ (ESHELMAN; SCHAFFER, 1993).

Figura 2 – *Blend Crossover* (BLX)

Fonte: Sudhoff (2007).

A mutação é o operador genético usado em um AG para manter a diversidade genética dos cromossomos de uma população entre uma geração e a seguinte. É análoga à mutação biológica. A mutação altera um ou mais genes aleatoriamente de acordo com uma taxa de mutação definida (SIMÕES, 2000). Tanto na reprodução sexuada, com uso de *crossover*, quanto na assexuada, a mutação é necessária para evitar a estagnação da busca, introduzindo diversidade na nova geração. Sem mutação, a evolução só pode combinar os indivíduos da população para formar as próximas gerações. Como apenas os melhores indivíduos são selecionados para se reproduzir, a população tende a convergir para uma única configuração melhor. Portanto, após algumas gerações, os genes da população permanecerão iguais e a população deixará de melhorar, pois não poderá mais ser modificada. A mutação introduz uma nova diversidade ao alterar aleatoriamente alguns genes do cromossomo toda vez que os indivíduos se reproduzem. Portanto, modificando sua

cadeia genes e permitindo que a evolução continue. No entanto, as taxas de mutação precisam ser baixas, caso contrário, a pesquisa tende a degenerar em uma caminhada aleatória (HARVEY; THOMPSON, 1997).

Estratégias de mutação adaptativa variam a taxa ou a forma de mutação durante a execução do AG, permitindo que o espaço de busca seja explorado de maneira uniforme no início e mais localmente no final (THOMPSON, 1997). Uma pequena taxa de mutação tenderá a convergir a população para um ótimo local (HARVEY, 1993). Se a taxa de mutação aumentar, a população poderá se espalhar ao redor desses ótimos locais buscando na vizinhança da solução por caminhos em direção a picos com soluções melhores, mas se a taxa de mutação se tornar muito grande, o *fitness* da população poderá se dispersar completamente. Contudo, mesmo usando mutação variável, os AGs geralmente não são fortes o suficiente para demonstrar conclusivamente que um ótimo global foi alcançado (CLIFF; HARVEY; HUSBANDS, 1997) (HARVEY, 1997).

ALGORITMO E IMPLEMENTAÇÃO

Nesta seção é descrito o algoritmo genético desenvolvido durante o presente trabalho, detalhada sua implementação e a geração dos bancos de dados de pacientes.

Todos os códigos-fonte foram desenvolvidos na linguagem de programação R, uma linguagem de software grátis criada para computação estatística muito usada em aplicações de ciência de dados ([R CORE TEAM, 2019](#)). Os códigos foram desenvolvidos, testados e executados na versão R `version 3.6.1`. Foi usado também o ambiente gráfico de desenvolvimento RStudio Version 1.2.5019 ([RSTUDIO TEAM, 2019](#)).

Adicionalmente aos recursos básicos da linguagem R, foram também usados nesta pesquisa os pacotes de recursos ISLR `version 1.2` ([JAMES *et al.*, 2017](#)) e pROC `version 1.16.2` ([ROBIN *et al.*, 2011](#)).

3.1 Geração do Banco de Dados

Foi gerado um banco de dados representando um conjunto de 20 mil pacientes hipotéticos que teriam chegado a um determinado hospital de uma determinada região para atendimento de trauma. Os dados para cada paciente neste banco incluem um identificador numérico sequencial, características do seu estado de saúde no momento do atendimento (FR, PAS, índice GCS e índice ISS), sua idade e um campo indicando sua sobrevivência ao final do tratamento. Por questão de simplicidade, decidiu-se neste trabalho por restringir o experimento apenas a traumas contusos, assim convencionou-se que todos os traumas representados no banco de dados são deste tipo. Os campos do banco de dados estão detalhados na [Tabela 1 \(Campos do banco de dados de pacientes, Página 27\)](#). Cada campo representando características do estado de saúde do paciente, além da idade, teve seu valor gerado dentro dos limites mínimo e máximo (inclusive), definidos pelo autor, que correspondem aos que seriam possíveis e razoáveis em uma medição real. Cada valor foi gerado aleatoriamente por distribuição probabilística uniforme usando a

função `sample` do R. Os limites mínimo e máximo de cada campo estão na [Tabela 7](#).

Tabela 7 – Limites dos campos do banco de dados de pacientes

Campo	Mínimo	Máximo
FR	0	39
PAS	0	180
GLASGOW_RESULTADO	3	15
ISS_RESULTADO	1	75
IDADE	0	90

O método TRISS foi usado para preencher o campo indicador de sobrevivência com base na probabilidade de sobrevivência estimada pelo método. Sua equação foi alimentada com os dados de cada paciente, resultando na sua probabilidade de sobrevivência P_s ($0 \leq P_s \leq 1$). Para cada paciente, foi sorteado um valor aleatório x ($0 \leq x \leq 1$) por distribuição probabilística uniforme usando a função `runif` do R. O valor x foi comparado com a probabilidade P_s : caso x estivesse dentro da probabilidade P_s ($x \leq P_s$), considerou-se que o resultado indicou sobrevivência. Para reduzir o ruído nos dados, este sorteio foi repetido por 10 vezes para cada paciente, contando-se a quantidade de vezes em que o resultado tenha sido sua sobrevivência. Caso esta quantidade tenha sido maior que cinco, considerou-se que o paciente tenha sobrevivido, caso contrário não. Como o método TRISS foi desenvolvido a partir de dados dos EUA e do Canadá, este banco de dados representa a qualidade de atendimento médico disponível nos hospitais desta região.

Tabela 8 – Cálculo de sobrevivência para cada banco de dados de pacientes

Nome	Paciente Sobreviveu Caso
BD-A	$x \leq P_s$
BD-B10	$x \leq [P_s \cdot (1 - 10\%)]$
BD-B20	$x \leq [P_s \cdot (1 - 20\%)]$
BD-B30	$x \leq [P_s \cdot (1 - 30\%)]$
BD-B50	$x \leq [P_s \cdot (1 - 50\%)]$

Foram geradas também outras quatro variações do mesmo banco de dados, com os mesmos pacientes, mas com alterações na forma de calcular a sobrevivência para representar realidades diversas em que a qualidade do atendimento médico disponível seja inferior, como seria o caso de hospitais brasileiros de diferentes níveis de desempenho. A forma de calcular a sobrevivência para cada variação do banco de dados foi modificada reduzindo-se a probabilidade de sobrevivência estimada pelo método TRISS em 10%, 20%, 30% e 50%, respectivamente. Assim, o valor aleatório x foi comparado com a probabilidade P_s reduzida correspondente a cada variação do banco de dados, como detalhado na [Tabela 8](#).

No total, foram gerados cinco bancos de dados representando cinco realidades diversas de qualidade de atendimento, nomeados conforme descrito na [Tabela 2 \(Nomes dos bancos de dados de pacientes, Página 28\)](#). O código que gerou os bancos de dados foi nomeado como “Gerar Dados.R” e está listado no [Código-fonte 5](#), no [Apêndice A](#).

3.2 Implementação do TRISS-AG

Foi desenvolvido um código-fonte que implementa o AG que ajusta os coeficientes do TRISS, seguindo o modelo clássico ([MITCHELL; FORREST, 1994](#)). O método foi nomeado como “TRISS-AG”. Ele começa gerando a população inicial de indivíduos. Cada indivíduo possui um cromossomo composto por cinco genes que correspondem, na ordem, ao limite de idade do parâmetro AGE e aos quatro coeficientes usados na regressão do TRISS: b_0 e multiplicadores de RTS, ISS e AGE. A correspondência dos genes ao componente do cálculo do TRISS está resumida na [Tabela 9](#), incluindo os valores padrão que compõem o cálculo original do método. Os genes são inicializados aleatoriamente com distribuição probabilística uniforme na faixa de 0 a 110 para o gene 1 e de -4 a 4 para os demais genes. Estas faixas podem ser ajustadas por argumento da função principal do código (`algoritmo.genetico`), que implementa o AG em si. Ele foi nomeado como “Algoritmo Genetico TRISS.R” e está listado no [Código-fonte 1](#), no [Apêndice A](#).

Tabela 9 – Genes do cromossomo de cada indivíduo no AG

Gene	Componente do Cálculo do TRISS	Valor Padrão
1	Limite de idade do parâmetro AGE	55
2	Coefficiente b_0 no cálculo do TRISS	$-1,2470$
3	Coefficiente de RTS no cálculo do TRISS	$0,9544$
4	Coefficiente de ISS no cálculo do TRISS	$-0,0768$
5	Coefficiente de AGE no cálculo do TRISS	$-1,9052$

Com a população inicial pronta, o AG entra no laço de gerações que será repetido pelo número de vezes definido por argumento da função. Neste trabalho foi definido o limite de 100 gerações, estabelecido em testes como um limite equilibrado, suficiente para a convergência do AG, mas mantendo o desempenho e o tempo de execução dentro de limites práticos aceitáveis.

Na primeira execução do laço, com a geração inicial da população, apenas o cálculo do *fitness* é executado, como descrito adiante, de forma a preparar a população para aplicação dos operadores genéticos nas execuções seguintes.

Dentro do laço de gerações é aplicado o operador genético *crossover*, que criará uma nova geração por meio do cruzamento dos indivíduos da geração atual. O algoritmo usa a técnica do elitismo, onde o indivíduo mais adaptado, com melhor *fitness*, é sempre mantido, sendo incluído na nova população. Todos os outros indivíduos da população serão substituídos por novos indivíduos criados por cruzamento. A seleção dos indivíduos que sofrem cruzamento é

feita conforme definido por argumento da função entre as opções: *ranking*, *tournament* de dois e *roulette*. Em qualquer dos métodos de seleção, é aplicado o cruzamento também definido por argumento da função entre as opções: *uniform*, *average* e BLX, como descrito na [Seção 2.7 \(Algoritmos Genéticos\)](#). Desta forma, são possíveis nove combinações de métodos de seleção e cruzamento. Na execução do experimento deste trabalho são exploradas as nove combinações.

Na sequência, o próximo passo é aplicar o operador genético de mutação a todos os indivíduos, exceto o de melhor *fitness*. O número de genes a serem mutados é escolhido aleatoriamente entre 1 e 5, com as probabilidades de 0,5 para 1 gene; 0,2 para 2; 0,15 para 3; 0,1 para 4 e 0,05 para 5 genes. Cada gene que sofre mutação tem seu valor alterado para mais ou para menos em um montante entre 1% e 30% da sua faixa de valores possíveis. Tanto o montante da mutação quanto o sinal (+ ou -) são escolhidos aleatoriamente. O código que implementa os operadores genéticos foi nomeado como “Operators.R” e está listado no [Código-fonte 6](#), no [Apêndice A](#).

Após criada a nova geração da população, o AG calcula o *fitness* dos indivíduos, representando quão bem eles estão “adaptados” ao problema do seu ambiente, ou seja, a qualidade da solução que eles representam. Neste AG, o *fitness* de um indivíduo representa quanto a previsão de sobrevivência feita pelo TRISS, usando os coeficientes a que correspondem os genes do indivíduo, se aproxima, para o conjunto de todos os pacientes, do resultado (sobrevivência ou não) indicado no banco de dados, ou seja, quanto a solução representada pelo indivíduo se ajusta ao dados de pacientes.

O valor do *fitness* foi calculado com base na quantidade de acertos da previsão de sobrevivência com auxílio de análise de curvas *Receiver Operating Characteristic*, um método para avaliação de testes diagnósticos e também para medir a capacidade de um modelo de risco clínico em discriminar entre óbitos hospitalares e sobreviventes abordado mais adiante no [Capítulo 4 \(Resultados\)](#). O cálculo do *fitness* de um indivíduo foi, então, implementado de neste AG como um valor entre 0 e 1 que corresponde à quantidade de acertos do TRISS-AG ao prever a sobrevivência dos pacientes dividido pela quantidade total de pacientes, usando os genes de um indivíduo como argumento. Considerou-se um acerto quando o índice calculado pelo TRISS-AG para um dado paciente foi superior a um determinado *threshold*, ou limite de corte, e o paciente sobreviveu, ou seja, um sobrevivente estimado corretamente, ou também quando o índice foi inferior ao limite de corte e o paciente não sobreviveu, ou seja, uma morte estimada corretamente. Para determinar o limite de corte, o TRISS-AG foi calculado para todos os pacientes e foi construída uma curva *Receiver Operating Characteristic* com estes índices e com os dados de sobrevivência indicados no banco de dados de pacientes, escolhendo-se como limite o índice correspondente ao ponto da curva que apresenta a melhor combinação de especificidade e sensibilidade, ou seja, o ponto que representa o limite de corte que apresente o máximo de verdadeiros positivos e verdadeiros negativos e o mínimo de falsos positivos e falsos negativos. O ponto de corte foi calculado usando as funções *coords* e *roc* do pacote *pROC*

(ROBIN *et al.*, 2011). Neste método de cálculo, valores de *fitness* são tão melhores quanto mais se aproximam de 1.

O código que implementa a função de *fitness* foi nomeado como “Fitness.R” e está listado no Código-fonte 4, no Apêndice A. O código que implementa o cálculo do TRISS e do RTS, como descrito na Seção 2.6 (*Trauma and Injury Severity Score (TRISS)*), foi nomeado como “Scores.R” e está listado no Código-fonte 8, no mesmo Apêndice A.

A execução do cálculo do *fitness* para todos os indivíduos em cada geração é a operação que mais exige tempo e processamento computacional em todo o algoritmo. Por isso, o código implementado faz uso de computação paralela de forma a acelerar a sua execução. O cálculo do *fitness* de cada indivíduo é lançado para execução em um *thread* diferente e estas são distribuídas entre todos os *cores* lógicos disponíveis no microprocessador do computador. Para tanto, o código usa a biblioteca *parallels* nativa do R. São usadas as funções *clusterApply* ou *mclapply*, conforme o sistema operacional em que o código está sendo executado, Windows^{®1} ou macOS^{®2} / Linux^{®3}, respectivamente, detectado automaticamente. São usadas funções diferentes pois a primeira tem melhor desempenho, porém é suportada apenas no sistema operacional Windows[®].

Com objetivo principal de reduzir o risco de *overfitting*, mas também melhorar sua velocidade de execução, o código implementa a técnica conhecida como *Random Sampling Technique* (RST). O *overfitting* ocorre quando o algoritmo converge de forma tão específica para os dados com que foi treinado, que ele não consegue prever de forma confiável resultados para observações futuras diferentes dos dados de treinamento. O RST consiste em escolher aleatoriamente um subconjunto diferente dos dados de treinamento do algoritmo a cada geração, ao invés de usar o banco de dados completo. Dessa forma, apenas soluções com bom desempenho em vários subconjuntos diferentes de dados permanecerão na população após várias gerações. Com esse desempenho superior, os indivíduos sobreviventes têm maior probabilidade de capturar os relacionamentos subjacentes dos dados ao invés de incorrer em *overfitting* (LIU; KHOSH-GOFTAAR, 2004). Esta técnica é efetiva tanto para controlar o *overfitting* quanto para melhorar a velocidade de execução, devido redução da quantidade de cálculos (GONÇALVES, 2011). Recomenda-se que o tamanho do subconjunto dos dados seja reduzido, com bons resultados alcançados com subconjuntos tão pequenos quanto 1% do banco de dados (GONÇALVES; SILVA, 2011) (GONÇALVES *et al.*, 2012).

Na sequência, após o cálculo do *fitness*, o código aplica a estratégia da predação desenvolvida por Simões (2000) a cada vez que um número determinado de gerações é processado. A predação é uma estratégia eficiente, capaz de melhorar o desempenho do algoritmo eliminando indivíduos pouco adaptados (ou seja, com menor *fitness*), substituindo-os por novos indivíduos com genes gerados aleatoriamente de forma a trazer mais diversidade à população. Em uma

¹ Windows é uma marca registrada da Microsoft Corporation

² macOS é uma marca registrada da Apple Inc.

³ Linux é uma marca registrada da The Linux Foundation

analogia à natureza, a população pode sofrer ataques regulares de um “predador” que seleciona o pior indivíduo da geração especificada e o destrói (“mata”), abrindo espaço na população para migração de novos indivíduos, trazendo mais diversidade genética para o grupo. A frequência dos ataques é um fator importante, pois gerações suficientes devem ser preservadas para permitir que a população se recupere dos ataques. De forma a acelerar a absorção desta diversidade, o algoritmo garante que o novo indivíduo que substitui o predado será selecionado para cruzamento com o melhor indivíduo na próxima geração (SIMÕES; BARONE, 2002).

Uma alternativa de substituição do indivíduo com menor *fitness*, chamada de síntese, é também implementada pelo algoritmo. Ela consiste em substituir, a cada geração, o indivíduo menos adaptado por um novo, cujos genes correspondam à média dos genes correspondentes dos demais indivíduos da geração. A síntese pode melhorar o desempenho do algoritmo, permitindo a convergência com menos gerações, especialmente com populações com poucos indivíduos e altas taxas de mutação (CROCOMO; SIMÕES; DELBEM, 2004) (CROCOMO, 2008).

Com a população da nova geração completamente definida após a aplicação de todas as técnicas e estratégias descritas, o algoritmo ordena os indivíduos da nova população pelo seu *fitness* em ordem decrescente, de forma a prepará-la para a repetição do processo na próxima repetição do laço de gerações.

Neste ponto, o algoritmo implementa a técnica de mutação variável. Esta técnica consiste em aumentar sucessivamente a amplitude da mutação aplicada ao se processar uma nova geração quando o *fitness* do melhor indivíduo não apresentar uma melhoria significativa (acima de um dado *threshold*) após algumas gerações, o que pode denotar que o algoritmo está aprisionado em um ótimo local e não consegue explorar o espaço de busca na tentativa de encontrar outras soluções potencialmente melhores. Desta forma, a mutação com amplitude maior permitirá ao algoritmo se distanciar do ótimo local e seguir a busca (HARVEY, 1993). Neste algoritmo, ajusta-se um multiplicador a ser aplicado ao montante de mutação na próxima geração, dobrando o mesmo após decorridas cinco gerações sem que o *fitness* do melhor indivíduo tenha melhorado além do *threshold* estabelecido como argumento. O ajuste pode ocorrer seguidamente se necessário, dobrando novamente o multiplicador de mutação a cada cinco gerações até atingir o valor de 32. Se ainda assim o *threshold* não for superado, como medida extrema para aumentar a diversidade da população e escapar do ótimo local, o algoritmo executa um genocídio, ou mutação cataclísmica, em que toda a população, exceto o melhor indivíduo, é substituída por novos indivíduos com genes definidos aleatoriamente (WHITLEY; SUTTON, 2012).

Finalizando o laço de gerações, o algoritmo coleta estatísticas para permitir análises posteriores de sua execução. Após a execução do número de gerações determinado por argumento, o algoritmo constrói a estrutura de dados que sua função principal retorna, contendo todo o histórico de execução, todas as gerações, estatísticas e os argumentos passados à mesma.

Para a execução dos experimentos deste trabalho, alguns códigos-fonte auxiliares foram desenvolvidos. O principal deles tem como objetivo executar o AG com as nove diferentes combi-

nações entre os três métodos de seleção e os três métodos de cruzamento que este implementa, de forma a explorar as diferentes combinações e comparar seus resultados. Adicionalmente, também tem o objetivo de centralizar e alimentar o AG com os argumentos usados no experimento. Ele foi nomeado como “Executar.R” e está listado no [Código-fonte 2](#), no [Apêndice A](#).

Um segundo código-fonte auxiliar foi desenvolvido para plotar gráficos com a evolução do *fitness* durante a execução do AG e gráficos comparando os resultados entre as diferentes variações de métodos de seleção e cruzamento. Ele foi nomeado como “Plot.R” e está listado no [Código-fonte 7](#), no [Apêndice A](#).

Finalmente, um último código-fonte foi desenvolvido para armazenar a sequência de passos seguida nos experimentos deste trabalho. Ele é o ponto de partida para a execução e repetição destes. Ele foi nomeado como “Experimento.R” e está listado no [Código-fonte 3](#), no [Apêndice A](#).

Além de serem reproduzidos no apêndice, os códigos-fonte foram armazenados na ferramenta de controle de versões GitHub e o repositório está disponível em <https://github.com/andre-ennes/Algoritmo-Genetico-TRISS>.

RESULTADOS

Na presente seção são apresentados os resultados dos experimentos com a aplicação do algoritmo genético nos bancos de dados de pacientes.

4.1 Aplicação do TRISS aos Banco de Dados

Foram gerados cinco bancos de dados que representam pacientes aleatórios que seriam atendidos por cinco hospitais de características de qualidade de atendimento diferentes, como descrito no [Capítulo 1 \(Introdução\)](#). Os bancos de dados foram nomeados da seguinte forma: BD-A, representando dados de pacientes de um hospital norte-americano; BD-B10, representando um hospital brasileiro de boa qualidade; BD-B20, representando um hospital brasileiro de menor qualidade; BD-B30, representando um hospital brasileiro de qualidade ainda menor; e BD-B50, representando um hospital brasileiro de qualidade inferior. A nomenclatura adotada para os bancos de dados está resumida na [Tabela 2 \(Nomes dos bancos de dados de pacientes, Página 28\)](#).

A primeira hipótese que se pretendia testar na presente pesquisa era a de que o método TRISS apresentaria maior taxa de erro para os bancos de dados que representam serviços de saúde de menor qualidade, mostrando que o método criado com dados norte-americanos apresenta problemas ao ser usado para realidades diversas, como a brasileira.

Foi, então, feita uma avaliação do erro médio entre a estimativa de sobrevivência feita usando o método TRISS e o índice de sobrevivência apresentado pelos cinco bancos de dados. Inicialmente, uma comparação simples foi feita usando o Erro Quadrático Médio (EQM), dado pela [Equação 4.1](#).

$$EQM(\hat{\theta}_i) = \frac{1}{N} \sum_{i=1}^N (\hat{\theta}_i - \theta_i)^2 \quad (4.1)$$

Os valores de EQM da aplicação do TRISS nos cinco bancos de dados estão na [Tabela 10](#).

Nota-se que o EQM aumenta progressivamente conforme cai a qualidade de atendimento que cada banco de dados representa.

Tabela 10 – EQM aplicando TRISS nos dos bancos de dados

Nome	EQM
BD-A	0,0628572
BD-B10	0,0780909
BD-B20	0,1116578
BD-B30	0,1716629
BD-B50	0,2811738

Entretanto, o EQM não é o melhor método para avaliar a qualidade do ajuste (ou *goodness of fit*) de uma regressão logística, como é o caso do TRISS (HOSMER; LEMESHOW, 2000). Por outro lado, há uma extensa literatura sobre o uso de curvas *Receiver Operating Characteristic* (ROC) pela comunidade médica como avaliação de testes diagnósticos (FAWCETT, 2006).

A análise de curvas ROC é um método útil para medir a capacidade de um modelo de risco clínico em discriminar entre óbitos hospitalares e sobreviventes. Seu uso na medicina se originou como um método para sintetizar a especificidade e a sensibilidade de testes diagnósticos distribuídas em um espectro de possíveis pontos de corte. A área sob a curva, ou *Area Under Curve* (AUC), pode ser interpretada como a probabilidade de classificação ou previsão correta (GRUNKEMEIER; JIN, 2001).

$$\text{sensibilidade} = \frac{\text{Positivos corretamente classificados (verdadeiro-positivos)}}{\text{Total de positivos de fato}}$$

$$\text{especificidade} = \frac{\text{Negativos corretamente classificados (verdadeiro-negativos)}}{\text{Total de negativos de fato}} \quad (4.2)$$

$$\text{especificidade} = 1 - \frac{\text{Positivos incorretamente classificados (falso-positivos)}}{\text{Total de negativos de fato}}$$

Para interpretar curvas ROC, é importante entender os conceitos de sensibilidade e especificidade de um teste diagnóstico, definidos na Equação 4.2. A sensibilidade é definida como a probabilidade de um teste resultar positivo dado que o caso seja realmente positivo, ou seja, a sensibilidade é a proporção de resultados verdadeiro-positivos com relação aos casos positivos de fato. Por outro lado, a especificidade é a probabilidade de um teste resultar negativo, dado que o caso seja negativo de fato, ou seja, a especificidade é a proporção de resultados verdadeiro-negativos com relação aos casos negativos de fato. A especificidade corresponde também à probabilidade inversa de um resultado ser positivo dado que o caso não seja de fato, ou seja, a proporção de falso-positivos com relação aos casos negativos de fato (FAWCETT, 2006).

Usamos curvas ROC para fazer uma avaliação global da precisão de um teste diagnóstico por meio do cálculo da área sob a curva, ou AUC. Os valores da AUC podem variar de 0 a 1 e valores maiores indicam que a precisão do teste diagnóstico é melhor. A curva ROC é uma representação gráfica de sensibilidade contra especificidade para todos os valores de corte possíveis de um dado teste. Há sempre um *trade-off* entre sensibilidade e especificidade quando definimos, para testes quantitativos, um valor de corte que determina o ponto que separa um resultado positivo de um resultado negativo. Se um novo teste fosse perfeito, haveria uma separação completa de valores entre pacientes com e sem a doença, o valor de corte seria o menor valor em pacientes com a doença e a AUC seria igual a 1. No entanto, como não há testes perfeitos, sempre haverá resultados falso-positivos ou falso-negativos. Quanto mais preciso for um teste, maior será a AUC, que é a probabilidade de que o valor da medida seja maior em uma pessoa qualquer com a doença do que em uma pessoa qualquer sem a doença (FERREIRA; PATINO, 2018).

Figura 3 – Curvas ROC aplicando TRISS nos bancos de dados

Foi feita, então, a análise de curvas ROC para os cinco bancos de dados gerados, como apresentado no gráfico da Figura 3. De acordo com a primeira hipótese desta pesquisa, a aplicação do método TRISS sobre os bancos de dados que representam serviços de saúde de

menor qualidade deveria apresentar menos precisão. O comparativo dos valores de AUC para cada banco de dados é apresentado na [Tabela 11](#).

Tabela 11 – Valores de AUC aplicando TRISS nos bancos de dados

Nome	AUC
BD-A	0,9863906
BD-B10	0,9854750
BD-B20	0,9756289
BD-B30	0,9505311
BD-B50	0,9105654

Nota-se que o valor de AUC é progressivamente menor para os bancos de dados que representam qualidades de atendimento respectivamente inferiores. Uma redução equivalente na precisão do método TRISS é notável nas curvas ROC dos respectivos bancos de dados, mostrando que o método criado com dados norte-americanos apresenta problemas ao ser usado para realidades diversas, como seria o caso de um hospital brasileiro.

4.2 Aplicação do TRISS-AG e comparação com o TRISS

Foi desenvolvido um AG, chamado de TRISS-AG, que ajusta os coeficientes da equação do TRISS utilizando para treinamento um banco de dados de pacientes no formato dos cinco bancos de dados já descritos, como detalhado no [Capítulo 3 \(Algoritmo e Implementação\)](#). O TRISS-AG foi executado com os cinco bancos de dados, gerando novas equações como a do TRISS, porém com coeficientes ajustados. Como o TRISS-AG foi implementado com três opções de operador de seleção e três opções de operador de cruzamento, cada execução do mesmo para cada um dos cinco bancos de dados ajustou nove variações da equação do TRISS, correspondentes às diferentes combinações de operadores de seleção e cruzamento, totalizando então 45 ajustes. Comparando-se os resultados das execuções, é possível determinar qual combinação de operadores apresentou o melhor desempenho e selecioná-la como a que gerou o melhor ajuste. Foi considerado como medida de desempenho o valor do *fitness* mais alto alcançado pelo melhor indivíduo em uma execução do AG com uma determinada combinação de operadores.

O cálculo do *fitness* foi feito dividindo-se a quantidade de acertos da previsão de sobrevivência pela quantidade total de pacientes, com auxílio de curvas ROC para determinar o valor de corte que define uma previsão de sobrevivência ou não. Considerou-se um acerto quando o índice calculado pelo TRISS-AG para um dado paciente foi superior ao valor de corte e o paciente sobreviveu, ou seja, um sobrevivente estimado corretamente, ou também quando o índice foi inferior ao valor de corte e o paciente não sobreviveu, ou seja, uma morte estimada corretamente. Para determinar o limite de corte, o TRISS-AG foi calculado para todos os pacientes e foi construída uma curva ROC com estes índices e com os dados de sobrevivência indicados no

banco de dados de pacientes, escolhendo-se como valor de corte o índice correspondente ao ponto da curva que apresenta a melhor combinação de especificidade e sensibilidade.

Após a execução do TRISS-AG com as nove variações de operadores para os cinco bancos de dados, foram obtidos os resultados de desempenho ilustrados nos gráficos a seguir. Cada gráfico representa a evolução do *fitness* do melhor indivíduo em cada geração do AG de todas as variações de operadores para um determinado banco de dados, comparando-as. O gráfico da [Figura 4](#) corresponde aos resultados da execução do TRISS-AG para o BD-A, o da [Figura 5](#) para o BD-B10, o da [Figura 6](#) para o BD-B20, o da [Figura 7](#) para o BD-B30 e o da [Figura 8](#) para o BD-B50.

Figura 4 – Evolução do *fitness* máximo do TRISS-AG por variação de operadores para o BD-A

A partir dos resultados, é possível notar uma prevalência da variação do AG que combina o operador de *crossover* BLX com o operador de seleção *ranking* com elitismo (abreviado no gráfico somente como “elitismo” por simplicidade) como melhor *fitness* na geração final para a maioria dos bancos de dados. Por isso, esta combinação foi considerada como a de melhor desempenho e foi selecionada para aplicação nos experimentos subsequentes. Os indivíduos com melhor *fitness* ao final da execução do algoritmo com esta combinação para cada banco de dados representam as novas equações do TRISS calculadas pelo TRISS-AG. Os coeficientes destas equações são listados na [Tabela 12](#).

Figura 5 – Evolução do *fitness* máximo do TRISS-AG por variação de operadores para o BD-B10 Figura 6 – Evolução do *fitness* máximo do TRISS-AG por variação de operadores para o BD-B20

Figura 7 – Evolução do *fitness* máximo do TRISS-AG por variação de operadores para o BD-B30 Figura 8 – Evolução do *fitness* máximo do TRISS-AG por variação de operadores para o BD-B50

Tabela 12 – Coeficientes das novas equações do TRISS calculados pelo TRISS-AG

Componente do Cálculo do TRISS	BD-A	BD-B10	BD-B20	BD-B30	BD-B50
Limite de idade do parâmetro AGE	54,0855803	54,5357100	54,2735165	54,63103259	37,6655866
Coeficiente b_0 no cálculo do TRISS	1,7447702	0,8416545	10,4386258	0,46834332	3,0099986
Coeficiente de RTS no cálculo do TRISS	2,8479108	3,4901629	1,4802570	1,27165858	3,6471711
Coeficiente de ISS no cálculo do TRISS	-0,2183383	-0,2888133	-0,1291247	-0,07279626	-0,3030213
Coeficiente de AGE no cálculo do TRISS	-5,7412552	-7,4827204	-3,4652974	-1,83935300	-24,1161002

A evolução do *fitness* a cada geração durante a execução do TRISS-AG que gerou as equações acima, usando a combinação de operadores selecionada, é ilustrada nos gráficos a seguir. Cada gráfico representa a evolução do *fitness* do melhor indivíduo, do pior indivíduo, do *fitness* médio da população e do desvio padrão a cada geração do AG para um dado banco de dados. O gráfico da Figura 9 corresponde à evolução durante a execução do TRISS-AG para o BD-A, o da Figura 10 para o BD-B10, o da Figura 11 para o BD-B20, o da Figura 12 para o BD-B30 e o da Figura 13 para o BD-B50. É possível notar nos gráficos que o AG converge rapidamente, em menos de 20 gerações, para depois manter um patamar de estabilidade com pouca evolução adicional, mesmo após 100 gerações. Nota-se também o efeito da estratégia de predação que ocorre a cada cinco gerações, reduzindo o *fitness* do pior indivíduo quando o substitui por um novo com genes aleatórios para aumentar a diversidade genética da população.

Figura 9 – Evolução do *fitness* durante a execução do TRISS-AG para o BD-A

Figura 10 – Evolução do *fitness* durante a execução do TRISS-AG para o BD-B10 Figura 11 – Evolução do *fitness* durante a execução do TRISS-AG para o BD-B20

Figura 12 – Evolução do *fitness* durante a execução do TRISS-AG para o BD-B30 Figura 13 – Evolução do *fitness* durante a execução do TRISS-AG para o BD-B50

Após as novas equações TRISS serem ajustadas pelo TRISS-AG, seguiu-se com a avaliação do erro entre a estimativa de sobrevivência feita por estas novas equações e o índice de sobrevivência apresentado pelos cinco bancos de dados.

A segunda hipótese que se pretendia testar nesta pesquisa é a de que o TRISS-AG apresentaria menor erro na estimativa de sobrevivência dos pacientes quando comparado à equação original do TRISS, mostrando que é possível usar um AG para ajustar as equações do TRISS e que elas poderiam prever melhor a sobrevivência dos pacientes ao levarem em conta a realidade dos serviços de saúde que originaram os dados, especialmente quando são diferentes da realidade norte-americana em que se baseou o método original, como é o caso dos serviços de saúde brasileiros.

Foi, então, feita uma comparação da quantidade de acertos nas estimativas de sobrevivência feitas pelo TRISS e pelo TRISS-AG para cada um dos cinco bancos de dados gerados, usando o mesmo método adotado para o cálculo do *fitness*. O gráfico da [Figura 14](#) apresenta a comparação para o BD-A com a evolução da quantidade de acertos durante a execução do AG, da mesma forma que a [Figura 15](#) para o BD-B10, a [Figura 16](#) para o BD-B20, a [Figura 17](#) para o BD-B30 e a [Figura 18](#) para o BD-B50. Nota-se que a quantidade final de acertos do TRISS-AG supera a do TRISS levemente para os bancos de dados BD-A, BD-B10 e BD-B20, contudo supera de forma relevante para os bancos de dados BD-B30 e BD-B50, com grande vantagem para este último.

Figura 14 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-A

Figura 15 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B10

Figura 16 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B20

Figura 17 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B30

Figura 18 – Quantidade de Acertos do TRISS e do TRISS-AG no BD-B50

A comparação da quantidade final de acertos nas estimativas de sobrevivência feitas pelo TRISS e pelo TRISS-AG para cada um dos cinco bancos de dados gerados é apresentada na [Tabela 13](#).

Tabela 13 – Quantidade de acertos do TRISS e do TRISS-AG para os bancos de dados

Nome	Acertos TRISS	Acertos TRISS-AG	Diferença	%
BD-A	18.676	18.709	+33	+0,18%
BD-B10	18.717	18.747	+30	+0,16%
BD-B20	18.397	18.468	+71	+0,39%
BD-B30	17.340	17.665	+325	+1,87%
BD-B50	15.766	17.410	+1.644	+10,43%

Para confirmar a segunda hipótese desta pesquisa, as quantidades de acertos obtidas pelo TRISS-AG deveriam ser maiores do que as obtidas pelo TRISS original. De fato, os valores obtidos pelo TRISS-AG foram superiores para todos os bancos de dados, mostrando que é possível usar um AG para ajustar as equações do TRISS e que ele pode prever melhor a sobrevivência dos pacientes. A diferença foi maior para os bancos de dados que representam os piores serviços de saúde, mostrando também que a previsão de sobrevivência do AG é ainda melhor em comparação ao TRISS original quando os dados representam realidades diferentes da norte-americana.

CONCLUSÃO

Os índices de classificação de trauma têm um papel chave no atendimento médico e na melhoria de qualidade dos serviços de saúde, tanto para guiar decisões clínicas no momento do atendimento como posteriormente para avaliar o desempenho do serviço. Eles permitem mensurar a gravidade do paciente e estimar sua probabilidade de sobrevivência a partir de modelos matemáticos que consideram os sinais vitais e avaliações feitas pelos médicos a respeito do seu estado. Um índice muito usado é o TRISS, que combina características anatômicas, fisiológicas e a idade do paciente para estimar sua probabilidade de sobrevivência e que foi construído por meio de uma regressão logística em um estudo dos anos 80 com dados de pacientes de trauma dos EUA e do Canadá. Embora amplamente aceito, o TRISS tem sido criticado por não refletir corretamente a realidade de países em desenvolvimento, uma vez que se baseou em dados de países desenvolvidos que contam com melhores sistemas de saúde. Estudos tem sido feitos buscando adaptar o TRISS à realidade de países como o Brasil, usando dados locais e técnicas alternativas, incluindo inteligência computacional. Neste sentido, o presente trabalho buscou propor uma abordagem para ajustar os coeficientes do método TRISS aplicando algoritmos genéticos, procurando testar se é possível utilizar tal técnica neste contexto e se por meio desta abordagem é possível obter melhor desempenho na previsão de sobrevivência dos pacientes em comparação ao método TRISS original, especialmente quando é aplicado sobre dados que representam uma realidade diferente da norte-americana.

Foram sintetizados cinco bancos de dados representando pacientes aleatórios que seriam atendidos por cinco hospitais, sendo um representante de um hospital americano e outros quatro de hospitais brasileiros, com níveis de qualidade de atendimento progressivamente piores. A sobrevivência dos pacientes foi estimada usando o TRISS, mas reduzindo-se a probabilidade respectivamente para cada banco de dados. O desempenho do TRISS original em estimar sobrevivência para todos os bancos de dados foi, então, medido e comparado usando curvas ROC.

Um algoritmo genético, chamado de TRISS-AG, foi desenvolvido para ajustar novos coeficientes para a equação do TRISS usando os dados dos pacientes para treinamento de forma a prever sobrevivência com mais precisão quando são usados dados de realidades diversas da que originou o TRISS. O desempenho do TRISS-AG foi comparado com o do TRISS quando aplicados aos cinco bancos de dados gerados.

Esta pesquisa pretendeu validar duas hipóteses durante seu desenvolvimento. A primeira foi a de que o método TRISS apresentaria maior taxa de erro para os bancos de dados que representam serviços de saúde de menor qualidade. A segunda hipótese foi a de que o TRISS-AG apresentaria maior precisão na estimativa de sobrevivência dos pacientes quando comparado ao TRISS.

Os resultados confirmaram a primeira hipótese. O desempenho do TRISS medido por meio de curvas ROC se mostrou progressivamente menor para os bancos de dados que representam qualidades de atendimento respectivamente inferiores, mostrando que o método criado com dados norte-americanos apresenta problemas ao ser usado para realidades diversas, como a brasileira.

A segunda hipótese também foi confirmada pelos resultados. Eles mostraram que é, sim, possível usar um AG para ajustar a equação do TRISS e que o seu desempenho medido pela quantidade de acertos na previsão de sobrevivência foi superior à do TRISS original, especialmente quando os dados representam realidades de atendimento de saúde diferentes da norte-americana.

Outros estudos similares a este foram desenvolvidos simultaneamente por outros pesquisadores usando Inteligência Artificial (IA) como alternativa ao TRISS original. [Rau et al. \(2019\)](#) conduziu um estudo usando dados reais que comparou o TRISS com regressão logística, *support vector machines* e redes neurais e concluiu que os quatro modelos apresentaram desempenho equivalente. Outro estudo feito por [Kim et al. \(2018\)](#) também com dados reais comparou o TRISS com regressão logística, *random forest* e redes neurais, também concluindo que eles apresentaram desempenho comparável. Neste trabalho foi alcançado um desempenho superior do AG comparado ao TRISS original.

As pesquisas citadas reforçam a relevância do tema tratado neste trabalho. O TRISS tem resistido há mais de três décadas como referência na previsão de sobrevivência, apesar das críticas à sua confiabilidade. Desde a sua criação, técnicas de IA vem sendo desenvolvidas e aplicadas com sucesso a uma infinidade de conjuntos de dados e tem grande potencial para melhorar ou substituir o TRISS com vantagens. Este trabalho apresenta sua contribuição na forma de uma nova tentativa neste sentido, tendo tido sucesso em evidenciar deficiências do TRISS e alcançado um desempenho superior.

É importante reforçar que os resultados e conclusões apresentados se basearam em experimentos utilizando bancos de dados sintetizados de atendimento de pacientes de trauma que

simulam os cenários planejados para os experimentos, dada a dificuldade em se obter dados reais de atendimento de pacientes por questões de privacidade e disponibilidade dos mesmos. Os dados de teste foram gerados utilizando o próprio método TRISS como preditor de sobrevivência. A escolha desta abordagem viabilizou a realização do estudo, mas implica na necessidade de se validar os resultados e conclusões obtidos por meio da repetição dos experimentos utilizando dados reais. Esta é, portanto, uma importante limitação deste trabalho.

Fica clara, então, a oportunidade de se desenvolver trabalhos futuros utilizando dados reais de atendimentos de pacientes de trauma, tanto repetindo o presente estudo como elaborando outros, superando assim a sua limitação mais importante. Trabalhos futuros poderiam também investigar variações nos parâmetros do AG e o uso de outros algoritmos ou outras técnicas de inteligência computacional.

REFERÊNCIAS

BOYD, C. R.; TOLSON, M. A.; COPEL, W. S. Evaluating trauma care: the TRISS method. **The Journal of Trauma**, v. 27, n. 4, p. 370–378, abr. 1987. ISSN 0022-5282. Disponível em: <http://europepmc.org/abstract/MED/3106646>. Citado nas páginas 35 e 37.

BROOKS, R. From robot dreams to reality. **Nature**, Springer Science and Business Media LLC, v. 406, n. 6799, p. 945–947, ago. 2000. Disponível em: <https://doi.org/10.1038/35023200>. Citado na página 39.

CHAMPION, H. R.; COPEL, W. S.; SACCO, W. J.; LAWNICK, M. M.; KEAST, S. L.; BAIN, L. W.; FLANAGAN, M. E.; FREY, C. F. The major trauma outcome study: establishing national norms for trauma care. **The Journal of Trauma**, v. 30, n. 11, p. 1356–1365, nov. 1990. ISSN 0022-5282. Disponível em: <http://europepmc.org/abstract/MED/2231804>. Citado nas páginas 26, 35, 36 e 37.

CHAMPION, H. R.; SACCO, W. J.; COPEL, W. S.; GANN, D. S.; GENNARELLI, T. A.; FLANAGAN, M. E. A revision of the trauma score. **The Journal of Trauma: Injury, Infection, and Critical Care**, Ovid Technologies (Wolters Kluwer Health), v. 29, n. 5, p. 623–629, maio 1989. Disponível em: <https://doi.org/10.1097/00005373-198905000-00017>. Citado nas páginas 34 e 35.

CHRISTIE, S. A.; HUBBARD, A. E.; CALLCUT, R. A.; HAMEED, M.; DISSAK-DELON, F. N.; MEKOLO, D.; SAIDOU, A.; MEFIRE, A. C.; NSONGOO, P.; DICKER, R. A.; COHEN, M. J.; JUILLARD, C. Machine learning without borders? An adaptable tool to optimize mortality prediction in diverse clinical settings. **Journal of Trauma and Acute Care Surgery**, Ovid Technologies (Wolters Kluwer Health), v. 85, n. 5, p. 921–927, nov. 2018. Disponível em: <https://doi.org/10.1097/ta.0000000000002044>. Citado na página 26.

CLIFF, D.; HARVEY, I.; HUSBANDS, P. Artificial evolution of visual control systems for robots. In: SRINIVASAN, M.; VENKATESH, S. (Ed.). **From Living Eyes to Seeing Machines**. Oxford University Press, 1997. p. 126–157. ISBN 978-0-19-857785-0. Disponível em: <https://dl.acm.org/doi/book/10.5555/549645>. Citado na página 42.

CROCOMO, M. K. **Um algoritmo evolutivo para aprendizado on-line em jogos eletrônicos**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2008. Citado na página 48.

CROCOMO, M. K.; SIMÕES, E. d. V.; DELBEM, A. C. B. Otimização automática de parâmetros de sistemas evolutivos de acordo com a aplicação. In: **Congresso Brasileiro de Computação**. [S.l.: s.n.], 2004. Citado na página 48.

DOMINGUES, C. d. A.; COIMBRA, R.; POGGETTI, R. S.; NOGUEIRA, L. d. S.; SOUSA, R. M. C. d. Performance of new adjustments to the TRISS equation model in developed and developing countries. **World Journal of Emergency Surgery**, Springer Nature, v. 12, n. 1, mar. 2017. Disponível em: <https://doi.org/10.1186/s13017-017-0129-2>. Citado nas páginas 26 e 32.

_____. New Trauma and Injury Severity Score (TRISS) adjustments for survival prediction. **World Journal of Emergency Surgery**, Springer Nature, v. 13, n. 1, mar. 2018. Disponível em: <<https://doi.org/10.1186/s13017-018-0171-8>>. Citado na página 26.

DOMINGUES, C. d. A.; NOGUEIRA, L. d. S.; SETTERVALL, C. H. C.; SOUSA, R. M. C. d. Desempenho dos ajustes do Trauma and Injury Severity Score (TRISS): revisão integrativa. **Revista da Escola de Enfermagem da USP**, FapUNIFESP (SciELO), v. 49, n. spe, p. 138–146, dez. 2015. Disponível em: <<https://doi.org/10.1590/s0080-623420150000700020>>. Citado nas páginas 25, 35 e 36.

ESHELMAN, L. J.; SCHAFFER, J. D. Real-coded genetic algorithms and interval-schemata. In: **Foundations of Genetic Algorithms**. Elsevier, 1993. p. 187–202. Disponível em: <<https://doi.org/10.1016/b978-0-08-094832-4.50018-0>>. Citado na página 41.

FANI-SALEK, M. H.; TOTTEN, V. Y.; TEREZAKIS, S. A. Trauma scoring systems explained. **Emergency Medicine Australasia**, Wiley, v. 11, n. 3, p. 155–166, set. 1999. Disponível em: <<https://doi.org/10.1046/j.1442-2026.1999.00039.x>>. Citado nas páginas 25, 26, 31, 32, 33, 34, 35, 36 e 37.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, Elsevier BV, v. 27, n. 8, p. 861–874, jun. 2006. Disponível em: <<https://doi.org/10.1016/j.patrec.2005.10.010>>. Citado na página 52.

FERREIRA, J. C.; PATINO, C. M. Understanding diagnostic tests (part 3). **Jornal Brasileiro de Pneumologia**, FapUNIFESP (SciELO), v. 44, n. 1, p. 4–4, fev. 2018. Disponível em: <<https://doi.org/10.1590/s1806-37562018000000017>>. Citado na página 53.

GONÇALVES, I.; SILVA, S. Experiments on controlling overfitting in genetic programming. In: **Proceedings of the 15th Portuguese Conference on Artificial Intelligence, EPIA 2011**. Lisbon, Portugal: [s.n.], 2011. Disponível em: <<https://www2.cisuc.uc.pt/download-file/809/qb4R3ZlgoZwi1rtuAZhC>>. Citado na página 47.

GONÇALVES, I.; SILVA, S.; MELO, J. B.; CARREIRAS, J. M. B. Random sampling technique for overfitting control in genetic programming. In: **Lecture Notes in Computer Science**. Springer Berlin Heidelberg, 2012. p. 218–229. Disponível em: <https://doi.org/10.1007/978-3-642-29139-5_19>. Citado na página 47.

GONÇALVES, I. C. P. **Controlling overfitting in genetic programming**. Dissertação (Mestrado) — Universidade de Coimbra, Portugal, 2011. Disponível em: <https://www.cisuc.uc.pt/publication/showfile?fn=1335718367_Ivo_Goncalves_MSc_Dissertation.pdf>. Citado na página 47.

GRUNKEMEIER, G. L.; JIN, R. Receiver Operating Characteristic curve analysis of clinical risk models. **The Annals of Thoracic Surgery**, Elsevier BV, v. 72, n. 2, p. 323–326, ago. 2001. Disponível em: <[https://doi.org/10.1016/s0003-4975\(01\)02870-3](https://doi.org/10.1016/s0003-4975(01)02870-3)>. Citado na página 52.

HARVEY, I. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In: LANGTON, C. (Ed.). **In Artificial Life III**. Addison-Wesley, 1993. p. 299–326. Disponível em: <<http://users.sussex.ac.uk/~inmanh/csrp222.pdf>>. Citado nas páginas 40, 42 e 48.

_____. Artificial evolution for real problems. In: GOMI, T. (Ed.). **Evolutionary Robotics**. [S.l.]: AAI Books, Ontario, Canada, 1997. p. 187–220. ISBN 0-9698872-1-3. Citado na página 42.

HARVEY, I.; THOMPSON, A. Through the labyrinth evolution finds a way: a silicon ridge. In: **Evolvable Systems: From Biology to Hardware**. Springer Berlin Heidelberg, 1997. p. 406–422. Disponível em: <https://doi.org/10.1007/3-540-63173-9_62>. Citado na página 42.

HERRERA, F.; LOZANO, M.; VERDEGAY, J. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. **Artificial Intelligence Review**, Springer Science and Business Media LLC, v. 12, n. 4, p. 265–319, 1998. Disponível em: <<https://doi.org/10.1023/a:1006504901164>>. Citado na página 41.

HOSMER, D. W.; LEMESHOW, S. **Applied logistic regression**. Second edition. John Wiley & Sons, Inc., 2000. Disponível em: <<https://doi.org/10.1002/0471722146>>. Citado na página 52.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **ISLR: data for an introduction to statistical learning with applications in R**. [S.l.], 2017. R package version 1.2. Disponível em: <<https://CRAN.R-project.org/package=ISLR>>. Citado na página 43.

KIM, D.; YOU, S.; SO, S.; LEE, J.; YOON, S.; JANG, D. P.; KIM, I. Y.; PARK, E.; CHO, K.; CHA, W. C.; SHIN, D. W.; CHO, B. H.; PARK, H.-K. A data-driven artificial intelligence model for remote triage in the prehospital environment. **PLOS ONE**, Public Library of Science (PLoS), v. 13, n. 10, p. e0206006, oct 2018. Disponível em: <<https://doi.org/10.1371/journal.pone.0206006>>. Citado na página 68.

LIU, Y.; KHOSHGOFTAAR, T. Reducing overfitting in genetic programming models for software quality classification. In: **Proceedings of the Eighth IEEE International Conference on High Assurance Systems Engineering**. Tampa, Florida, USA: IEEE Computer Society, 2004. (HASE 2004), p. 56–65. ISBN 0769520944. Disponível em: <<https://doi.org/10.1109/hase.2004.1281730>>. Citado na página 47.

MITCHELL, M.; FORREST, S. Genetic algorithms and artificial life. **Artificial Life**, MIT Press - Journals, v. 1, n. 3, p. 267–289, abr. 1994. Disponível em: <<https://doi.org/10.1162/artl.1994.1.3.267>>. Citado nas páginas 39 e 45.

R CORE TEAM. **R: a language and environment for statistical computing**. Vienna, Austria, 2019. Disponível em: <<https://www.R-project.org/>>. Citado na página 43.

RAU, C.-S.; WU, S.-C.; CHUANG, J.-F.; HUANG, C.-Y.; LIU, H.-T.; CHIEN, P.-C.; HSIEH, C.-H. Machine learning models of survival prediction in trauma patients. **Journal of Clinical Medicine**, MDPI AG, v. 8, n. 6, p. 799, jun 2019. Disponível em: <<https://doi.org/10.3390/jcm8060799>>. Citado na página 68.

ROBIN, X.; TURCK, N.; HAINARD, A.; TIBERTI, N.; LISACEK, F.; SANCHEZ, J.-C.; MÜLLER, M. pROC: an open-source package for R and S+ to analyze and compare ROC curves. **BMC Bioinformatics**, v. 12, p. 77, 2011. Disponível em: <<https://cran.r-project.org/package=pROC>>. Citado nas páginas 43 e 47.

RSTUDIO TEAM. **RStudio: integrated development environment for R**. Boston, MA, 2019. Disponível em: <<http://www.rstudio.com/>>. Citado na página 43.

SIMÕES, E. D. V. **Development of an embedded evolutionary controller to enable collision-free navigation of a population of autonomous mobile robots**. Tese (Doutorado) — University of Kent at Canterbury, 2000. Disponível em: <<https://sites.icmc.usp.br/simoes/web/pages/thesis.html>>. Citado nas páginas 39, 40, 41 e 47.

SIMÕES, E. D. V.; BARONE, D. A. C. Predation: an approach to improving the evolution of real robots with a distributed evolutionary controller. In: **Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)**. [s.n.], 2002. v. 1, p. 664–669. Disponível em: <<https://doi.org/10.1109/robot.2002.1013434>>. Citado na página 48.

SUDHOFF, S. D. **Lecture notes on engineering analysis and design using genetic algorithms: real coded genetic algorithms**. Purdue University, 2007. Disponível em: <<https://engineering.purdue.edu/~sudhoff/ee630/>>. Citado na página 41.

TAKAHASHI, M.; KITA, H. A crossover operator using independent component analysis for real-coded genetic algorithms. In: **Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)**. [s.n.], 2001. v. 1, p. 643–649. Disponível em: <<https://ieeexplore.ieee.org/document/934452>>. Citado na página 41.

THEODORAKI, E.-M.; KOUKOUVINOS, C.; PARPOULA, C. Neural networks for prediction of trauma victims' outcome. comparison with the TRISS and Revised Trauma Score. In: **Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine**. IEEE, 2010. Disponível em: <<https://doi.org/10.1109/itab.2010.5687802>>. Citado na página 26.

THOMPSON, A. Temperature in natural and artificial systems. In: HUSBANDS, P.; HARVEY, I. (Ed.). **Proceedings of the Fourth European Conference on Artificial Life**. MIT Press, 1997. p. 388–397. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.7947>>. Citado na página 42.

TODD, P. M.; MILLER, G. F. **Biodiversity through sexual selection**. 1997. Disponível em: <http://www.unm.edu/~gfmiller/new_papers2/todd1997biodiversity.pdf>. Citado na página 40.

TOMASSINI, M. A survey of genetic algorithms. In: **Annual Reviews of Computational Physics III**. World Scientific, 1995. p. 87–118. Disponível em: <https://doi.org/10.1142/9789812830647_0003>. Citado nas páginas 39 e 40.

_____. Evolutionary algorithms. In: **Towards Evolvable Hardware**. Springer Berlin Heidelberg, 1996. p. 19–47. Disponível em: <https://doi.org/10.1007/3-540-61093-6_2>. Citado na página 40.

UMBARKAR, A. J.; SHETH, P. D. Crossover operators in genetic algorithms: a review. **ICTACT Journal on Soft Computing**, ICT Academy, v. 06, n. 01, p. 1083–1092, out. 2015. Disponível em: <<https://doi.org/10.21917/ijsc.2015.0150>>. Citado na página 41.

WATSON, R. A.; POLLACK, J. B. Recombination without respect: schema combination and disruption in genetic algorithm crossover. In: **Proceedings of the 2000 Genetic and Evolutionary Computation Conference**. Morgan Kaufmann, 2000. p. 1–55860. Disponível em: <<http://www.cs.bham.ac.uk/~wbl/biblio/gecco2000/GA200.pdf>>. Citado nas páginas 39 e 40.

WHITLEY, D. A genetic algorithm tutorial. **Statistics and Computing**, Springer Science and Business Media LLC, v. 4, n. 2, jun. 1994. Disponível em: <<https://doi.org/10.1007/bf00175354>>. Citado na página 39.

WHITLEY, D.; SUTTON, A. M. Genetic algorithms - a survey of models and methods. In: _____. **Handbook of Natural Computing**. Springer Berlin Heidelberg, 2012. v. 2-4, p. 637–671. ISBN 9783540929093. Disponível em: <https://link.springer.com/referenceworkentry/10.1007/978-3-540-92910-9_21>. Citado na página 48.

WORLD HEALTH ORGANIZATION. **Guidelines for essential trauma care**. Geneva, 2004. Disponível em: <https://www.who.int/violence_injury_prevention/publications/services/guidelines_traumacare/en/>. Citado na página 25.

_____. **Guidelines for trauma quality improvement programmes**. Geneva, 2009. Disponível em: <<https://www.who.int/emergencycare/trauma/essential-care/guidelines/en/>>. Citado nas páginas 25 e 26.

GLOSSÁRIO

Benchmark: comparação do desempenho entre dois ou mais sistemas concorrentes ou entre sistemas e um padrão de desempenho reconhecido como sendo de excelência, buscando obter referências que orientem processos de melhoria no desempenho dos sistemas avaliados.

Crossover: operador genético aplicado na reprodução de indivíduos em um algoritmo genético que combina partes dos cromossomos de dois indivíduos pais para formar o cromossomo de um novo indivíduo filho.

Fitness: medida de adaptação ao ambiente atribuída a um indivíduo em um algoritmo genético. Indica a capacidade ou desempenho de um indivíduo em solucionar um problema de otimização quando comparado a outros indivíduos, de forma análoga à capacidade de um organismo biológico em se adaptar ao ambiente em que vive.

Overfitting: produção de um modelo estatístico que se ajusta de forma muito específica a um conjunto determinado de dados e, portanto, pode falhar em ajustar dados adicionais ou prever observações futuras diversas com confiabilidade.

Trade-off: decisão que consiste na escolha de uma opção em detrimento de outra ou outras opções conflitantes ou excludentes. O *trade-off* subentende um conflito de escolha e uma consequente relação de compromisso, porque a escolha de uma coisa em relação à outra implica em não usufruir os benefícios da coisa que não é escolhida. Para que se configure o *trade-off*, deve haver conhecimento dos lados positivos e negativos das opções possíveis de escolha.

Mutação: operador genético aplicado na reprodução de indivíduos em um algoritmo genético que produz pequenas variações (ou erros intencionais) no cromossomo do indivíduo filho de forma a aumentar a diversidade genética da população de indivíduos como um todo.

CÓDIGO-FONTE

São listados neste apêndice os códigos-fonte com a implementação do algoritmo genético e com a execução dos experimentos deste trabalho. Cada arquivo está listado separadamente, em ordem alfabética de nome.

Código-fonte 1 – Algoritmo Genetico TRISS.R

```
1: algoritmo.genetico <- function(dados, populacao.inicial = NA,
  tamanho.populacao = 50, geracoes = 10,
2:                               tipo.selecao = 1, opcao.selecao
  .predacao = 0, opcao.selecao.sintese = FALSE,
3:                               tipo.crossover = 1, delta.
  mutacao.perc = c(1.0, 3.0), opcao.mutacao.genocidio = FALSE,
4:                               subset.dados.fitness = 1,
5:                               range.genes = rbind(c(0, 110),
  c(-4, 4), c(-4, 4), c(-4, 4), c(-4, 4)),
6:                               cores.computacao.paralela = 0)
  {
7:   # Inicializando
8:   populacao = NULL
9:   fitness.populacao = NULL
10:  acertos.fitness.pontos = NULL
11:  predado.substituido.cruzar = 0
12:  geracoes.variacao.minima = 0
13:  threshold.variacao.minima = 0.0001
14:  multiplicador.mutacao = 0.1
15:  stats = NULL
16:  populacoes = list()
17:  multiplicadores.mutacao = NULL
```

```
18: genocidio.ocorreu = NULL
19: predacao.ocorreu = NULL
20: sintese.ocorreu = NULL
21:
22: if (!is.na(populacao.inicial)) {
23:   populacao = populacao.inicial
24:   tamanho.populacao = NROW(populacao.inicial)
25: }
26:
27: if (tamanho.populacao %% 2 != 0) {
28:   stop("Tamanho da população deve ser par")
29: }
30:
31: # Gerando população inicial aleatória, caso não tenha sido
fornecida
32: if (is.na(populacao.inicial)) {
33:   populacao = matrix(0, tamanho.populacao, 5)
34:
35:   for (indiv in 1:tamanho.populacao) {
36:     for (gen in 1:5) {
37:       populacao[indiv, gen] = runif(1, range.genes[gen, 1],
range.genes[gen, 2])
38:     }
39:   }
40:
41:   # Debug
42:   # Inserindo indivíduo com TRISS original como melhor
43:   # populacao[1, ] = c(55, -1.2470, 0.9544, -0.0768, -1.9052)
44: }
45:
46: # Debug
47: clock.inicio.execucao = Sys.time()
48:
49: # Loop das gerações
50: for (geracao in 1:geracoes) {
51:   genocidio.na.geracao = 0
52:   predacao.na.geracao = 0
53:   sintese.na.geracao = 0
54:
55:   # Gerando a próxima população (exceto para primeira iteraçã
o, com a população inicial)
56:   if (geracao != 1) {
```

```
57:     prox.populacao = NULL
58:
59:     ## Métodos de Seleção
60:     ##
61:     ## Tipo 1: Elitismo
62:     ## Tipo 2: Torneio de 2
63:     ## Tipo 3: Roleta
64:
65:     # Crossover (cruzamento - reprodução sexuada)
66:
67:     # Mantendo melhor indivíduo sempre na nova população
68:     individuo = populacao[1, ]
69:
70:     prox.populacao = rbind(prox.populacao, individuo)
71:
72:     inicio.loop.crossover = 2
73:
74:     # Caso tenha havido predação na última geração, forçar
substituto a cruzar com o melhor
75:     if (predado.substituido.cruzar > 0) {
76:         individuo = crossover(populacao[1, ], populacao[predado
77:         .substituido.cruzar,], tipo.crossover)
78:
79:         prox.populacao = rbind(prox.populacao, individuo)
80:
81:         predado.substituido.cruzar = 0
82:
83:         inicio.loop.crossover = 3
84:     }
85:
86:     # Seguindo com o restante da população pelo tipo de seleç
ão determinado
87:
88:     if (tipo.selecao == 1) { # Elitismo
89:         for (indiv in inicio.loop.crossover:tamanho.populacao)
90:         {
91:             individuo = crossover(populacao[1, ], populacao[indiv
92:             ,], tipo.crossover)
93:
94:             prox.populacao = rbind(prox.populacao, individuo)
95:         }
```

```
94:         } else if (tipo.selecao == 2) { # Torneio de 2
95:
96:             for (i in inicio.loop.crossover:tamanho.populacao) {
97:                 candidatos = sample(1:tamanho.populacao, 4)
98:
99:                 candidatos1 = candidatos[1:2]
100:                 candidatos2 = candidatos[3:4]
101:
102:                 pai1 = candidatos1[which.max(fitness.populacao[
candidatos1])]]
103:                 pai2 = candidatos2[which.max(fitness.populacao[
candidatos2])]]
104:
105:                 individuo = crossover(populacao[pai1, ], populacao[
pai2, ], tipo.crossover)
106:
107:                 prox.populacao = rbind(prox.populacao, individuo)
108:             }
109:
110:         } else if (tipo.selecao == 3) { # Roleta
111:
112:             roleta = cumsum(fitness.populacao / sum(fitness.
populacao))
113:
114:             for (i in inicio.loop.crossover:tamanho.populacao) {
115:                 pai1 = which(roleta > runif(1))[1]
116:                 pai2 = which(roleta > runif(1))[1]
117:
118:                 individuo = crossover(populacao[pai1, ], populacao[
pai2, ], tipo.crossover)
119:
120:                 prox.populacao = rbind(prox.populacao, individuo)
121:             }
122:
123:         } else {
124:             stop("Tipo de seleção inválido")
125:         }
126:
127:         populacao = prox.populacao
128:
129:         # Mutação probabilística
130:         for (indiv in 2:tamanho.populacao) {
```

```
131:
132:     ## Mutar todos menos o melhor
133:     populacao[indiv, ] = mutation(populacao[indiv, ], delta
134:     .mutacao.perc * multiplicador.mutacao, range.genes)
135: }
136:
137:     # Selecionando apenas uma parte da população por geração
para testar fitness para evitar overfitting
138:     if (subset.dados.fitness < 1) {
139:         dados.subset = dados[sample(NROW(dados), floor(NROW(dados)
140:         ) * subset.dados.fitness)), ]
141:     } else {
142:         dados.subset = dados
143:     }
144:
145:     # Calculando o fitness da geração
146:     if (cores.computacao.paralela == 0) {
147:         # Sem usar multithreading
148:
149:         fitness.populacao = rep(0, tamanho.populacao)
150:
151:         for (indiv in 1:nrow(populacao)) {
152:             fitness.populacao[indiv] = fitness(populacao[indiv, ],
153:             dados.subset)
154:         }
155:     } else {
156:
157:         # Usando multithreading
158:         library(parallel)
159:
160:         func.fitness.parallel <- function(indiv) { fitness(
161:         populacao[indiv, ], dados.subset) }
162:
163:         if (Sys.info()[1] == "Windows") { # Usando clusterApply
164:         no Windows (único suportado)
165:             cluster <- makeCluster(cores.computacao.paralela)
166:             clusterExport(cluster, list("RTS", "ISS", "TRISS.
167:             precalc", "TRISS", "fitness", "roc", "auc"))
```

```
166:         fitness.lista = clusterApply(cluster, 1:nrow(populacao)
, func.fitness.parallel)
167:
168:         stopCluster(cluster)
169:
170:     } else { # Demais sistemas usam mclapply, pois tem melhor
performance
171:
172:         fitness.lista = mclapply(1:nrow(populacao), func.
fitness.parallel, mc.cores = cores.computacao.paralela)
173:     }
174:
175:     fitness.populacao = unlist(fitness.lista, use.names=FALSE
)
176: }
177:
178: ## Opções de Seleção
179: ##
180: ## Predação -> A cada X (10 por exemplo) gerações, pega o
pior indivíduo (no mesmo loop do Elitismo) e troca por um
aleatório
181: ## Síntese -> A cada geração, pega o pior e troca pela sí
ntese (média ou moda gen a gen da população, sendo moda para
variáveis discretas)
182:
183: # Predação
184: if ((opcao.selecao.predacao > 0) && (geracao != 1)) {
185:     if (geracao %% opcao.selecao.predacao == 0) {
186:         predacao.na.geracao = 1
187:
188:         pior.indiv = which.min(fitness.populacao)
189:
190:         for (gen in 1:5) {
191:             populacao[pior.indiv, gen] = runif(1, range.genes[gen
, 1], range.genes[gen, 2])
192:         }
193:
194:         fitness.populacao[pior.indiv] = fitness(populacao[pior.
indiv,], dados.subset)
195:
196:         predado.substituido.cruzar = pior.indiv
197:     }
```

```

198:     }
199:
200:     # Síntese
201:     if (opcao.selecao.sintese && (geracao != 1)) {
202:
203:         # Não pode ser feita síntese na mesma geração em que
houve predação
204:         if ((opcao.selecao.predacao == 0) || (geracao %% opcao.
selecao.predacao != 0)) {
205:             sintese.na.geracao = 1
206:
207:             pior.indiv = which.min(fitness.populacao)
208:
209:             for (gen in 1:5) {
210:                 populacao[pior.indiv, gen] = mean(populacao[, gen])
211:             }
212:
213:             fitness.populacao[pior.indiv] = fitness(populacao[pior.
indiv,], dados.subset)
214:         }
215:     }
216:
217:     # Ordenando
218:     ordem = order(fitness.populacao, decreasing = TRUE)
219:     populacao = populacao[ordem,]
220:     fitness.populacao = fitness.populacao[ordem]
221:
222:     if (predado.substituido.cruzar > 0) {
223:         predado.substituido.cruzar = ordem[predado.substituido.
cruzar]
224:     }
225:
226:     # Salvando estatísticas de acertos do fitness por pontos
227:     acertos.fitness.pontos[geracao] = contar.acertos.fitness.
pontos(populacao[1, ], dados)
228:
229:     # Controlando mutação variável
230:     if (geracao > 1) {
231:
232:         if (abs(fitness.populacao[1] - stats[nrow(stats), "Max"])
< threshold.variacao.minima) {
233:             geracoes.variacao.minima = geracoes.variacao.minima + 1

```

```
234:
235:     if (geracoes.variacao.minima > 5) {
236:         multiplicador.mutacao = multiplicador.mutacao * 2
237:
238:         if (multiplicador.mutacao > 32) {
239:
240:             ## Caso a mutação variável não melhore a população,
fazer genocídio
241:             ## gerando uma nova população aleatória, mas
mantendo o melhor indivíduo
242:
243:             genocidio.na.geracao = 1
244:
245:             for (indiv in 2:tamanho.populacao) {
246:                 for (gen in 1:5) {
247:                     populacao[indiv, gen] = runif(1, range.genes[
gen, 1], range.genes[gen, 2])
248:                 }
249:
250:                 fitness.populacao[indiv] = fitness(populacao[
indiv,], dados.subset)
251:             }
252:
253:             # Reordenando
254:             ordem = order(fitness.populacao, decreasing = TRUE)
255:             populacao = populacao[ordem,]
256:             fitness.populacao = fitness.populacao[ordem]
257:
258:             # Debug
259:             print("Genocídio!")
260:
261:             multiplicador.mutacao = 0.1
262:         }
263:
264:         geracoes.variacao.minima = 0
265:     }
266: } else {
267:     multiplicador.mutacao = 0.1
268:     geracoes.variacao.minima = 0
269: }
270: }
271:
```

```

272:     # Salvando histórico do multiplicador de mutação e se houve
        genocídio, predação ou síntese
273:     multiplicadores.mutacao = rbind(multiplicadores.mutacao,
multiplicador.mutacao)
274:     genocidio.ocorreu = rbind(genocidio.ocorreu, genocidio.na.
geracao)
275:     predacao.ocorreu = rbind(predacao.ocorreu, predacao.na.
geracao)
276:     sintese.ocorreu = rbind(sintese.ocorreu, sintese.na.geracao
)
277:
278:     # Calculando estatísticas
279:     stats = rbind(stats, c(Geracao = geracao, Mean = mean(
fitness.populacao), SD = sd(fitness.populacao), Min = min(
fitness.populacao), Max = max(fitness.populacao)))
280:
281:     # Armazenando populacao
282:     populacoes[[geracao]] = populacao
283:
284:     # Debug
285:     clock.final.execucao = Sys.time()
286:     print(paste("Concluída geração: ", geracao, " | Seleção: ",
tipo.selecao, " | Crossover: ", tipo.crossover, " | ", Sys.
time(), " | Duração: ", difftime(clock.final.execucao, clock
.inicio.execucao, units = "s"), " s", sep = ""))
287:     clock.inicio.execucao = clock.final.execucao
288:     # Sys.sleep(1) # Pausa para processar eventos de interface
em máquinas menos potentes
289: }
290:
291: # Construindo retorno
292:
293: ret = list()
294:
295: ret$melhor.individuo = populacao[1, ]
296: ret$populacao.final = populacao
297: ret$fitness.final = fitness.populacao
298: ret$stats = stats
299: ret$multiplicadores.mutacao = multiplicadores.mutacao
300: ret$genocidio.ocorreu = genocidio.ocorreu
301: ret$predacao.ocorreu = predacao.ocorreu
302: ret$sintese.ocorreu = sintese.ocorreu

```

```

303:   ret$populacoes = populacoes
304:
305:   ret$parametos = list()
306:
307:   ret$parametos$populacao.inicial = populacao.inicial
308:   ret$parametos$tamanho.populacao = tamanho.populacao
309:   ret$parametos$geracoes = geracoes
310:   ret$parametos$tipo.selecao = tipo.selecao
311:   ret$parametos$opcao.selecao.predacao = opcao.selecao.predacao
312:   ret$parametos$opcao.selecao.sintese = opcao.selecao.sintese
313:   ret$parametos$tipo.crossover = tipo.crossover
314:   ret$parametos$delta.mutacao.perc = delta.mutacao.perc
315:   ret$parametos$range.genes = range.genes
316:   ret$parametos$cores.computacao.paralela = cores.computacao.
      paralela
317:   # ret$parametos$dados = dados
318:
319:   ret$acertos.fitness.pontos = acertos.fitness.pontos
320:
321:   return (ret)
322: }

```

Código-fonte 2 – Executar.R

```

1: executar <- function(dados = NULL, arquivo = "", label = "") {
2:   library(parallel)
3:   library(ISLR)
4:
5:   source("Scores.R", encoding = 'UTF-8')
6:   source("Fitness.R", encoding = 'UTF-8')
7:   source("Operators.R", encoding = 'UTF-8')
8:   source("Gerar Dados.R", encoding = 'UTF-8')
9:   source("Regressao Logistica.R", encoding = 'UTF-8')
10:  source("Plot.R", encoding = 'UTF-8')
11:  source("Algoritmo Genetico TRISS.R", encoding = 'UTF-8')
12:
13:  path = paste("Resultados/Resultados (", label,") ", format(
      Sys.time(), "%Y-%m-%d %H-%M-%S"), sep = "")
14:  dir.create(path)
15:
16:  if (is.null(dados)) {
17:    dados = read.csv(arquivo)

```

```
18:   }
19:
20:   k.delta.mutacao = 1
21:
22:   tamanho.populacao = 20
23:   geracoes = 100
24:   opcao.selecao.predacao = 5
25:   opcao.selecao.sintese = TRUE
26:   delta.mutacao.perc = c(1.0, 30.0) * k.delta.mutacao
27:   opcao.mutacao.genocidio = TRUE
28:
29:   subset.dados.fitness = 1
30:
31:   cores = detectCores()
32:   results = c()
33:
34:   for (tipo.selecao in 1:3) {
35:     for (tipo.crossover in 1:3) {
36:       result = algoritmo.genetico(dados, NA, tamanho.populacao
= tamanho.populacao, geracoes = geracoes, tipo.selecao =
tipo.selecao, tipo.crossover = tipo.crossover, opcao.selecao
.predacao = opcao.selecao.predacao, opcao.selecao.sintese =
opcao.selecao.sintese, delta.mutacao.perc = delta.mutacao.
perc, opcao.mutacao.genocidio = opcao.mutacao.genocidio,
subset.dados.fitness = subset.dados.fitness, cores.
computacao.paralela = cores)
37:
38:       results = rbind(results, c(tipo.selecao = tipo.selecao,
tipo.crossover = tipo.crossover, result))
39:
40:       saveRDS(result, paste(path, "/Result Sel ", tipo.selecao,
" Cross ", tipo.crossover, ".rds", sep = ""))
41:       plotar.stats.png(result$stats, paste("Evolução do Fitness
Médio - Seleção por ", switch(tipo.selecao, "Elitismo", "
Torneio2", "Roulette"), " e Crossover ", switch(tipo.
crossover, "Uniform", "Average", "BLX"), sep = ""), paste(
path, "/Plot Sel ", switch(tipo.selecao, "Elitismo", "
Torneio2", "Roulette"), " Cross ", switch(tipo.crossover, "
Uniform", "Average", "BLX"), ".png", sep = ""))
42:     }
43:   }
44:
```

```

45:   saveRDS(results, paste(path, "/Results.rds", sep = ""))
46:   plotar.stats.max.png(results, paste(path, "/Plot Max.png",
    sep = ""), c(0, 1))
47:
48:   return (results)
49: }

```

Código-fonte 3 – Experimento.R

```

1: library(pROC)
2: source("Executar.R", encoding = 'UTF-8')
3:
4: plotar.arquivo = TRUE
5:
6: BD = read.csv("Dados/BD.csv")
7:
8: # rand.num = runif(nrow(BD))
9:
10: erro.BD.A = mean(((BD["sobreviveu.BD.A"]-BD["triss"]) ^ 2)[,
    1])
11: erro.BD.B10 = mean(((BD["sobreviveu.BD.B10"]-BD["triss"]) ^ 2)
    [, 1])
12: erro.BD.B20 = mean(((BD["sobreviveu.BD.B20"]-BD["triss"]) ^ 2)
    [, 1])
13: erro.BD.B30 = mean(((BD["sobreviveu.BD.B30"]-BD["triss"]) ^ 2)
    [, 1])
14: erro.BD.B50 = mean(((BD["sobreviveu.BD.B50"]-BD["triss"]) ^ 2)
    [, 1])
15:
16: rocobj.BD.A <- roc(BD$sobreviveu.BD.A, BD$triss, direction = "<
    ", levels = c(0, 1))
17: rocobj.BD.B10 <- roc(BD$sobreviveu.BD.B10, BD$triss, direction
    = "<", levels = c(0, 1))
18: rocobj.BD.B20 <- roc(BD$sobreviveu.BD.B20, BD$triss, direction
    = "<", levels = c(0, 1))
19: rocobj.BD.B30 <- roc(BD$sobreviveu.BD.B30, BD$triss, direction
    = "<", levels = c(0, 1))
20: rocobj.BD.B50 <- roc(BD$sobreviveu.BD.B50, BD$triss, direction
    = "<", levels = c(0, 1))
21:
22: auc.BD.A = auc(rocobj.BD.A)[1]
23: auc.BD.B10 = auc(rocobj.BD.B10)[1]

```

```

24: auc.BD.B20 = auc(rocobj.BD.B20)[1]
25: auc.BD.B30 = auc(rocobj.BD.B30)[1]
26: auc.BD.B50 = auc(rocobj.BD.B50)[1]
27:
28: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-BDs
    .png", 600, 600, "px", pointsize = 14) }
29: plot.roc(rocobj.BD.A, col = "black", lwd = 0.5, print.auc =
    TRUE,
30:         print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
    xlab = "Especificidade", ylab = "Sensibilidade") # main = "
    Curvas ROC dos Bancos de Dados de Pacientes"
31: plot.roc(rocobj.BD.B10, col = "blue", lwd = 0.5, print.auc =
    TRUE,
32:         print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
33: plot.roc(rocobj.BD.B20, col = "darkgreen", lwd = 0.5, print.auc
    = TRUE,
34:         print.auc.x = 0.57, print.auc.y = 0.57, add = TRUE)
35: plot.roc(rocobj.BD.B30, col = "orange", lwd = 0.5, print.auc =
    TRUE,
36:         print.auc.x = 0.53, print.auc.y = 0.53, add = TRUE)
37: plot.roc(rocobj.BD.B50, col = "red", lwd = 0.5, print.auc =
    TRUE,
38:         print.auc.x = 0.49, print.auc.y = 0.49, add = TRUE)
39: legend(0, 0,
40:       legend = c("BD-A", "BD-B10", "BD-B20", "BD-B30", "BD-B50
    "),
41:       col = c("black", "blue", "darkgreen", "orange", "red"),
42:       lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
43: if (plotar.arquivo) { dev.off() }
44:
45: results.BD.A = executar(arquivo = "Dados/BD-A.csv", label = "BD
    -A")
46: results.BD.B10 = executar(arquivo = "Dados/BD-B10.csv", label =
    "BD-B10")
47: results.BD.B20 = executar(arquivo = "Dados/BD-B20.csv", label =
    "BD-B20")
48: results.BD.B30 = executar(arquivo = "Dados/BD-B30.csv", label =
    "BD-B30")
49: results.BD.B50 = executar(arquivo = "Dados/BD-B50.csv", label =
    "BD-B50")
50:
51: melhor.combinacao = 3

```

```

52:
53: melhor.individuo.BD.A = results.BD.A[melhor.combinacao, ]$
    melhor.individuo
54: melhor.individuo.BD.B10 = results.BD.B10[melhor.combinacao, ]$
    melhor.individuo
55: melhor.individuo.BD.B20 = results.BD.B20[melhor.combinacao, ]$
    melhor.individuo
56: melhor.individuo.BD.B30 = results.BD.B30[melhor.combinacao, ]$
    melhor.individuo
57: melhor.individuo.BD.B50 = results.BD.B50[melhor.combinacao, ]$
    melhor.individuo
58:
59: triss.ag.BD.A = matrix(nrow = nrow(BD), ncol = 1)
60: triss.ag.BD.B10 = matrix(nrow = nrow(BD), ncol = 1)
61: triss.ag.BD.B20 = matrix(nrow = nrow(BD), ncol = 1)
62: triss.ag.BD.B30 = matrix(nrow = nrow(BD), ncol = 1)
63: triss.ag.BD.B50 = matrix(nrow = nrow(BD), ncol = 1)
64:
65: for (i in 1:nrow(BD)) {
66:   triss.ag.BD.A[i] = TRISS(BD$fr[i], BD$pas[i], BD$glasgow_
    resultado[i], BD$iss_resultado[i], BD$idade[i],
67:                           TRUE,
68:                           melhor.individuo.BD.A[1],
69:                           rbind(melhor.individuo.BD.A[2:5],
    melhor.individuo.BD.A[2:5]))
70:
71:   triss.ag.BD.B10[i] = TRISS(BD$fr[i], BD$pas[i], BD$glasgow_
    resultado[i], BD$iss_resultado[i], BD$idade[i],
72:                             TRUE,
73:                             melhor.individuo.BD.B10[1],
74:                             rbind(melhor.individuo.BD.B10
    [2:5], melhor.individuo.BD.B10[2:5]))
75:
76:   triss.ag.BD.B20[i] = TRISS(BD$fr[i], BD$pas[i], BD$glasgow_
    resultado[i], BD$iss_resultado[i], BD$idade[i],
77:                             TRUE,
78:                             melhor.individuo.BD.B20[1],
79:                             rbind(melhor.individuo.BD.B20
    [2:5], melhor.individuo.BD.B20[2:5]))
80:
81:   triss.ag.BD.B30[i] = TRISS(BD$fr[i], BD$pas[i], BD$glasgow_
    resultado[i], BD$iss_resultado[i], BD$idade[i],

```

```

82:                                     TRUE,
83:                                     melhor.individuo.BD.B30[1],
84:                                     rbind(melhor.individuo.BD.B30
      [2:5], melhor.individuo.BD.B30[2:5]))
85:
86:   triss.ag.BD.B50[i] = TRISS(BD$fr[i], BD$pas[i], BD$glasgow_
      resultado[i], BD$iss_resultado[i], BD$idade[i],
87:                               TRUE,
88:                               melhor.individuo.BD.B50[1],
89:                               rbind(melhor.individuo.BD.B50
      [2:5], melhor.individuo.BD.B50[2:5]))
90: }
91:
92: rocobj.ag.BD.A <- roc(BD$sobreviveu.BD.A, triss.ag.BD.A[, 1],
      direction = "<", levels = c(0, 1))
93: rocobj.ag.BD.B10 <- roc(BD$sobreviveu.BD.B10, triss.ag.BD.B10[,
      1], direction = "<", levels = c(0, 1))
94: rocobj.ag.BD.B20 <- roc(BD$sobreviveu.BD.B20, triss.ag.BD.B20[,
      1], direction = "<", levels = c(0, 1))
95: rocobj.ag.BD.B30 <- roc(BD$sobreviveu.BD.B30, triss.ag.BD.B30[,
      1], direction = "<", levels = c(0, 1))
96: rocobj.ag.BD.B50 <- roc(BD$sobreviveu.BD.B50, triss.ag.BD.B50[,
      1], direction = "<", levels = c(0, 1))
97:
98: auc.ag.BD.A = auc(rocobj.ag.BD.A)[1]
99: auc.ag.BD.B10 = auc(rocobj.ag.BD.B10)[1]
100: auc.ag.BD.B20 = auc(rocobj.ag.BD.B20)[1]
101: auc.ag.BD.B30 = auc(rocobj.ag.BD.B30)[1]
102: auc.ag.BD.B50 = auc(rocobj.ag.BD.B50)[1]
103:
104: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-AG-
      BD-A.png", 600, 600, "px", pointsize = 14) }
105: plot.roc(rocobj.BD.A, col = "black", lwd = 0.5, print.auc =
      TRUE,
106:           print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
      xlab = "Especificidade", ylab = "Sensibilidade") # main = ""
107: plot.roc(rocobj.ag.BD.A, col = "blue", lwd = 0.5, print.auc =
      TRUE,
108:           print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
109: legend(0, 0,
110:        legend = c("TRISS", "TRISS-AG"),
111:        col = c("black", "blue"),

```

```
112:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
113: if (plotar.arquivo) { dev.off() }
114:
115: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-AG-
    BD-B10.png", 600, 600, "px", pointsize = 14) }
116: plot.roc(rocobj.BD.B10, col = "black", lwd = 0.5, print.auc =
    TRUE,
117:         print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
            xlab = "Especificidade", ylab = "Sensibilidade") # main = ""
118: plot.roc(rocobj.ag.BD.B10, col = "blue", lwd = 0.5, print.auc =
    TRUE,
119:         print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
120: legend(0, 0,
121:       legend = c("TRISS", "TRISS-AG"),
122:       col = c("black", "blue"),
123:       lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
124: if (plotar.arquivo) { dev.off() }
125:
126: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-AG-
    BD-B20.png", 600, 600, "px", pointsize = 14) }
127: plot.roc(rocobj.BD.B20, col = "black", lwd = 0.5, print.auc =
    TRUE,
128:         print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
            xlab = "Especificidade", ylab = "Sensibilidade") # main = ""
129: plot.roc(rocobj.ag.BD.B20, col = "blue", lwd = 0.5, print.auc =
    TRUE,
130:         print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
131: legend(0, 0,
132:       legend = c("TRISS", "TRISS-AG"),
133:       col = c("black", "blue"),
134:       lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
135: if (plotar.arquivo) { dev.off() }
136:
137: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-AG-
    BD-B30.png", 600, 600, "px", pointsize = 14) }
138: plot.roc(rocobj.BD.B30, col = "black", lwd = 0.5, print.auc =
    TRUE,
139:         print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
            xlab = "Especificidade", ylab = "Sensibilidade") # main = ""
140: plot.roc(rocobj.ag.BD.B30, col = "blue", lwd = 0.5, print.auc =
    TRUE,
141:         print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
```

```

142: legend(0, 0,
143:         legend = c("TRISS", "TRISS-AG"),
144:         col = c("black", "blue"),
145:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
146: if (plotar.arquivo) { dev.off() }
147:
148: if (plotar.arquivo) { png(filename = "Experimento/ROC-TRISS-AG-
      BD-B50.png", 600, 600, "px", pointsize = 14) }
149: plot.roc(rocobj.BD.B50, col = "black", lwd = 0.5, print.auc =
      TRUE,
150:         print.auc.x = 0.65, print.auc.y = 0.65, grid = 0.1,
      xlab = "Especificidade", ylab = "Sensibilidade") # main = ""
151: plot.roc(rocobj.ag.BD.B50, col = "blue", lwd = 0.5, print.auc =
      TRUE,
152:         print.auc.x = 0.61, print.auc.y = 0.61, add = TRUE)
153: legend(0, 0,
154:         legend = c("TRISS", "TRISS-AG"),
155:         col = c("black", "blue"),
156:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
157: if (plotar.arquivo) { dev.off() }
158:
159: if (plotar.arquivo) { png(filename = "Experimento/Acertos-TRISS
      -AG-BD-A.png", 900, 600, "px", pointsize = 14) }
160: qtd.geracoes = NROW(results.BD.A[melhor.combinacao, ]$acertos.
      fitness.pontos)
161: acertos.triss = contar.acertos.fitness.pontos(c(55, -1.2470,
      0.9544, -0.0768, -1.9052), read.csv("Dados/BD-A.csv"))
162: plot(1:qtd.geracoes, rep(acertos.triss, qtd.geracoes),
163:      col = "red", type = "l", lwd = 0.5, ylim = c(nrow(BD) -
      5000, nrow(BD)),
164:      xlab = "Geração", ylab = "Acertos")
165: lines(1:qtd.geracoes, results.BD.A[melhor.combinacao, ]$acertos
      .fitness.pontos,
166:      col = "blue", type = "l", lwd = 0.5)
167: legend(qtd.geracoes, nrow(BD) - 5000,
168:         legend = c("TRISS", "TRISS-AG"),
169:         col = c("red", "blue"),
170:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
171: if (plotar.arquivo) { dev.off() }
172:
173: if (plotar.arquivo) { png(filename = "Experimento/Acertos-TRISS
      -AG-BD-B10.png", 900, 600, "px", pointsize = 14) }

```

```
174: qtd.geracoes = NROW(results.BD.B10[melhor.combinacao, ]$acertos
    .fitness.pontos)
175: acertos.triss = contar.acertos.fitness.pontos(c(55, -1.2470,
    0.9544, -0.0768, -1.9052), read.csv("Dados/BD-B10.csv"))
176: plot(1:qtd.geracoes, rep(acertos.triss, qtd.geracoes),
177:      col = "red", type = "l", lwd = 0.5, ylim = c(nrow(BD) -
    5000, nrow(BD)),
178:      xlab = "Geração", ylab = "Acertos")
179: lines(1:qtd.geracoes, results.BD.B10[melhor.combinacao, ]$
    acertos.fitness.pontos,
180:      col = "blue", type = "l", lwd = 0.5)
181: legend(qtd.geracoes, nrow(BD) - 5000,
182:      legend = c("TRISS", "TRISS-AG"),
183:      col = c("red", "blue"),
184:      lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
185: if (plotar.arquivo) { dev.off() }
186:
187: if (plotar.arquivo) { png(filename = "Experimento/Acertos-TRISS
    -AG-BD-B20.png", 900, 600, "px", pointsize = 14) }
188: qtd.geracoes = NROW(results.BD.B20[melhor.combinacao, ]$acertos
    .fitness.pontos)
189: acertos.triss = contar.acertos.fitness.pontos(c(55, -1.2470,
    0.9544, -0.0768, -1.9052), read.csv("Dados/BD-B20.csv"))
190: plot(1:qtd.geracoes, rep(acertos.triss, qtd.geracoes),
191:      col = "red", type = "l", lwd = 0.5, ylim = c(nrow(BD) -
    5000, nrow(BD)),
192:      xlab = "Geração", ylab = "Acertos")
193: lines(1:qtd.geracoes, results.BD.B20[melhor.combinacao, ]$
    acertos.fitness.pontos,
194:      col = "blue", type = "l", lwd = 0.5)
195: legend(qtd.geracoes, nrow(BD) - 5000,
196:      legend = c("TRISS", "TRISS-AG"),
197:      col = c("red", "blue"),
198:      lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
199: if (plotar.arquivo) { dev.off() }
200:
201: if (plotar.arquivo) { png(filename = "Experimento/Acertos-TRISS
    -AG-BD-B30.png", 900, 600, "px", pointsize = 14) }
202: qtd.geracoes = NROW(results.BD.B30[melhor.combinacao, ]$acertos
    .fitness.pontos)
203: acertos.triss = contar.acertos.fitness.pontos(c(55, -1.2470,
    0.9544, -0.0768, -1.9052), read.csv("Dados/BD-B30.csv"))
```

```

204: plot(1:qtd.geracoes, rep(acertos.triss, qtd.geracoes),
205:       col = "red", type = "l", lwd = 0.5, ylim = c(nrow(BD) -
              5000, nrow(BD))),
206:       xlab = "Geração", ylab = "Acertos")
207: lines(1:qtd.geracoes, results.BD.B30[melhor.combinacao, ]$
              acertos.fitness.pontos,
208:       col = "blue", type = "l", lwd = 0.5)
209: legend(qtd.geracoes, nrow(BD) - 5000,
210:       legend = c("TRISS", "TRISS-AG"),
211:       col = c("red", "blue"),
212:       lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
213: if (plotar.arquivo) { dev.off() }
214:
215: if (plotar.arquivo) { png(filename = "Experimento/Acertos-TRISS
              -AG-BD-B50.png", 900, 600, "px", pointsize = 14) }
216: qtd.geracoes = NROW(results.BD.B50[melhor.combinacao, ]$acertos
              .fitness.pontos)
217: acertos.triss = contar.acertos.fitness.pontos(c(55, -1.2470,
              0.9544, -0.0768, -1.9052), read.csv("Dados/BD-B50.csv"))
218: plot(1:qtd.geracoes, rep(acertos.triss, qtd.geracoes),
219:       col = "red", type = "l", lwd = 0.5, ylim = c(nrow(BD) -
              5000, nrow(BD))),
220:       xlab = "Geração", ylab = "Acertos")
221: lines(1:qtd.geracoes, results.BD.B50[melhor.combinacao, ]$
              acertos.fitness.pontos,
222:       col = "blue", type = "l", lwd = 0.5)
223: legend(qtd.geracoes, nrow(BD) - 5000,
224:       legend = c("TRISS", "TRISS-AG"),
225:       col = c("red", "blue"),
226:       lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
227: if (plotar.arquivo) { dev.off() }

```

Código-fonte 4 – Fitness.R

```

1: fitness <- function(individuo, dados) {
2:   # Calcula fitness pela quantidade de acertos usando melhor
      threshold
3:   triss = vector(length = nrow(dados))
4:
5:   for (i in 1:nrow(dados)) {
6:     triss[i] = TRISS(dados$fr[i], dados$pas[i], dados$glasgow_
              resultado[i], dados$iss_resultado[i], dados$idade[i],

```

```
7:             TRUE,
8:             individuo[1],
9:             rbind(individuo[2:5], individuo[2:5]))
10:   }
11:
12:   rocobj <- roc(dados$sobreviveu, triss, direction = "<",
13:               levels = c(0, 1))
14:
15:   x = coords(rocobj, "best", ret = "threshold", transpose =
16:             FALSE)[1, 1]
17:
18:   acertos = sum(((triss >= x) & (dados$sobreviveu == "1")) | ((
19:               triss < x) & (dados$sobreviveu == "0"))))
20:
21:   return (acertos / nrow(dados))
22: }
23:
24: contar.acertos.fitness.pontos <- function(individuo, dados) {
25:   # Conta acertos do fitness usando melhor threshold
26:   triss = vector(length = nrow(dados))
27:
28:   for (i in 1:nrow(dados)) {
29:     triss[i] = TRISS(dados$fr[i], dados$pas[i], dados$glasgow_
30:                     resultado[i], dados$iss_resultado[i], dados$idade[i],
31:                     TRUE,
32:                     individuo[1],
33:                     rbind(individuo[2:5], individuo[2:5]))
34:   }
35:
36:   rocobj <- roc(dados$sobreviveu, triss, direction = "<",
37:               levels = c(0, 1))
38:
39:   x = coords(rocobj, "best", ret = "threshold", transpose =
40:             FALSE)[1, 1]
41:
42:   acertos = sum(((triss >= x) & (dados$sobreviveu == "1")) | ((
43:               triss < x) & (dados$sobreviveu == "0"))))
44:
45:   return (acertos)
46: }
```

Código-fonte 5 – Gerar Dados.R

```

1: gerar.dados <- function(observacoes = 20000) {
2:   dados.gerados = matrix(nrow = observacoes, ncol = 11)
3:   colnames(dados.gerados) = c("fr", "pas", "glasgow_resultado",
      "iss_resultado", "idade", "triss", "sobreviveu.BD.A", "
      sobreviveu.BD.B10", "sobreviveu.BD.B20", "sobreviveu.BD.B30"
      , "sobreviveu.BD.B50")
4:
5:   for (i in 1:observacoes) {
6:     dados.gerados[i, 1] = sample(0:39, 1, TRUE) # fr
7:     dados.gerados[i, 2] = sample(0:180, 1, TRUE) # pas
8:     dados.gerados[i, 3] = sample(3:15, 1, TRUE) # glasgow_
      resultado
9:     dados.gerados[i, 4] = ISS(sample(0:6, 3, TRUE)) # iss_
      resultado
10:    dados.gerados[i, 5] = sample(0:90, 1, TRUE) # idade
11:
12:    triss = TRISS(dados.gerados[i, 1], dados.gerados[i, 2],
      dados.gerados[i, 3], dados.gerados[i, 4], dados.gerados[i,
      5])
13:
14:    dados.gerados[i, 6] = triss
15:
16:    for (j in 1:5) {
17:      trials = vector(length = 10)
18:      mult = switch(j, 1, 1 - 0.10, 1 - 0.20, 1 - 0.30, 1 -
      0.50)
19:
20:      if ((j > 1) && (dados.gerados[i, j + 6 - 1] == 0)) {
21:
22:        dados.gerados[i, j + 6] = 0
23:
24:      } else {
25:
26:        for (k in 1:length(trials)) {
27:          x = runif(1)
28:
29:          trials[k] = ifelse(x <= (triss * mult), 1, 0)
30:        }
31:
32:        dados.gerados[i, j + 6] = ifelse(sum(trials) > (length(

```

```

        trials) / 2), 1, 0)
33:     }
34:   }
35: }
36:
37: for (i in 7:11) {
38:   dados.salvar = dados.gerados[, c(1:5, i)]
39:   colnames(dados.salvar)[6] = "sobreviveu"
40:   write.csv(dados.salvar, switch(i - 6, "Dados/BD-A.csv", "
    Dados/BD-B10.csv", "Dados/BD-B20.csv", "Dados/BD-B30.csv", "
    Dados/BD-B50.csv"), row.names = FALSE)
41: }
42:
43: write.csv(dados.gerados, "Dados/BD.csv", row.names = FALSE)
44:
45: return(as.data.frame(dados.gerados))
46: }

```

Código-fonte 6 – Operators.R

```

1: mutation <- function(individuo, delta.mutacao.perc = c(1.0,
    3.0), range.genes = rbind(c(0, 110), c(-4, 4), c(-4, 4), c
    (-4, 4), c(-4, 4))) {
2:   # Alterado por performance apos profiling
3:   #
4:   # prob.mutacao = rbind(c(0.95, 5),
5:   #
6:   #
7:   #
8:   #
9:   #
10:  # nivel.mutacao = runif(1)
11:  # n.gens.mutar = prob.mutacao[nivel.mutacao >= prob.mutacao[,
    1], , drop = FALSE][1, 2]
12:
13:  # Sortear quantos genes serão mutados
14:  nivel.mutacao = runif(1)
15:
16:  n.gens.mutar = 0
17:
18:  if (nivel.mutacao < 0.5) {
19:    n.gens.mutar = 1

```

```

20: } else if (nivel.mutacao < 0.7) {
21:   n.gens.mutar = 2
22: } else if (nivel.mutacao < 0.85) {
23:   n.gens.mutar = 3
24: } else if (nivel.mutacao < 0.95) {
25:   n.gens.mutar = 4
26: } else {
27:   n.gens.mutar = 5
28: }
29:
30: gens.mutar = sample(1:length(individuo), n.gens.mutar)
31:
32: for (gen in gens.mutar) {
33:   # Aumentar ou diminuir cada gen sorteado por uma
porcentagem sorteada (default entre 1% e 3%) do seu range de
valores
34:   individuo[gen] = individuo[gen] + sample(c(-1, 1), 1) *
     runif(1, delta.mutacao.perc[1], delta.mutacao.perc[2]) / 100
     * (range.genes[gen, 2] - range.genes[gen, 1])
35: }
36:
37: return (individuo)
38: }
39:
40: crossover <- function(pai1, pai2, tipo = 1) {
41:   pais = rbind(pai1, pai2)
42:   filho = rep(0, length(pai1))
43:
44:   if (tipo == 1) { # Uniform: Percorrer gen a gen e sortear
cara ou coroa de qual pai o filho herda
45:
46:     for (gen in 1:length(filho)) {
47:       filho[gen] = pais[sample(c(1, 2), 1), gen]
48:     }
49:
50:   } else if (tipo == 2) { # Average: Calcular gen a gen a média
dos pais (atenção para genes que permitem)
51:
52:     filho = (pai1 + pai2) / 2
53:
54:   } else if (tipo == 3) { # BLX: Preferido Delbem - Sortear
para cada gen um número entre 1 e 5 e escolher: pai - diff /

```

```

2 ; pai ; pai + diff / 2; mae; mae + diff / 2
55:
56:   for (gen in 1:length(filho)) {
57:     dif = abs((pai2[gen] - pai1[gen]) / 2)
58:
59:     filho[gen] = switch(sample(1:5, 1), pai1[gen] - dif, pai1
[gen], pai1[gen] + dif, pai2[gen], pai2[gen] + dif)
60:   }
61:
62: } else {
63:   stop("Tipo de crossover inválido")
64: }
65:
66: return (filho)
67: }

```

Código-fonte 7 – Plot.R

```

1: plotar.stats <- function(stats, titulo, limites.y = c(0, 1)) {
2:   plot(stats[, "Geracao"], stats[, "Mean"], type = "l", lwd = 1,
      col = "blue",
3:     main = titulo, xlab = "Geração", ylab = "Fitness Médio
com Desvio Padrão",
4:     xlim = range(stats[, "Geracao"]), ylim = limites.y,
5:     panel.first = grid(col = "lightgray"))
6:
7:   polygon(c(stats[, "Geracao"], rev(stats[, "Geracao"])), c(stats
[, "Mean"] + stats[, "SD"], rev(stats[, "Mean"] - stats[, "SD"])
),
8:     col = rgb(1, 0, 0, alpha = 0.1, maxColorValue = 1),
border = NA)
9:
10:  lines(stats[, "Geracao"], stats[, "Mean"] + stats[, "SD"], "l",
lwd = 0.5, col = "red")
11:  lines(stats[, "Geracao"], stats[, "Mean"] - stats[, "SD"], "l",
lwd = 0.5, col = "red")
12:  lines(stats[, "Geracao"], stats[, "Max"], "l", lwd = 0.5, col =
"darkgoldenrod")
13:  lines(stats[, "Geracao"], stats[, "Min"], "l", lwd = 0.5, col =
"darkgoldenrod")
14:
15:  lines(stats[, "Geracao"], stats[, "Mean"], "l", lwd = 1, col =

```

```

        "blue") # Repetindo plot da média
16:
17:   legend(max(stats[, "Geracao"]), 0,
18:         legend = c(
19:           "Média",
20:           "Desvio Padrão (+ / -)",
21:           "Máximo / Mínimo"
22:         ),
23:         col = c(
24:           "blue",
25:           "red",
26:           "darkgoldenrod"
27:         ),
28:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)
29: }
30:
31: plotar.stats.png <- function(stats, titulo, nome.arquivo) {
32:   png(filename = nome.arquivo, 900, 600, "px", pointsize = 14)
33:   plotar.stats(stats, titulo)
34:   dev.off()
35: }
36:
37: plotar.stats.max <- function(resultados, limites.y = c(0, 1))
38: {
39:   comparar.max = list()
40:   comparar.max$ElitUnif = resultados[[1,6]][,5]
41:   comparar.max$ElitAver = resultados[[2,6]][,5]
42:   comparar.max$ElitBLX = resultados[[2,6]][,5]
43:   comparar.max$ElitBLX = resultados[[3,6]][,5]
44:   comparar.max$Tor2Unif = resultados[[4,6]][,5]
45:   comparar.max$Tor2Aver = resultados[[5,6]][,5]
46:   comparar.max$Tor2BLX = resultados[[6,6]][,5]
47:   comparar.max$RoulUnif = resultados[[7,6]][,5]
48:   comparar.max$RoulAver = resultados[[8,6]][,5]
49:   comparar.max$RoulBLX = resultados[[9,6]][,5]
50:
51:   plot(1:NROW(comparar.max[[1]]), comparar.max[[1]], type = "l"
        , lwd = 1, col = "blue",
52:       main = "Comparativo de Fitness Máximo entre Algoritmos",
        xlab = "Geração", ylab = "Fitness Máximo",
53:       xlim = c(1, NROW(comparar.max[[1]])), ylim = limites.y
        ,

```

```

54:     panel.first = grid(col = "lightgray"))
55:   lines(1:NROW(comparar.max[[2]]), comparar.max[[2]], "l", lwd
      = 1, col = "red")
56:   lines(1:NROW(comparar.max[[3]]), comparar.max[[3]], "l", lwd
      = 1, col = "green")
57:   lines(1:NROW(comparar.max[[4]]), comparar.max[[4]], "l", lwd
      = 1, col = "yellow")
58:   lines(1:NROW(comparar.max[[5]]), comparar.max[[5]], "l", lwd
      = 1, col = "cyan")
59:   lines(1:NROW(comparar.max[[6]]), comparar.max[[6]], "l", lwd
      = 1, col = "magenta")
60:   lines(1:NROW(comparar.max[[7]]), comparar.max[[7]], "l", lwd
      = 1, col = "pink")
61:   lines(1:NROW(comparar.max[[8]]), comparar.max[[8]], "l", lwd
      = 1, col = "darkgrey")
62:   lines(1:NROW(comparar.max[[9]]), comparar.max[[9]], "l", lwd
      = 1, col = "gold")
63:
64:   legend(NROW(comparar.max[[1]]), limites.y[1],
65:         legend = c(
66:           "Seleção por Elitismo e Crossover Uniform",
67:           "Seleção por Elitismo e Crossover Average",
68:           "Seleção por Elitismo e Crossover BLX",
69:           "Seleção por Torneio2 e Crossover Uniform",
70:           "Seleção por Torneio2 e Crossover Average",
71:           "Seleção por Torneio2 e Crossover BLX",
72:           "Seleção por Roulette e Crossover Uniform",
73:           "Seleção por Roulette e Crossover Average",
74:           "Seleção por Roulette e Crossover BLX"
75:         ),
76:         col = c(
77:           "blue",
78:           "red",
79:           "green",
80:           "yellow",
81:           "cyan",
82:           "magenta",
83:           "pink",
84:           "darkgrey",
85:           "gold"
86:         ),
87:         lty = c(1, 1), cex=0.8, xjust = 1, yjust = 0)

```

```
88: }
89:
90: plotar.stats.max.png <- function(resultados, nome.arquivo,
    limites.y = c(0, 1)) {
91:   png(filename = nome.arquivo, 900, 600, "px", pointsize = 14)
92:   plotar.stats.max(resultados, limites.y)
93:   dev.off()
94: }
```

Código-fonte 8 – Scores.R

```
1: RTS <- function(FR, PAS, GCS) {
2:   score.FR = 0
3:   score.PAS = 0
4:   score.GCS = 0
5:
6:   pesos = c(0.2908, 0.7326, 0.9368)
7:
8:   if (FR >= 29) {
9:     score.FR = 3
10:  } else if (FR >= 10) {
11:    score.FR = 4
12:  } else if (FR >= 6) {
13:    score.FR = 2
14:  } else if (FR >= 1) {
15:    score.FR = 1
16:  }
17:
18:   if (PAS >= 89) {
19:     score.PAS = 4
20:  } else if (PAS >= 76) {
21:    score.PAS = 3
22:  } else if (PAS >= 50) {
23:    score.PAS = 2
24:  } else if (PAS >= 1) {
25:    score.PAS = 1
26:  }
27:
28:   if (GCS >= 13) {
29:     score.GCS = 4
30:  } else if (GCS >= 9) {
31:    score.GCS = 3
```

```
32: } else if (GCS >= 6) {
33:   score.GCS = 2
34: } else if (GCS >= 4) {
35:   score.GCS = 1
36: }
37:
38: return(score.FR * pesos[1] + score.PAS * pesos[2] + score.GCS
        * pesos[3])
39: }
40:
41: ISS <- function(AIS = c(1, 1, 1)) {
42:   if (any((AIS == 6))) {
43:     ret = 75
44:   } else {
45:     ret = sum(AIS ^ 2)
46:   }
47:
48:   return(ret)
49: }
50:
51: TRISS.precalc <- function(RTS, ISS, idade, blunt = TRUE,
                           limiteIdade = 55,
52:                           b = rbind(c(-1.2470, 0.9544, -0.0768,
53:                                       -1.9052),
54:                                       c(-0.6029, 1.1430, -0.1516,
55:                                       -2.6676))) {
56:   if (idade >= limiteIdade) {
57:     age = 1
58:   } else {
59:     age = 0
60:   }
61:
62:   if (blunt) {
63:     i = 1
64:   } else {
65:     i = 2
66:   }
67:
68:   B = b[i, 1] + b[i, 2] * RTS + b[i, 3] * ISS + b[i, 4] * age
69:
70:   return(1 / (1 + exp(-B)))
71: }
```

```
70:
71: TRISS <- function(FR, PAS, GCS, ISS, idade, blunt = TRUE,
    limiteIdade = 55,
72:                    b = rbind(c(-1.2470, 0.9544, -0.0768,
    -1.9052),
73:                               c(-0.6029, 1.1430, -0.1516,
    -2.6676))) {
74:   RTS = RTS(FR, PAS, GCS)
75:
76:   return(TRISS.precalc(RTS, ISS, idade, blunt, limiteIdade, b))
77: }
```