2GCMA: Um gerador automático de modelos causais baseado em algoritmo evolutivo

Leandro L. A. Vieira, Gabriel T. P. Coimbra, Fabrício A. Silva

Universidade Federal de Viçosa - Campus Florestal, Brasil
{leandro.lazaro,gabriel.coimbra, fabricio.asilva}@ufv.br

Resumo. Apesar da associação ser uma ferramenta poderosa para os negócios, ela nem sempre indica causalidade. Por isso, nos últimos anos muito tem se estudado e pesquisado a respeito dessa área. Como consequência, novos problemas surgiram. Dentre eles a geração automatizada de modelos causais, parte crucial do estudo da causalidade. Apesar de existirem ferramentas que tentam atender a essa demanda, para casos envolvendo muitas variáveis elas não o fazem tão bem. Por isso, neste trabalho é proposta uma solução eficiente baseada em algoritmo evolutivo para sugestão de modelos causais dedicada principalmente para casos envolvendo um grande número de parâmetros. A solução foi avaliada em um conjunto de dados com 14 mil entradas e 63 dimensões. Os resultados mostraram que foi possível criar um modelo causal relevante utilizando a abordagem de algoritmo evolutivo, enquanto duas ferramentas da literatura não alcançaram resultados satisfatórios. Como trabalhos futuros, pretende-se aperfeiçoar a ferramenta proposta neste trabalho com a finalidade de aumentar sua precisão e eficácia na descoberta de efeitos causais.

1. INTRODUÇÃO

Ao longo dos anos, o desenvolvimento de ferramentas e mecanismos computacionais para descobrir associação entre variáveis tem sido promissor [6]. No entanto, correlacionar eventos nem sempre fornece o entendimento necessário acerca de um problema, pois correlação não necessariamente indica causalidade. Apesar dessa afirmação causar confusão para os que nunca tiveram contato com ela, causa e correlação são bastante diferentes embora sejam facilmente confundidas. Ferramentas dedicadas a descoberta de associações geralmente não são muito complexas, já que em resumo, o objetivo final é encontrar uma função que melhor represente o alinhamento entre duas ou mais variáveis [13]. O problema da causalidade é que apenas essa abordagem não é suficiente, sendo necessário enfrentar desafios adicionais para afirmar se uma variável tem relação causal com outra. Um desses grandes desafios é criar modelos causais que representem com alguma credibilidade uma relação causal [4], pois muitas vezes ainda é necessário gerar esses modelos manualmente, principalmente devido a natureza complexa do problema. Tais modelos são grafos direcionados que não podem ser cíclicos devido a limitações de algumas ferramentas utilizadas. Além disso, um outro significativo problema envolve os próprios dados usados para realizar as descobertas, já que a identificação de causa muitas vezes requer experimentos controlados, como testes A/B. Porém, esses experimentos podem ser caros, antiéticos, e muitas vezes inviáveis devido aos eventos já terem ocorrido. Portanto, um outro desafio é o de identificar causalidade com base em dados históricos.

Felizmente existem ferramentas poderosas que prometem auxiliar na descoberta dessas relações, como algoritmos de descoberta de modelos causais e bibliotecas de estimativa desses modelos. Essas ferramentas usam técnicas avançadas que permitem muitas vezes identificar causa e efeito mesmo em conjunto de dados observacionais. O problema das ferramentas que se propõem a construir modelos causais é que nem sempre conseguem cumprir sua função, pois não funcionam bem para grande volume de variáveis, seja por sua complexidade computacional, pela construção de grafos confusos ou por algumas vezes gerar grafos cíclicos, grafos que em nenhuma hipótese podem seguir para a etapa de inferência de causa e efeito.

Com o objetivo de tentar preencher a lacuna da criação de modelos causais automáticos, minimamente factíveis e eficientes para qualquer volume de variáveis, será apresentado nesse trabalho o 2GCMA (Genetic Algorithm Generator of Causal Models), um algoritmo evolutivo para geração de modelos causais. O objetivo do algoritmo é sugerir um número significativo de modelos minimante realistas ordenados em função de sua relevância para que o usuário possa interpretá-los e a partir destes construir manualmente um modelo causal definitivo. Ou seja, com o 2GCMA é possível gerar vários modelos baseados no conjunto de dados fornecidos para o algoritmo. Para validar nossa solução utilizamos um conjunto de dados reais de *Churn* de um banco digital e utilizamos outro algoritmo

de geração de modelos causais para fins de comparação.

O trabalho foi organizado da seguinte forma: na seção 2 estão descritos os trabalhos relacionados; na seção 3 estão detalhados o desenvolvimento e funcionamento da solução; a seção 4 fala a respeito dos dados de teste e resultados obtidos e, por fim; a seção 5 são as considerações finais.

2. CONCEITOS E TRABALHOS RELACIONADOS

Para descoberta de eventos causais é necessário dividir o problema em três etapas [9]: elaboração de um modelo causal, inferência do efeito causal do modelo e a refutação. O modelo causal consiste em uma forma intuitiva de se representar a suposta relação causal entre a variável de tratamento e as variáveis auxiliares em relação a variável objetivo. Apesar da variável de tratamento ser o nosso principal objeto de investigação, as variáveis auxiliares nos permitem diminuir ou eliminar o viés das análises, isto é, os indivíduos são agrupados em relação às variáveis auxiliares e é analisado o impacto médio causado pela diferença da variável de tratamento na variável objetivo.

A inferência do efeito causal estima o efeito que a variável de tratamento tem na variável objetivo. Ou seja, calcula a probabilidade da variável de tratamento causar a variável objetivo. Essa estimativa é feita com base no modelo causal. Existem alguns algoritmos que podem ser usados para realizar essa etapa, dentre eles a regressão linear [1].

Talvez a refutação seja a etapa mais importante. Nela, a estimativa obtida no passo anterior é testada por meio de algoritmos que modificam os dados e o modelo na tentativa de alterar o resultado da inferência. No teste de variável placebo, por exemplo, caso a nova estimativa retorne um valor próximo de zero, significa que o modelo é útil, caso contrário, o modelo não é útil, isto é, a variável de tratamento não tende a causar a variável objetivo.

Existem algumas ferramentas que permitem realizar as etapas de inferência e refutação de forma eficiente. Uma delas é o DoWhy [10], uma biblioteca de código aberto mantida pela Microsoft e escrita em Python para inferência causal. Essa biblioteca fornece desde uma estrutura de dados para representação dos modelos causais até algoritmos de inferência e refutação. Na figura 1 é possível visualizar as etapas realizadas pela ferramenta.

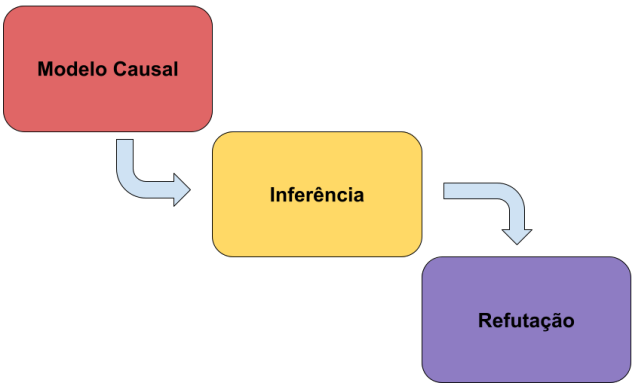

Fig. 1. Fluxo de execução de um algoritmo de estimativa causal

O problema que envolve o método de descoberta causal descrito está concentrado na etapa de construção do modelo causal. Atualmente não existem técnicas eficientes para geração de modelos causais apesar de já existirem alguns algoritmos que tentam fazer isso, como o *Causation, Prediction, and Search* (PC) [12], *Greedy Equivalence Search* (GES) [2], *Non-gaussian Linear Causal Models* (LINGAM) [11].

Atualmente o algoritmo PC possui implementações um pouco diferentes da versão original dedicadas principalmente a melhorar seu desempenho [7]. Isso se deve em grande parte ao fato do PC ser o primeiro algoritmo que se propôs a encontrar relações causais automaticamente. O algoritmo PC inicia-se com um grafo cíclico completo. Em seguida ele realiza exaustivos testes de independência condicional para cada umas das arestas. Se duas variáveis são condicionalmente independentes, o algoritmo elimina a aresta que existe entre elas. Esse procedimento

é repetido até que todas as combinações de testes sejam executadas e ao final da execução o resultado é um grafo contendo a suposta relação causal entre as variáveis que passaram pelo teste de independência condicional.

Diferente do algoritmo PC, GES [2] não inicia com um grafo cíclico e completo. Ao invés disso, inicia-se com um grafo vazio e adiciona e remove arestas de acordo com um conjunto de regras. A primeira etapa é inserir arestas com base em alguma métrica, como *Bayesian Information Criterion* (BIC) [14] por exemplo. Ao longo de cada ciclo de execução o algoritmo verifica qual aresta, caso inserida no grafo, diminuirá o valor do BIC. Quando não é mais possível melhorar o BIC com inserção de arestas, o algoritmo parte para a segunda etapa que consiste em retirar arestas caso a remoção impacte positivamente no ajuste.

O LINGAM [11] é o mais recente e popular algoritmo do gênero proposto para resolver a demanda de geração de modelos causais. Basicamente o LINGAM utiliza da mesma técnica do algoritmo PC, que é encontrar as dependências entre as variáveis do grafo completo cíclico inicial, somado ao uso de técnicas avançadas de manipulação da matriz do grafo para encontrar o melhor modelo.

O problema desses algoritmos é que por vezes acabam gerando modelos que sequer podem passar para a etapa de inferência, pois podem acabar gerando modelos cíclicos, modelos que não podem passar para a etapa de inferência. Além disso, esses métodos não são eficientes para um grande número de variáveis devido a sua complexidade computacional e ao fato de gerarem um modelo muito complexo ao final de sua execução, fazendo com que sua interpretação não seja possível.

Somando-se a isso, gerar manualmente modelos causais, além de ser uma tarefa exaustiva, exige algum tipo de conhecimento prévio do negócio para que os modelos sejam no mínimo coerentes. Tal geração de modelos pode ser tendenciosa, devido a preconceitos do usuário da ferramenta de causalidade. Caso isso aconteça, a etapa de refutação mostrará tal viés. Sendo assim, o usuário terá que gerar outro modelo, que novamente poderá ser refutado. Assim, tal uso poderá se tornar extremamente exaustivo.

3. GERADOR AUTOMÁTICO DE MODELO CAUSAL GENÉTICO

Para atender melhor a demanda da geração automática de modelos causais, foi criado o 2GCMA (Gerador Automático de Modelo Causal Genético), um algoritmo evolutivo que permite sugerir um conjunto de modelos causais minimamente factíveis, isto é, não são obtidos de forma totalmente aleatória, mas com base em uma função de *fitness*. Para o funcionamento do mesmo, a única etapa que deve ser realizada manualmente é a configuração da variável objetivo, o número de gerações, que nesse caso representa também o critério de parada, e de indivíduos por geração, o que é uma necessidade comum aos algoritmos genéticos.

Algorithm 1: 2GCMA

```
data; subjects;
selected_subjects;
max_vertices; subjects_num;
generations_num;
fitness_inferior_limit;
generate_first_subjects(subjects, data, subjects_num, max_vertices);
select_subjects(subjects, selected_subjects  fitness_inferior_limit);
while generations_num do
    cross_subjects(subjects, data, max_vertices);
    select_subjects(subjects, selected_subjects, fitness_inferior_limit);
end
```

As primeiras sete linhas do Algoritmo 1 (2GCMA) representam respectivamente o conjunto de dados analisado, a variável de armazenamento dos indivíduos de cada geração, a variável de armazenamento dos melhores indivíduos selecionados ao longo de cada geração, o número máximo de vértices permitido para o grafo de cada indivíduo, o número de indivíduos por geração, o número de gerações e um limite inferior mínimo destinado para o controle dos melhores indivíduos. A função *generate_first_subjects* cria aleatoriamente a geração inicial de indivíduos em função das variáveis fornecidas. No caso de um *dataset*, o nome das variáveis nada mais é do que o nome das colunas com exceção da coluna da variável objetivo. A função *select_subjects* seleciona os melhores indivíduos de cada geração e armazena na variável *selected_subjects*. Em termos de algoritmos genéticos, essa função seleciona os indivíduos com base em um elitismo [3], ou seja, seleciona apenas os melhores indivíduos em

função do limite inferior definido pela variável *fitness_inferior_limit*.

A função *cross_subjects* realiza os cruzamentos entre os indivíduos sendo que o número de cruzamentos é definido também pela variável *subjects_num*. Para tornar o cruzamento eficiente e não tendencioso, utilizamos o algoritmo de roleta [5] para selecionar os pais de cada um dos cruzamentos. Esse algoritmo prioriza o cruzamento de indivíduos com *fitness* melhores sem ignorar o cruzamento de indivíduos com *fitness* piores, isto é, soluções com *fitness* maiores tem mais chance de permanecerem no conjunto de soluções escolhidas, enquanto que soluções com *fitness* menor tem menos chances de permanecerem. Dessa forma é possível manter a diversidade do algoritmo e maximizar o número de possibilidades exploradas. Além disso, a função *cross_subjects* conta com uma taxa de mutação por indivíduo de 1%. Isso quer dizer que cada indivíduo novo gerado possui uma chance de adquirir vértices que não pertenciam aos indivíduos que o originaram.

Por último mas não menos importante, o código1 representa a estrutura de dados de cada indivíduo. A variável *outcome* representa a variável objetivo. *treatment*, a variável de tratamento, isto é, a variável objeto de estudo. *common_causes* contém um vetor de causas comuns, isto é, vértices que causam tanto a variável objetivo quanto a variável de tratamento. *instruments* armazena o conjunto de vértices que causam exclusivamente a variável de tratamento. Por último, o *fitness* carrega consigo o valor probabilístico causal do modelo que é também utilizado para mensurar a qualidade do indivíduo.

```
{
    outcome:String,
    treatment:String,
    common_causes:[],
    instruments:[],
    fitness:float
}
```

Listing 1. Estrutura de dados dos indivíduos

Nesse cenário, o cálculo do valor de *fitness* é uma métrica importante para se gerar boas soluções. Como se tratam de modelos causais, era necessário alguma ferramenta que pudesse mensurar a qualidade desses modelos. Com base nessa necessidade, foi utilizado o já citado DoWhy[1]. O DoWhy fornece 7 métodos distintos para fazer a estimativa da inferência. No caso foi decidido por utilizar o algoritmo *Propensity Score Stratification* [8] devido ao seu bom desempenho e boa precisão. Esse algoritmo agrupa os indivíduos baseado no cálculo da probabilidade (propensão) de cada indivíduo receber um tratamento. Em seguida ele divide os indivíduos em subgrupos baseado nos quartis da pontuação de propensão, compara os subgrupos que receberam tratamento com os que não receberam tratamento e retorna um valor equivalente ao impacto da variável de tratamento na variável objetivo do modelo. Uma limitação desse método é permitir apenas variáveis discretas como variáveis de tratamento, o que implica em uma limitação acerca de nossa ferramenta em permitir execuções apenas para conjuntos de dados binários.

O *fitness* é baseado no valor de retorno da função *Propensity Score Stratification* para cada modelo e embora essa seja uma forma eficiente de calcular o *fitness* e consequentemente encontrar bons modelos, esse método exige um tratamento adicional. Como foi explicado anteriormente, todo modelo deve passar por duas etapas, inferência e refutação. Nesse caso o *fitness* é gerado sem executar qualquer algoritmo de refutação em seguida. Isso implica que dentre os modelos resultantes, alguns não serão factíveis. Para mitigar esse problema, ao final da execução do algoritmo genético, foi aplicado em cada um dos indivíduos resultantes os algoritmos de refutação *Random Common Cause*, *Data Subset Refuter* e *Placebo Treatment Refuter*. O primeiro teste adiciona uma variável aleatória independente como causa comum no modelo e, se o modelo estiver correto, o valor retornado será próximo ao valor da inferência. O segundo método substitui o conjunto de dados por um subconjunto selecionado aleatoriamente. Dessa forma, ao invés da estimativa ser feita em função do conjunto original, é feita em função do novo conjunto de dados. Se o modelo for factível, o resultado deve ser próximo ao resultado da inferência. Por último, o terceiro método substitui a variável de tratamento por uma variável placebo gerada aleatoriamente. Diferente dos outros dois métodos, nesse caso, se o valor retornado não for igual a zero, significa que o modelo não é factível.

Apesar de idealmente ser necessário executar os métodos de refutação para cada modelo ao longo da execução do 2GCMA, optou-se por executar somente nos indivíduos resultantes devido ao elevado custo computacional desses algoritmos. Embora alguns indivíduos do conjunto de resultado sejam descartados, isso não gera um

[1] https://github.com/py-why/dowhy

grande problema, pois muitos ainda serão preservados. A Figura 2 representa de forma intuitiva o fluxograma de execução da solução proposta neste trabalho

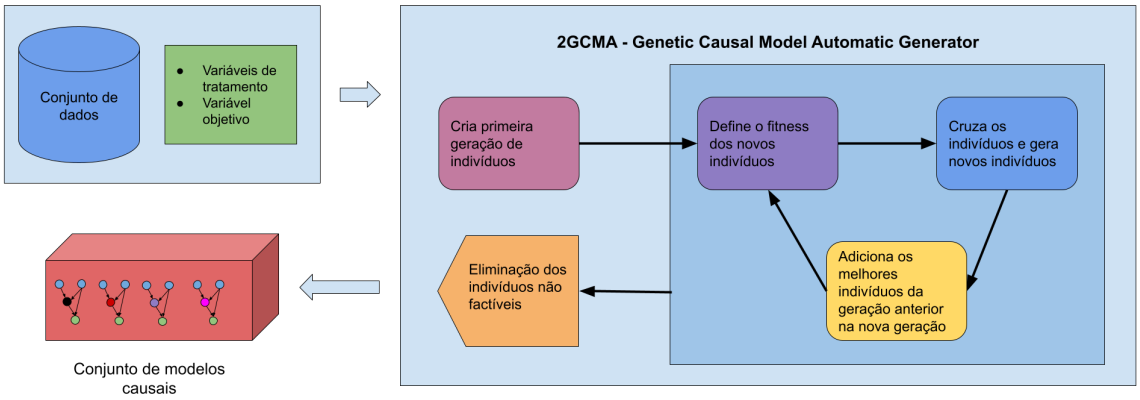

Fig. 2. Fluxograma de execução do algoritmo 2GCMA

4. AVALIAÇÃO

Para avaliar a solução proposta, foi considerado o problema de identificação de causa de *Churn* (i.e., abandono de serviço) de um banco digital.

4.1 Os Dados

Os dados foram obtidos após assinatura de um contrato de responsabilidade com uma empresa relacionada. A coleta ocorreu no *smartphone* de cada cliente e enviado para servidores centrais para armazenamento. É importante destacar que, somente após a instalação do aplicativo do banco digital com agente de coleta embutidos, os dados começam a ser coletados.

A base de dados contém 47,665 usuários, coletados entre 01-01-2021 até 30-07-2021. Além disso, para decisão de quais usuário são *Churners*, foi coletada informação de qual foi a data da última abertura do aplicativo por parte de cada cliente. Os *Churners* foram definidos como os usuários que ficaram 5 meses sem acessar o banco de digital (entre 30-07-2021 a 31-01-2022) pois é desejável ter alta precisão na classificação de quais usuários são *Churners*. Os cinco meses vão ser suficiente para considerar como *Churners* mesmo boa parte dos clientes que acessam pouco frequentemente o banco digital devido ao tipo de interação com o banco (i.e., conta de investimentos de longo prazo que precisa de pouco monitoramento). Devido a essa definição, nosso trabalho tende a detectar causas de *Churn* utilizando tendências de comportamento.

As informações coletadas pelo agente são: a data de acesso do aplicativo do banco digital, a lista de aplicativos instalados, informações que resumem o padrão de acesso ao aplicativos (i.e., estatísticas descritivas do número de sessões únicas, tempo de duração de cada sessão do usuário no banco digital). Além disso, também é usada a informação de qual o *smartphone* o usuário utiliza e o preço estimado do mesmo. A base de dados também foi enriquecida utilizado dados da Google Play que permitem saber dados dos aplicativos instalados pelo usuário como: categoria, popularidade (i.e., número de instalações).

4.2 Resultados

Para executar a solução proposta no conjunto de dados descrito anteriormente, foram excluídos todas as colunas que não fossem relevantes, como número de identificação do cliente, por exemplo. Em seguida foi balanceado o conjunto utilizando-se de subamostragem com base na variável indicativa de *Churn*, que é a variável objetivo. Foram categorizados os dados não binários e excluídos aqueles que não poderiam ser categorizados. Por último, foram convertidos todos os dados em binários, foi feita a média dos valores de cada coluna e excluídos os atributos

que possuíam uma média abaixo de 0.1, isto é, possuíam apenas menos de 10% dos valores da atributo como sendo verdadeiros. Essa decisão foi tomada ao perceber que essas variáveis não ajudavam na criação de modelo pois causavam ruído.

Após alguns testes, percebeu-se que, para o conjunto de dados em questão, 100 indivíduos e 100 gerações eram o suficiente para encontrar bons modelos causais. Após a execução foi constatado que na maioria dos casos os bancos digitais concorrentes eram os principais responsáveis pelo abandono do banco digital alvo de estudo. Em geral, caso o usuário utilize um banco digital concorrente, a probabilidade de que ele abandone o outro banco aumenta 14%. Na tabela[2] I é possível visualizar os principais causadores de *Churn* com seu respectivo impacto com base na execução de nosso algoritmo.

Contudo, como era esperado o 2GCMA também acusou como causadores outras variáveis que a princípio não fazem sentido, como um aplicativo de mensagem. Isso talvez tenha acontecido pelo fato do banco de dados não ter sido construído com base em estudos randomizados, que são o padrão ouro da análise causal, mas com base em estudos observacionais. Dessa forma, é comum encontrar variáveis espúrias dentre variáveis que aparentam realmente causar a variável objetivo. Ou seja, apesar de ter sido proposto um algoritmo gerador de modelos causais, ainda não é dispensável uma análise interpretativa acerca dos resultados.

Variáveis de tratamento	Aumento na chance de churn
Banco digital A	14.82%
Banco digital B	14.72%
Banco digital C	14.02%
Banco digital D	14.01%
Aplicativo de mensagem	13.34%

Table I. A tabela mostra a relação causal entre a variável de tratamento e a variável objetivo

A fim de realizar uma comparação entre a ferramenta proposta neste artigo e outras já existentes, foram executados os algoritmos GES e LINGAM para o conjunto de dados de teste. O algoritmo GES sequer conseguiu concluir sua execução devido a elevada complexidade computacional ocasionada pelo elevado número de dados e de dimensões. Isso mostrou que, apesar de ser uma boa solução, não é aplicável para dados volumosos. Diferente do GES, o LINGAM conseguiu concluir sua execução para o conjunto de dados de teste, porém o que foi percebido é que ele não gerou bons resultados, já que se tratava de um grande volume de dados, como é o caso dos dados utilizados neste trabalho. O grafo gerado foi de 57 vértices e 1140 arestas, tornando inviável interpretar o modelo gerado devido a sua complexidade. Além disso, ambos os algoritmos não permitem definir uma variável objetivo, ou seja, mesmo que seja gerado um bom modelo, a variável objetivo definida por ele pode não ser a variável de interesse que se deseja avaliar.

A solução proposta, apesar de não gerar um modelo definitivo, gera um conjunto de modelos simples, de fácil interpretação e que sempre serão direcionados para a variável objetivo previamente definida pelo usuário. Na Figura 3, é possível visualizar o melhor modelo gerado pelo 2GCMA com base no impacto da variável de tratamento na variável objetivo. Porém, os resultados foram coerentes, sendo que os Bancos A e B são concorrentes importantes do banco analisado para *Churn*, e o aplicativo Y é um aplicativo de mensagem com menos relevância.

5. CONCLUSÕES E TRABALHOS FUTUROS

Com este trabalho foi possível desenvolver, através da abordagem de algoritmo evolutivo, uma forma eficiente de sugerir modelos causais que possam ser utilizados como ferramenta de criação de um modelo causal definitivo. O grande diferencial observado foi a construção de modelos simples e facilmente interpretáveis, com destaque principalmente para quando a descoberta envolve um grande volume de dados, que é um elemento limitante para as ferramentas similares da literatura.

Como trabalhos futuros, espera-se aperfeiçoar a solução elevando a sua eficiência e precisão no que se trata de gerar modelos cada vez mais factíveis na tentativa de se aproximar ainda mais do objetivo de eliminar completamente a interferência manual durante o processo de descoberta causal.

[2]Os nomes dos bancos foram omitidos por questões de privacidade.

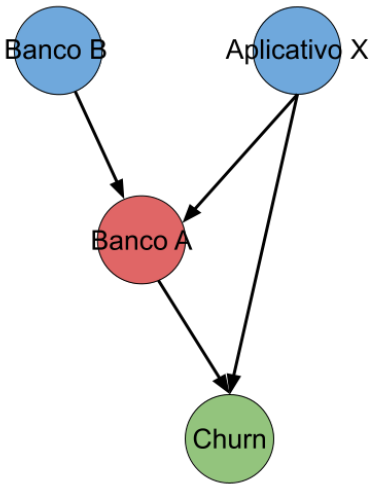

Fig. 3. Modelo da Tabela I com maior impacto causal entre a variável de tratamento (em vermelho) e a variável objetivo (em verde)

References

[1] Ambarish Chattopadhyay and Jose R Zubizarreta. “On the implied weights of linear regression for causal inference”. In: *arXiv preprint arXiv:2104.06581* (2021).

[2] David Maxwell Chickering. “Optimal structure identification with greedy search”. In: *Journal of machine learning research* 3.Nov (2002), pp. 507–554.

[3] Haiming Du et al. “Elitism and distance strategy for selection of evolutionary algorithms”. In: *IEEE Access* 6 (2018), pp. 44531–44541.

[4] Clark Glymour, Kun Zhang, and Peter Spirtes. “Review of causal discovery methods based on graphical models”. In: *Frontiers in genetics* 10 (2019), p. 524.

[5] David E Golberg. “Genetic algorithms in search, optimization, and machine learning”. In: *Addion wesley* 1989.102 (1989), p. 36.

[6] Markus Hegland. “The apriori algorithm–a tutorial”. In: *Mathematics and computation in imaging science and information processing* (2007), pp. 209–262.

[7] Thuc Duy Le et al. “A fast PC algorithm for high dimensional causal discovery with multi-core PCs”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 16.5 (2016), pp. 1483–1495.

[8] Jared K Lunceford and Marie Davidian. “Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study”. In: *Statistics in medicine* 23.19 (2004), pp. 2937–2960.

[9] Judea Pearl. “Causal inference”. In: *Causality: objectives and assessment* (2010), pp. 39–58.

[10] Amit Sharma and Emre Kiciman. “DoWhy: An end-to-end library for causal inference”. In: *arXiv preprint arXiv:2011.04216* (2020).

[11] Shohei Shimizu et al. “A linear non-Gaussian acyclic model for causal discovery.” In: *Journal of Machine Learning Research* 7.10 (2006).

[12] Peter Spirtes et al. *Causation, prediction, and search*. MIT press, 2000.

[13] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. “Linear regression”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.3 (2012), pp. 275–294.

[14] Scott I Vrieze. “Model selection and psychological theory: a discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC).” In: *Psychological methods* 17.2 (2012), p. 228.