UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL S. BEVILAQUA

ELEIÇÕES MAIS JUSTAS: SIMULANDO E INTERPRETANDO DIFERENTES SISTEMAS DE VOTAÇÃO

RIO DE JANEIRO
2019

GABRIEL S. BEVILAQUA

ELEIÇÕES MAIS JUSTAS: SIMULANDO E INTERPRETANDO DIFERENTES SISTEMAS DE VOTAÇÃO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Vinícius Gusmão Pereira de Sá

RIO DE JANEIRO

2019

CIP - Catalogação na Publicação

Bevilaqua, Gabriel Santos
B571e Eleições mais justas: simulando e interpretando diferentes sistemas de votação / Gabriel Santos Bevilaqua. -- Rio de Janeiro, 2019.
80 f.

Orientador: Vinícius Gusmão Pereira de Sá.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Sistemas Eleitorais. 2. Simulador. 3. Média de Satisfação. 4. Página Web. I. Sá, Vinícius Gusmão Pereira de, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

GABRIEL S. BEVILAQUA

ELEIÇÕES MAIS JUSTAS: SIMULANDO E INTERPRETANDO DIFERENTES SISTEMAS DE VOTAÇÃO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 09 de SETEMBRO de 2019

BANCA EXAMINADORA:

Prof. Vinícius Gusmão Pereira de Sá
D.Sc. (UFRJ)

Prof. Adriana Santarosa Vivacqua
D.Sc. (UFRJ)

Prof. Juliana Vianna Valério
D.Sc. (UFRJ)

AGRADECIMENTOS

Gostaria de agradecer primeiramente à UFRJ que me permitiu embarcar nessa jornada pela busca de conhecimento nesse fascinante campo que é a computação. Ao meu orientador, Vinícius Gusmão Pereira de Sá pela atenção e ajuda no esclarecimento dos melhores caminhos a se tomar na elaboração deste trabalho. Aos meus pais e a toda minha familia, pelo incentivo e apoio incondicional. A minha namorada, pela paciência e compreensão nesta complicada fase da minha formação acadêmica. E aos meus amigos que ja passaram ou que estão passando pelo mesmo processo de conclusão que eu e que compartilham das mesmas dificuldades.

"Elections are won by men and women chiefly because most people vote against somebody rather than for somebody."

Franklin P. Adams

RESUMO

Existem diversos tipos de sistemas eleitorais em uso ao redor do mundo. Cada país opta pelo método que melhor se adequa ao seu modo de governar, muitas vezes utilizando variações de um mesmo método ou até uma combinação destes, dependendo do cargo político em questão. Através da implementação de um programa que permite simular um extenso número de eleitores, esta tese procura descobrir quais métodos trazem resultados que maximizam a taxa de satisfação média da população quando testados sob diferentes cenários políticos. Cada eleitor simulado atribuirá notas para os candidatos que serão geradas, dependendo do cenário, de maneira aleatória, seguindo certa distribuição de probabilidade, ou deterministicamente através de perfis eleitorais. Também serão incluídos nas simulações fatores reais de uma eleição, como o voto útil e as coalizões entre candidatos, para que seja possível determinar de que maneira esses parâmetros influenciam no resultado final. Será possível perceber que alguns dos sistemas são muito eficazes na seleção dos melhores candidatos, porém afetados pelos votos táticos enquanto outros são resistentes a esses, mas pecam em eficiência.

Palavras-chave: Simulações. Sistemas Eleitorais. Satisfação Média.

ABSTRACT

There are several types of electoral systems in use around the world. Each country chooses the method that best fits their governance, often using variations of the same method or even a combination of these, depending on the political position in question. Through the implementation of a program that simulates an extensive number of voters, this thesis seeks to find out which methods bring results that maximize the rate of population satisfaction when tested under different political scenarios. Each simulated voter will assign notes to the candidates that will be generated, depending on the scenario, in random fashion, following a certain probability distribution, or deterministically through electoral profiles. Also included in the simulations are the real factors of an election, such as tactical votes and coalitions between candidates, so that it is possible to determine how these parameters influence the final result. It will be possible to see that some of the systems are very effective in the selection of the best candidates, however affected by tactical votes while others are resistant to these, but lack efficiency.

Keywords: Simulations. Electoral Systems. Satisfaction Rate.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FPTP	First Past The Post
TRS	Two-Round System
IRV	Instant Runoff Voting
AVS	Approval Voting System
BC	Borda Count
SVS	Score Voting System
BV	Bloc Vote

SUMÁRIO

1 INTRODUÇÃO

Em toda eleição existe pelo menos um grupo de indivíduos que sai completamente desfavorecido e insatisfeito com o resultado. O contentamento completo de uma população é certamente improvável e muitas vezes impossível considerando a gama de candidatos que se dispõe a concorrer. Em todo caso, o mínimo que pode ser feito é escolher de maneira eficiente o melhor candidato ao cargo dentre as opções disponíveis. Diante disso é necessário avaliar como essa escolha é feita e quais métodos são capazes de torná-la realidade.

1.1 MOTIVAÇÃO E OBJETIVO

Em conformidade com o extenso número de sistemas eleitorais pelo mundo, este trabalho tem como objetivo testar uma porção desses sistemas sob a simulação de diferentes cenários específicos a fim de determinar quais deles podem ser considerados como os mais justos perante a taxa de satisfação média da população.

1.2 MÉTODO

Para que essa análise seja possível, foi desenvolvido um programa que permite a criação e configuração de diferentes cenários, onde os sistemas eleitorais serão simulados. Nele, é possível definir o número total de eleitores e candidatos, como esses eleitores irão votar e o nível de popularidade de cada candidato. Além disso, é possível formar coalizões entre os candidatos e escolher a porcentagem dos eleitores que optarão pelo voto útil. Dessa maneira, será possível elaborar cenários que buscam expor os pontos fortes e fracos de cada sistema.

1.3 ESTRUTURA

Este trabalho é relatado ao longo de seis capítulos. No Capítulo 2 serão apresentados os sistemas eleitorais implementados no simulador, suas vantagens e desvantagens conhecidas e outras características específicas de cada um. A intenção do capítulo é tornar o texto mais autocontido, trazendo conceitos já estudados que ajudarão na melhor compreensão dos capítulos seguintes, por isso grande parte de sua composição é oriunda de diferentes fontes consultadas.

O Capítulo 3 é uma apresentação das funcionalidades que o simulador possui. Serão explicados todos os campos de entrada e os dois modos distintos nos quais o programa opera.

No Capítulo 4 os sistemas serão colocados em teste. Todos serão simulados em cenários específicos a fim de fazer sobressair aqueles com as maiores taxas de satisfação média.

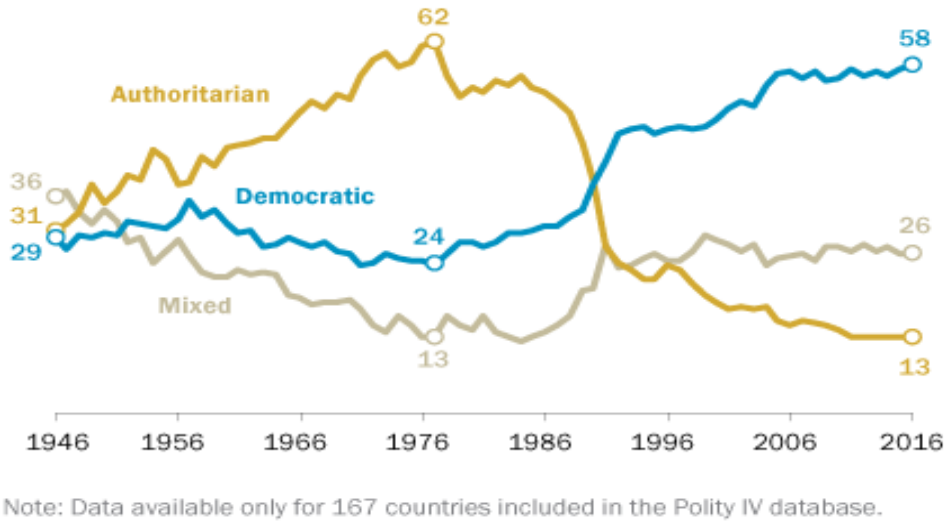
A estrutura e implementação do simulador serão abordadas em detalhes no Capítulo 5 assim como uma breve descrição da *stack* utilizada. Todos os métodos relevantes ao funcionamento das simulações serão explicados passo a passo.

No Capítulo 6 serão apresentadas as conclusões referentes à análise dos resultados obtidos no Capítulo 4 e as considerações finais do projeto como um todo.

2 SISTEMAS ELEITORAIS

Atualmente mais e mais países estão adotando sistemas democráticos de governo. Cerca de 6 em 10 governos no mundo são democracias (Pew Research Center, 2017), o que pode ser observado na Figura 1 abaixo. Em um sistema democrático, a população escolhe seus líderes através de seu sistema eleitoral. Existem diversos sistemas eleitorais em uso no mundo, muitos dos quais são usados em conjunto para eleger um único candidato. Em vários países mais de um sistema eleitoral pode ser utilizado também em diferentes níveis de governo, tanto presidencial quanto estadual ou municipal. Não apenas imersos no meio político, eles também podem ser usados em escopos menores, como em esportes ou competições artísticas, porém sempre com a finalidade de alocar um ou mais indivíduos a determinado cargo ou título.

Figura 1 – % de países de cada tipo de regime, 1946-2016



Fonte: Center of Systemic Peace's Polity IV Project

Os próximos três tópicos discutidos neste capítulo, referentes aos métodos *First Past the Post*, *Two-Round System* e *Instant Runoff Voting* possuem seus conceitos e informações relevantes, com exceção dos exemplos, traduzidos livremente do livro *Behind the Ballot Box: A Citizen's Guide to Voting Systems* (AMY, 2000).

2.1 FISRT PAST THE POST

O método *First Past the Post*, também conhecido como *Plurality Voting* ou *Winner Takes All*, é o método mais simples e direto e provavelmente o primeiro método que se pensa quando se fala de eleições. Os eleitores escolhem apenas um candidato e, após a

contagem dos votos, o candidato com o maior número de votos é eleito. O vencedor não necessita da aprovação de mais de 50% dos eleitores, ele apenas precisa de uma pluralidade dos votos, ou seja, apenas possuir mais votos que os outros candidatos.

Esse sistema é utilizado em escala nos Estados Unidos, para a maior parte das eleições voltadas a eleger apenas um candidato, como prefeitos e governadores, além de ser usada na disputa presidencial. Dentre outros países que empregam o sistema em suas corridas presidenciais estão listados o México, as Filipinas, a Coreia do Sul e a Venezuela.

2.1.1 Exemplo

Tabela 1 – Exemplo FPTP

Candidatos	Votos
A	36
B	25
C	22
D	17

Neste exemplo o candidato A possui uma pluralidade dos votos com 36% e ele é o candidato eleito mesmo não tendo uma maioria.

2.1.2 Vantagens e Desvantagens do FPTP

A vantagem do FPTP é a sua simplicidade. Não é difícil para um eleitor comum compreender o seu funcionamento, eles apenas precisam se decidir por um candidato. Ele também é muito simples e direto na sua contagem e seleção do vencedor.

Esse método também tende a criar listas de candidatos concorrentes mais curtas. Candidatos independentes ou de partidos menores frequentemente acabam se sentindo desencorajados a concorrer devido às suas baixas chances de vitória. Isso normalmente fornece aos eleitores uma fácil decisão entre um ou dois candidatos dos maiores partidos, ao mesmo tempo, porém, limitando suas opções.

No entanto, sua essência é sua principal desvantagem, a pluralidade. O fato do candidato eleito não necessitar de uma maioridade dos votos da população parece violar os princípios básicos da democracia. Democracia é, em sua origem, o poder(*kratos*) no povo(*demos*), porém, na pluralidade o "poder" de eleger um candidato pode residir em apenas uma fração relativamente pequena do "povo". Em um cenário hipotético, com apenas três candidatos, um deles apenas poderia necessitar de não mais do que 34% dos votos para ser eleito, o que significa que, no pior dos casos, 66% da população rejeita o candidato eleito. O cenário só piora com mais e mais candidatos concorrendo. Para quatro candidatos são necessários 26% dos votos no pior dos casos, para cinco são 21% e assim por diante.

2.1.3 Voto Útil no FPTP

No FPTP também existe a possibilidade de os eleitores evitarem votar em sua preferência verdadeira. No caso de sua preferência não aparentar ter boas chances de vitória, muitas vezes opta-se por uma segunda opção que esteja mais bem colocada nas eleições. O motivo disso é que nessas situações, votar em sua preferência pode parecer um desperdício do seu voto e, no pior dos cenários, optar por não transferir seu voto para uma segunda opção pode acabar resultando na eleição de um candidato de seu desgosto. Assim, votar de maneira verdadeira, muitas vezes pode acabar prejudicando o eleitor, que se vê forçado a abandonar seus interesses políticos mais imediatos e escolher o menor entre dois(ou mais) males. Os candidatos de partidos menores são os que mais sofrem com esse tipo de voto estratégico, por não conseguirem agregar as maiores quantidades de votos.

Entretanto, ao mesmo tempo que partidos maiores "roubam" votos de partidos menores com os votos táticos, partidos menores também tem a oportunidade de ganhar alguns votos com o denominado *spoiler effect*. Ao entrarem na eleição, candidatos de partidos menores podem adquirir votos de candidatos mais populares que possuem inclinação política semelhante. Logo, é possível que haja certo equilíbrio ao se contrapor as mudanças de voto devido aos votos úteis às mudanças oriundas do *spoiler effect*.

Porém, o *spoiler effect* não necessariamente ocorre apenas entre partidos maiores e menores, mas pode prejudicar candidatos de popularidades próximas. Por exemplo, entre três candidatos, A, B e C, onde A representa determinada posição no espectro político enquanto B e C possuem posições parecidas, o que pode acontecer é B e C se prejudicarem mutuamente já que ambos dividem os eleitores de um mesmo espectro político. Se esse espectro tem a maioria com 60%, por exemplo, com B e C compartilhando cada um 30% dos votos, o candidato A sairia vencedor com 40% mesmo que todos os eleitores de B prefiram C e todos os eleitores de C prefiram B. Esse exemplo demonstra claramente o problema com os sistemas de pluralidade.

2.2 TWO-ROUND SYSTEM

Uma das primeiras tentativas de lidar com a desvantagem dos sistemas por pluralidade, o *Two-round System* é usado desde o século XIX no ocidente e atualmente ainda é utilizado em corridas presidências de países como o Brasil, Bulgária, Chile, Colômbia, Finlândia, Polônia, Portugal, Rússia e em inúmeras eleições locais nos Estados Unidos.

O principal objetivo do TRS é garantir que o vencedor tenha conseguido uma maioria dos votos. Para isso as eleições são estruturadas em duas rodadas de votação. Na primeira rodada os eleitores se dirigem às urnas e votam no candidato de sua preferência. Após a contagem, se a quantidade de votos do candidato mais votado superar a proporção de 50% não há a necessidade de uma segunda rodada e este candidato é eleito. No entanto, se este não for o caso, haverá uma segunda rodada com apenas os dois candidatos mais

votados. Nesta rodada os eleitores retornam as urnas para escolher uma das duas opções, sendo eleito o candidato mais votado, agora definitivamente com uma maioria dos votos.

2.2.1 Exemplo

Utilizando a mesma distribuição de votos descrita na tabela 1. Neste caso, como o candidato A não possui uma maioria dos votos, ele e o candidato B irão concorrer em um segundo turno. Digamos que as distribuições dos votos dos eleitores dos candidatos C e D em relação aos candidatos A e B são as seguintes:

Tabela 2 – 2° turno do TRS

Candidatos	Votos dos eleitores de C	Votos dos eleitores de D	Votos do 1° turno
A	7	5	36
B	15	12	25

O candidato B agora é eleito com uma maioria dos votos, 52% contra 48% do candidato A.

2.2.2 Vantagens e Desvantagens do TRS

Por ser parecido com o FPTP, esse sistema também é de fácil compreensão do eleitor comum, porém, a maior vantagem do TRS é a garantia de que o candidato vencedor terá o apoio da maioria dos eleitores no segundo turno, eliminando a pluralidade. Argumenta-se que isso dá maior legitimidade política ao mandato do candidato eleito.

O TRS tende a abrigar uma lista maior de candidatos, sendo mais convidativo a candidatos de partidos menores, mesmo que suas chances não sejam maiores quando comparadas às eleições sob o FPTP. Mesmo assim, essa listagem mais abrangente é uma vantagem do sistema, pois dá aos eleitores uma variedade maior de opções de voto.

Em relação às suas desvantagens, o TRS possui duas principais. A primeira são os custos elevados do sistema decorrentes da necessidade da organização de um segundo turno de votações. A segunda é o aumento no índice de ausências dos eleitores às urnas. Essa ampliação no não comparecimento dos eleitores se dá também devido ao segundo turno por pura fatiga eleitoral, bem como consequência da desistência de eleitores que apoiam candidatos que já foram eliminados.

Outro ponto negativo do TRS é a sua não-monotonicidade. Isso significa que um candidato pode ser prejudicado com um aumento em sua popularidade e suporte e, do mesmo modo, um candidato pode se beneficiar com a perda de popularidade e suporte. Se os candidatos A e B são os esperados para ir ao segundo turno, onde A é mais forte do que B, um aumento no suporte pelo candidato A pode resultar na sua derrota quando esse aumento causa uma alteração nos candidatos que são selecionados para ir ao segundo turno. Se esse aumento na popularidade de A resultar na eliminação prévia de B, um

terceiro candidato C pode ser vitorioso sobre A no segundo turno, caso os eleitores de B prefiram C ao candidato A. Se não houvesse esse crescimento de popularidade, A teria derrotado B no segundo turno. Essa situação será vista em melhores detalhes na seção 4.3.

2.2.3 Voto Útil no TRS

Esse sistema minimiza os efeitos do voto útil em relação ao FPTP, pois um eleitor qualquer pode tranquilamente votar em sua preferência sem se preocupar em desperdiçá-lo mesmo que seu candidato não tenha chances de vitória. Se este não for eleito, ainda poderá escolher a sua preferência dentre os dois candidatos que prosseguirem para o segundo turno.

Apesar disso, ainda existe um cenário em que o eleitor pode ser tentado a não votar em sua preferência. É possível que ele opte pela sua segunda opção se souber que sua primeira opção possui mínimas chances de ir ao segundo turno ou até mesmo por saber que isso diminuirá as chances de algum outro candidato ser eleito. Mesmo assim, é evidente que o TRS não proporciona o mesmo apelo ao voto tático que o FPTP.

Outra característica do TRS é o incentivo a formação de coalizões entre candidatos de partidos mais populares com candidatos de partidos menos populares. As coalizões fornecem benefícios à ambas as partes, onde os candidatos classificados para o segundo turno, em geral, se juntam a candidatos já eliminados com a intenção de agregar mais votos enquanto os candidatos já eliminados apoiam um dos dois classificados na esperança de inserir propostas próprias ou modificações à chapa deles.

2.3 INSTANT-RUNOFF VOTING

Instant Runoff Voting foi uma alternativa aos métodos FPTP e TRS desenvolvida nos anos 70 por um professor do Instituto de Tecnologia de Massachusetts(MIT). O IRV elimina as principais desvantagens desses outros dois sistemas, a pluralidade do FPTP e a necessidade de mais de uma rodada de votação do TRS. Este método também opera em mais de uma rodada, porém, como o próprio nome sugere, essas rodadas são instantâneas não sendo necessário o retorno dos eleitores às urnas. Atualmente ele é utilizado na Irlanda nas eleições presidenciais, em Londres na eleição de seu prefeito e em distritos individuais na Austrália para eleger membros de sua câmara baixa do parlamento. O IRV também é usado por várias organizações privadas nos Estados Unidos, incluindo a *American Political Science Association* e a *American Psychological Association*, para eleger seus funcionários.

Nesse sistema os eleitores não são solicitados a votar em um único candidato, ao invés disso eles listam todos os candidatos concorrentes na ordem de sua preferência. A partir dessas listas as primeiras opções de cada eleitor recebem o seu voto e, após a contagem, se um candidato obtiver 50% dos votos mais um este candidato é eleito. Se

isso não ocorrer, o candidato menos votado é eliminado das eleições e seus votos são transferidos para a segunda opção de cada um de seus eleitores. Esse procedimento se repete, eliminando o candidato menos votado a cada rodada até que algum dos candidatos possuam a maioridade dos votos.

2.3.1 Exemplo

Na tabela 3 pode-se verificar a distribuição de votos por perfil de eleitor e na tabela 4 vê-se o desenvolvimento do IRV para essas distribuições. O candidato D é o menos votado e por consequência é o primeiro a ser eliminado. A segunda opção de todos os eleitores que votaram no candidato D é o candidato C, por isso este recebe todos os votos distribuídos na segunda rodada. Com isso o candidato B é eliminado com 25% dos votos e os candidatos A e C concorrem na última rodada. Mesmo com o candidato C na liderança na segunda rodada, a maior parte dos eleitores do B tem como preferência o candidato A, por isso A é eleito com 55% dos votos contra 45% do C.

Tabela 3 – Exemplo IRV

Rankings	N° de eleitores
A,B,C,D	20
A,B,D,C	16
B,D,A,C	19
B,D,C,A	6
C,A,B,D	7
C,D,B,A	15
D,C,A,B	5
D,C,B,A	12

Tabela 4 – Desenvolvimento do IRV

Turnos	D	C	B	A
1	17	22	25	36
2	eliminado	39	25	36
3	eliminado	45	eliminado	55

2.3.2 Vantagens e Desvantagens do IRV

Assim como no TRS, a pluralidade é irrelevante no IRV e um candidato só é eleito com a maioridade dos votos dando maior legitimidade ao mandato deste. No entanto, pode-se argumentar que o IRV faz isso com mais eficiência, pois não requer a execução de mais de uma rodada de votações. Isso diminui os custos das eleições como um todo ao mesmo tempo que proporciona um número menor de isenções por parte dos eleitores devido ao desgaste de todo o processo eleitoral.

Além disso, candidatos de partidos menores tendem a receber mais votos do que no FPTP e, portanto, são mais encorajados a se candidatar. Com uma lista maior de candidatos, eleitores desfrutam de uma variedade maior de opções de voto. Contudo, mesmo com mais candidatos menos populares, o *spoiler effect*, mencionado anteriormente, também não possui espaço nesse sistema de votação. Partidos menores não tem como "roubar" votos de partidos maiores visto que esses votos eventualmente serão transferidos de volta a eles durante o desenvolvimento das rodadas do IRV.

Em virtude de sua estrutura acredita-se que esse método tende a encorajar os candidatos também a elaborar suas campanhas de forma mais abrangente, ampliando sua influência para mais do que apenas um grupo específico de maneira que ele consiga agregar votos de segunda ou terceira opção além dos votos de seus eleitores principais.

A principal desvantagem do IRV é o fato de ele ainda não ser amplamente utilizado ao redor do mundo. Essa falta de familiaridade do sistema pode causar confusão ao eleitor comum em suas primeiras ocorrências. No entanto, nos países em que este é utilizado não parece haver confusão entre os eleitores. Além disso, na maioria dos casos haverá um custo potencialmente alto da substituição ou alteração dos sistemas das urnas eletrônicas para que seja possível haver a implementação do método.

Além disso, assim como o TRS, este sistema também possui não-monotonicidade, ou seja, um candidato pode ser prejudicado ao ser ranqueado mais alto enquanto que outro pode se beneficiar sendo ranqueado mais baixo pelos eleitores (ORNSTEIN, 2018), isso pode ocorrer quando esse ranqueamento causa alterações na ordem em que os candidatos são eliminados. Essa característica se manifesta em fenômenos complexos e específicos e por isso um exemplo de sua ocorrência será abordada em detalhes na seção 4.3.

2.3.3 Voto Útil no IRV

Como os sistemas de votação anteriores, o IRV não é completamente imune ao voto tático. No entanto, este método se mostra bem resistente a eles quando comparado a outros métodos como o FPTP e o *Borda Count* (Bartholdi, John J., III and Orlin, James B, 1990). Distintivamente, o IRV requer um maior esforço do eleitor para elaborar uma estratégia que melhor ajuste os resultados da eleição de acordo com a sua preferência. Além disso, é necessário que ele tenha informação suficiente sobre os rankings de outros eleitores, naturalmente obtida através de pesquisas eleitorais.

Existe também a possibilidade de ocorrer uma exaustão dos votos quando não há o completo preenchimento do ranking por parte dos eleitores. Nessas ocasiões o voto não pode continuar a ser transferido a outros candidatos e ele é descartado. Isso afeta a legitimidade do mandato do candidato eleito, pois o resultado final não representará 100% da população votante. Essas ocorrências, no entanto, podem ser minimizadas com uma melhor educação eleitoral ou, como foi implementado na Austrália, requerer que os rankings incluam uma totalidade dos candidatos concorrentes.

2.4 APPROVAL VOTING SYSTEM

No método *Approval Voting* os eleitores não são limitados pelo número de candidatos em que podem votar, por isso *approval*, cada eleitor pode votar em todos os candidatos os quais ele aprova. Após as votações o candidato com a maior quantidade de votos é eleito. Segundo Hamlin (2015), a cidade de Fargo, na Dakota do Norte se tornou a primeira cidade nos EUA a colocar em prática este sistema em suas eleições gerais em novembro de 2018. AVS também é utilizado para eleger o secretário-geral das Nações Unidas e em diversas outras organizações.

2.4.1 Exemplo

Na tabela 5 abaixo pode-se encontrar 8 perfis de eleitores com seus candidatos aprovados e quantidade de eleitores que se encaixam no perfil. Na tabela 6 temos o resultado da contabilização dos votos destes perfis. O candidato mais popular é o B que se encontra aprovado em 5 perfis distintos totalizando 60 votos ou aproximadamente 32%.

Tabela 5 – Exemplo AVS

Aprovações	N° de eleitores
A	20
A,B	16
B	19
B,D,C	6
C,A,B	7
C,D	15
D,C	5
D,C,B	12

Tabela 6 – Resultado AVS

Candidatos	Total de votos
A	43
B	60
C	45
D	38

2.4.2 Vantagens e Desvantagens do AVS

Além de simples, este método permite um nível maior de expressividade aos eleitores. Onde no FPTP e TRS eles são limitados pelo seu único voto, aqui os eleitores têm a liberdade de demonstrar exatamente quais candidatos consideram aptos ou aceitáveis ao cargo.

Outro efeito dessa liberdade trazida pelo AVS é a dizimação do *spoiler effect*. Candidatos de partidos menores não tem como "roubar" votos de partidos maiores, nesse sistema ambos os candidatos contabilizariam o voto.

2.4.3 Voto Útil no AVS

O tipo de voto útil no AVS é denominado *bullet voting*. Segundo The Center for Election Science (2015) esta tática envolve simplesmente ignorar a possibilidade de votar em mais de um candidato e votar apenas em sua primeira opção. Um eleitor pode ser levado a adotar essa estratégia quando percebe que sua preferência pode ser prejudicada com o voto em alguma segunda opção sua. Em um cenário extremo, onde 100% dos eleitores adotam esta estratégia as eleições são convertidas a um simples *First Past The Post*.

2.5 THE BORDA COUNT

Este método carrega o nome em homenagem a seu criador, Jean-Charles de Borda, que o desenvolveu em 1770 (LIPPMAN, 2012). A contagem de Borda utiliza a abordagem de ranqueamento dos candidatos, porém de maneira diferente ao IRV. Este é um sistema que atribui uma pontuação aos candidatos com base nestes rankings. O candidato na base do ranking não recebe pontos, o candidato imediatamente acima deste recebe 1 ponto, o próximo recebe 2 pontos e assim por diante até o topo do ranking. As pontuações de todos os rankings são contabilizadas e o candidato com maior pontuação é eleito. De acordo com Lippman variações deste método são encontradas em uso na premiação de diversos esportes como na seleção do MVP(*Most Valuable Player*) da MLB(principal liga de beisebol dos EUA) e no Troféu Heisman de futebol americano universitário.

2.5.1 Exemplo

Neste exemplo serão aproveitados os mesmos rankings apresentados na tabela 3. Por serem 4 candidatos, o primeiro colocado de cada ranking receberá 3 pontos, o segundo 2 pontos, o terceiro 1 e o quarto 0. Assim, o número máximo de pontos que um candidato pode atingir é igual a 300(se ficar na primeira posição em todos os rankings) e o número total de pontos no sistema é 600. O número máximo de pontos que um candidato pode atingir é calculado com:

$$(c-1) \times e$$

Sendo c igual ao número de candidatos e e o número de eleitores. Por sua vez, o número total de pontos no sistema é:

$$\frac{(c-1) \times c}{2} \times e$$

A contagem final dos pontos é apresentada na tabela 7 abaixo.

Tabela 7 – Resultado BC

Candidatos	Total de pontos
A	146
B	181
C	126
D	147

O candidato eleito é o B com 181 pontos ou aproximadamente 60% da pontuação máxima

2.5.2 Vantagens e Desvantagens do BC

Em comparação com os métodos abordados anteriormente, que apenas consideram as primeiras opções de cada eleitor, este método regularmente escolhe candidatos com aceitação mais ampla ao invés de uma preferência direta dos eleitores (Electoral Reform Society, 2017b).

Ele também encoraja uma abordagem estratégica por parte dos partidos, onde seria de seu interesse nomear mais de um candidato para concorrer, pois quanto mais candidatos estão no páreo maior é a pontuação dos primeiros colocados. Logo, a nomeação de candidatos menos populares pode beneficiar os principais candidatos de um partido.

2.5.3 Voto Útil no BC

Existem duas estratégias aplicáveis a esse sistema. A primeira, denominada *compromising*, de maneira similar a outros métodos de votação, se resume em ranquear em primeiro lugar um dos candidatos que estejam liderando as eleições mesmo que este não seja sua preferência real. A segunda estratégia, *burying*, é o oposto, esta se baseia em ranquear em último lugar um dos candidatos líderes mesmo que este não seja o seu candidato de maior desgosto. Ambas estratégias podem ser aplicadas ao mesmo tempo por um eleitor.

2.6 SCORE VOTING SYSTEM

Score Voting ou *Range Voting*, como também é conhecido, é mais um método que tenta trazer maior liberdade de expressão aos eleitores. Assim como o *Borda Count*, este sistema funciona baseado na pontuação dos candidatos, no entanto, ao invés dessa pontuação ser relativa à posição do candidato no ranking de cada eleitor, este por sua vez tem completa liberdade de atribuir qualquer quantidade de pontos aos candidatos dentro

de determinada escala. Um eleitor pode dar o mesmo número de pontos para diferentes candidatos e, após o somatório de todas as pontuações, o candidato com a pontuação mais alta é eleito. Incorporado ao escopo político, o SVS é utilizado no Partido Pirata Alemão bem como em diversas organizações tais como o *Fedora Project* e *Mozilla* e também em variados esportes olímpicos e *reality shows* como o *American Idol*, aponta Hamlin (2015).

2.6.1 Exemplo

Neste exemplo temos, apresentados na tabela 8 abaixo, 4 perfis distintos de eleitores e suas respectivas atribuições de pontos aos 4 candidatos. Cada eleitor pode atribuir qualquer pontuação a um candidato dentro da escala de -10 a 10. O resultado da eleição, após a soma de todos os pontos por candidato, se encontra na tabela 9 onde o candidato A é eleito com um total de 605 pontos.

Tabela 8 – Exemplo SVS

Perfil	A	B	C	D	N° de eleitores
1	10	-1	5	2	36
2	6	9	-3	4	25
3	2	0	8	4	22
4	3	3	3	10	17

Tabela 9 – Resultado SVS

Candidatos	Total de pontos
A	605
B	240
C	332
D	430

2.6.2 Vantagens e Desvantagens do SVS

A grande vantagem do SVS é como ele proporciona aos eleitores expressar de maneira muito próxima, potencialmente exata, suas impressões políticas de cada candidato. Ele é ainda mais expressivo que o AVS. Dessa maneira o *spoiler effect* também se torna nulo ao passo que toda intenção de voto é precisamente representada neste sistema.

Assim como outros métodos eleitorais, este sofre igualmente dos males produzidos pelo voto tático. Sua principal desvantagem é discutida na seção abaixo.

2.6.3 Voto Útil no SVS

Da mesma maneira que no *Approval Voting*, através do *bullet voting*, o SVS também pode ser convertido em uma eleição sob o método FPTP em um cenário com um grande

volume de eleitores maliciosos. O voto útil sob este sistema se manifesta pontuando-se apenas um candidato com uma nota positiva e todos os outros com as notas mais baixas da escala. Se todos os eleitores se aproveitarem dessa abordagem não existirá distinção efetiva entre uma eleição sob SVS e FPTP. Essencialmente cada eleitor está votando em apenas um candidato. Essa é a principal desvantagem do SVS e ela consegue eliminar sua principal vantagem: dar ao eleitor um poder maior de expressão.

2.7 BLOC VOTE

O *Bloc Vote* é um método utilizado para a eleição de candidatos a cargos no governo com mais de uma vacância. Pode ser considerado um meio-termo entre o FPTP e o AVS, pois os eleitores podem votar em mais de um candidato, porém são limitados pelo número de vagas. Em uma eleição com duas vagas, por exemplo, cada eleitor selecionará dois candidatos ou menos. Sensatamente, o candidato mais votado é eleito. Ele possui algumas utilizações em eleições locais em algumas partes da Inglaterra (Electoral Reform Society, 2017a).

2.7.1 Exemplo

Neste exemplo dois candidatos serão eleitos, logo cada eleitor votará em, no máximo, dois candidatos. A distribuição de votos pode ser conferida na tabela 10 e o resultado da contagem de votos na tabela 11 abaixo. Os candidatos eleitos são o C, totalizando 78% do máximo de votos possíveis para um candidato, e o A com 43%.

Tabela 10 – Exemplo BV

Aprovações	N° de eleitores
A,C	20
A,B	16
B,C	19
B,D	6
C,A	7
C,D	15
D,C	17

Tabela 11 – Resultado BV

Candidatos	Total de votos
A	43
B	41
C	78
D	38

2.7.2 Voto Útil no BV

Neste sistema é possível que um eleitor evite votar em sua primeira opção quando este não tem chances de vitória e se fazendo isto ele diminuirá as chances de algum outro candidato de seu desgosto ganhar. Porém, o sistema previne um outro tipo de voto útil. Em eleições com mais de uma vaga, onde os eleitores só podem votar em um candidato, se um eleitor sabe que sua primeira opção também é a primeira opção da maior parte da população e, portanto, muito provavelmente será um dos candidatos eleitos, é possível que ele arrisque votar em uma segunda opção sua na esperança de, a longo prazo, eleger suas duas principais opções de voto. Com os eleitores podendo votar no número exato de vagas disponíveis esse tipo de pensamento tático perde o sentido.

3 O SIMULADOR

Para este trabalho, foi implementado um simulador eleitoral na forma de uma aplicação web para que a interação do usuário fosse facilitada. Veremos a seguir como se dá sua utilização, de forma geral.

Figura 2 – Captura de tela 1

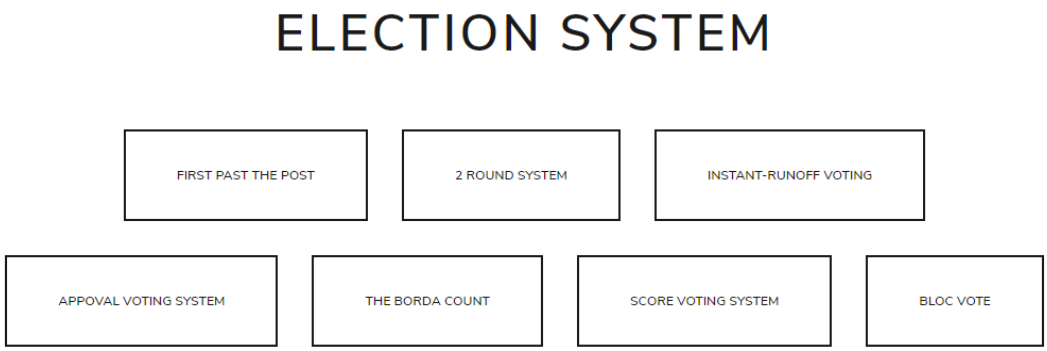
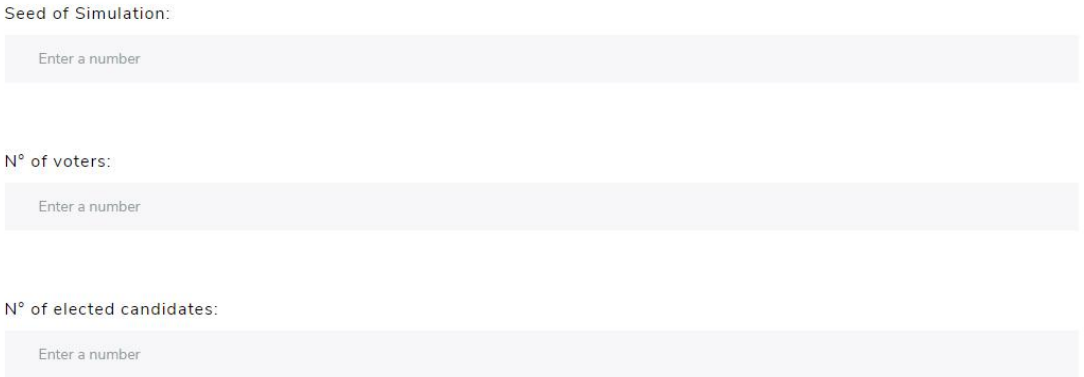


Figura 3 – Captura de tela 2

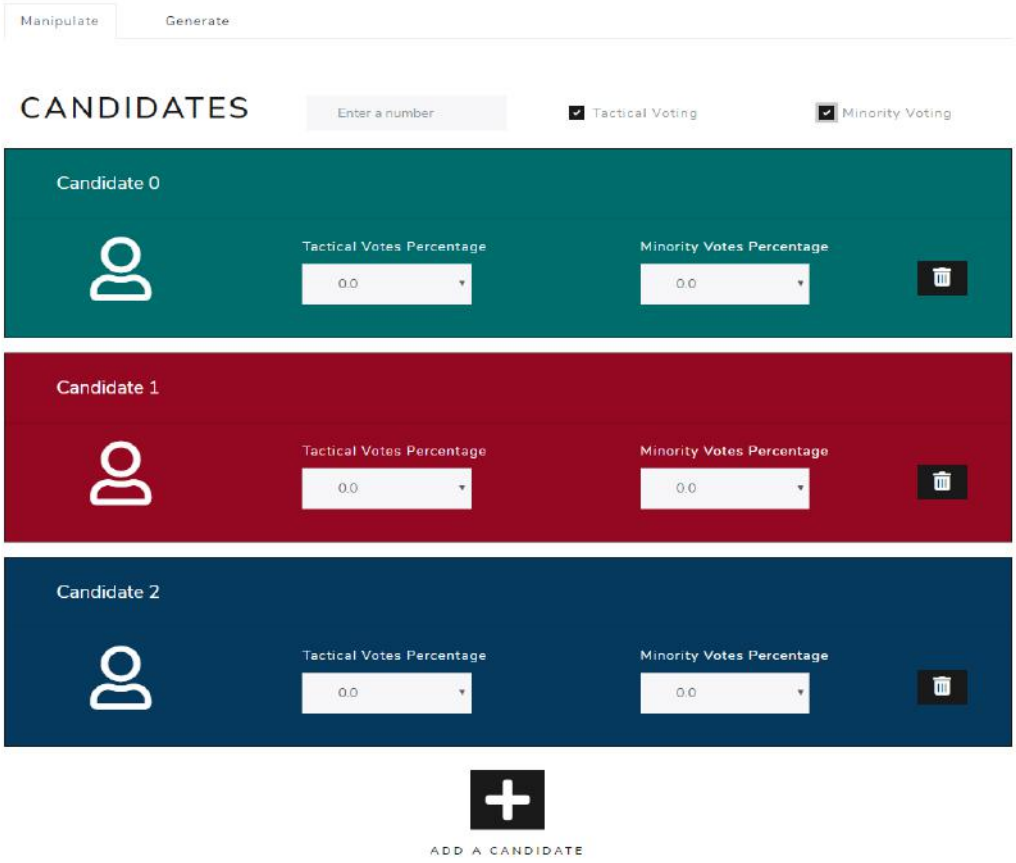


A primeira opção encontrada pelo usuário é a escolha de quais sistemas de votação serão simulados (Figura 2). Todos eles podem ser selecionados para rodar simultaneamente. Porém não é permitido rodar uma simulação do *Bloc Vote* para uma eleição onde apenas um candidato é eleito; além disso, não existe implementação dos métodos TRS e IRV para

eleições com mais de um candidato eleito já que eles se comportariam exatamente como no FPTP.

A maioria das linguagens de programação, senão todas, possuem um gerador de números pseudoaleatórios que fornece a possibilidade de definir uma *seed* para o gerador. Esta *seed* funciona como um ponto de partida para a geração dos número e, portanto, toda simulação que rodar com a mesma *seed* sempre terá os mesmos resultados. Se nenhum valor for fornecido pelo usuário o tempo atual do sistema é utilizado. As opções subsequentes são a escolha do número de eleitores gerados e o número de vagas (Figura 3). Se nenhum valor for fornecido o número de eleitores padrão utilizado é 1000 e o número de vagas igual a 1.

Figura 4 – Captura de tela 3



Este simulador possui dois modos para a criação dos rankings dos eleitores. No modo *Manipulate* (Figura 4), primeiramente se define o número de candidatos através do campo de entrada ao lado do título *Candidates* ou, alternativamente, pode-se adicionar candidatos um a um através do sinal de mais na parte inferior da página. Para cada candidato é possível definir a porcentagem de seus eleitores que optará pelo "voto tático" e pelo "voto de minoria". Também é possível alterar o nome dos candidatos se for do interesse do usuário.

Figura 5 – Captura de tela 4

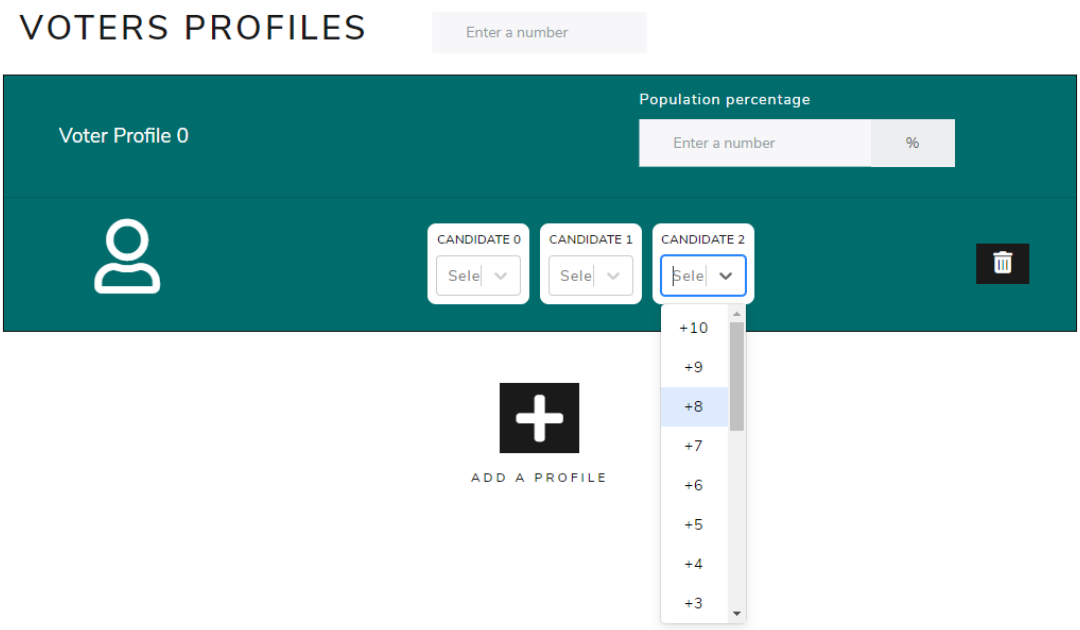


Figura 6 – Captura de tela 5

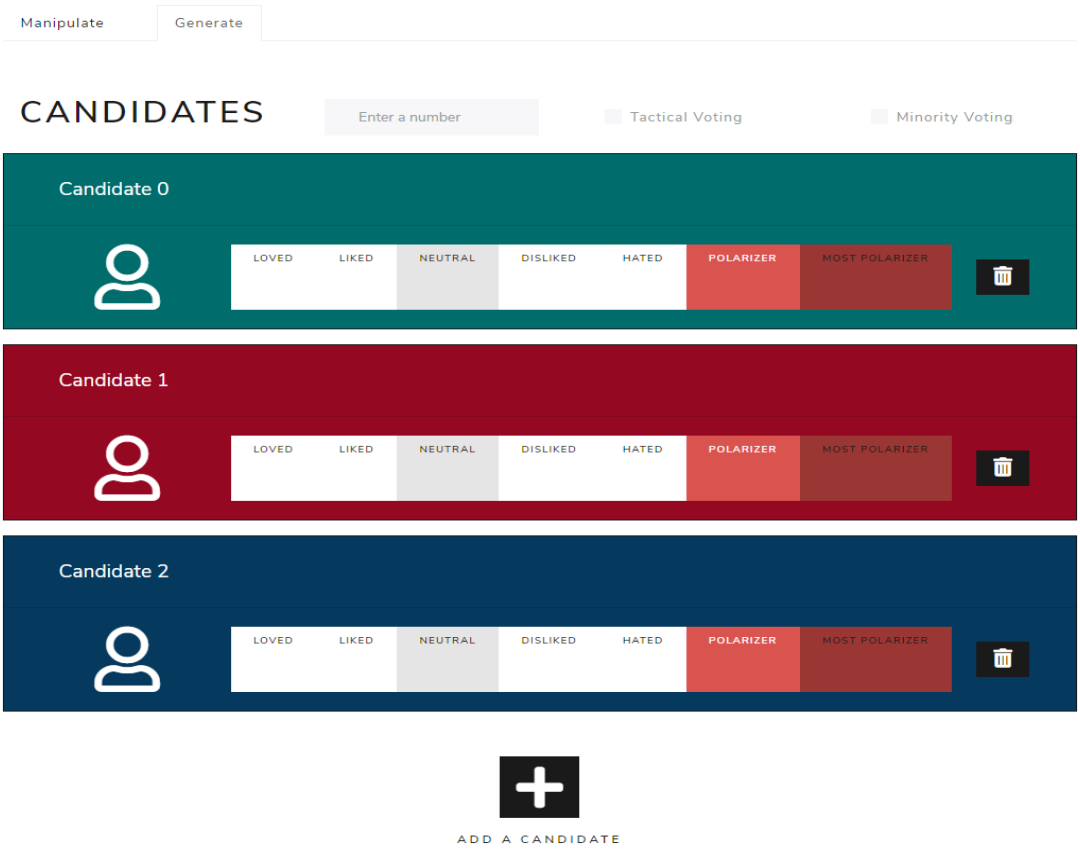
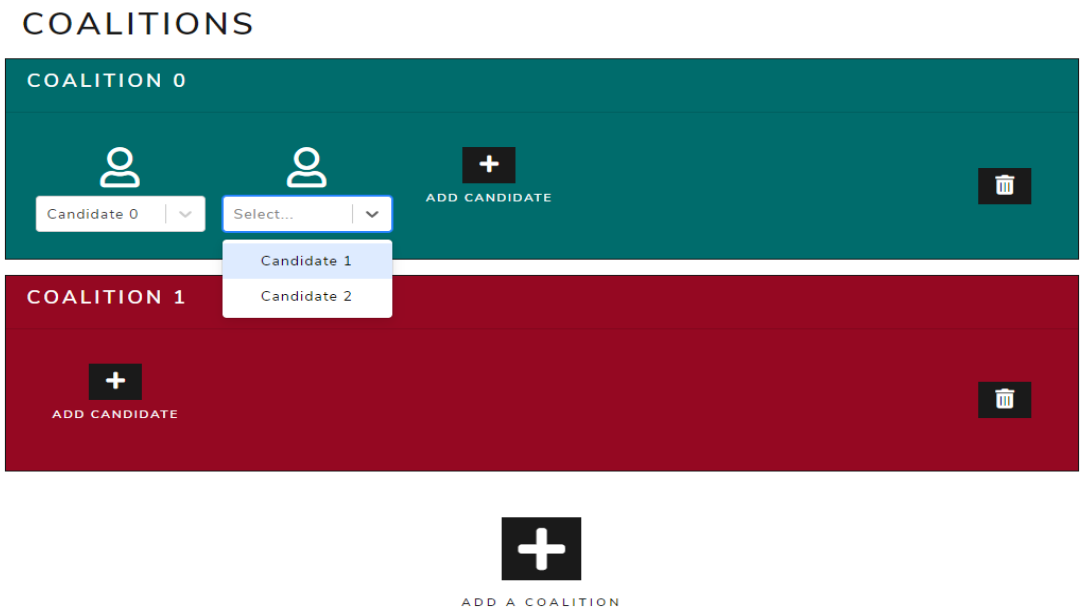


Figura 7 – Captura de tela 6



Após a criação dos candidatos o usuário pode criar perfis de eleitores que definirão de maneira exata como partes da população irá votar (Figura 5). Em cada perfil se define o valor em porcentos da população votante que o adotará e a nota de cada candidato. Também é possível nomear cada perfil livremente.

No modo *Generate* (Figura 6) é onde o gerador de números pseudoaleatórios é usado. Nesse modo, ao invés do usuário criar perfis de eleitores, ele define qual será a distribuição adotada para a geração de notas de cada candidato. Essas distribuições serão mais bem detalhadas no Capítulo 5. Da mesma maneira que no modo *Manipulate*, nesse modo também é possível definir os valores para os dois tipos de voto útil, os "votos táticos" e os "votos de minoria". No primeiro estão incluídas as estratégias de mudança de voto específicas de cada sistema, como o *bullet voting* no AVS e SVS ou quando a preferência de um eleitor claramente não possui chances de vitória e ele muda seu voto para um dos candidatos que esteja na liderança, o que ocorre no FPTP e TRS. O "voto de minoria" se apresenta apenas no FPTP em eleições onde mais de um candidato é eleito. Se a preferência de um eleitor claramente lidera a eleição por ser também a preferência da maior parte da população, esse eleitor pode mudar o seu voto para uma segunda opção na esperança de eleger dois de seus candidatos favoritos.

Por fim, em ambos os modos o usuário tem a opção de definir coalizões entre os candidatos se o sistema TRS estiver habilitado, pois as coalizões apenas se manifestam nesse sistema. Essas coalizões alterarão as notas dos candidatos dependendo dos outros candidatos pertencentes a elas (Figura 7).

A aplicação pode ser encontrada na página: http://electionsim.pythonanywhere.com

4 CENÁRIOS PERTINENTES

Para todos os cenários deste capítulo será usada a *seed* do simulador igual a 100 e a população de eleitores igual a 1000 para que exista consistência na análise e também para que possa haver replicação dos resultados se necessário.

4.1 CENÁRIO 1: VOTO TÁTICO

Neste exemplo teremos quatro candidatos, Ana, Beto, Carla e Diego. Ana e Beto serão gerados a partir da distribuição *Neutral* (Figura 47), Carla da distribuição *Disliked* (Figura 51) e Diego da *Hated* (Figura 55). Para a simulação com a *seed* 100 o melhor candidato a ser eleito, ou seja, aquele que maximiza a média das notas dos eleitores é o Beto com uma média de 0.073. Para cada sistema serão comparados os resultados sem "voto tático" aos com 20% de "voto tático" a favor da Ana. Abaixo se encontram os resultados para o sistema FPTP.

Figura 8 – FPTP - Cenário 1 sem voto tático

Figura 9 – TRS segundo turno - Cenário 1 sem voto tático

Figura 10 – IRV primeiro turno - Cenário 1

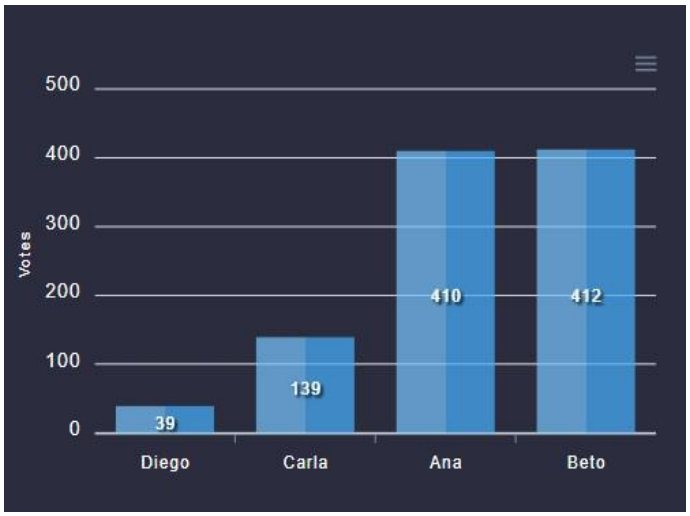


Figura 11 – IRV segundo turno - Cenário 1

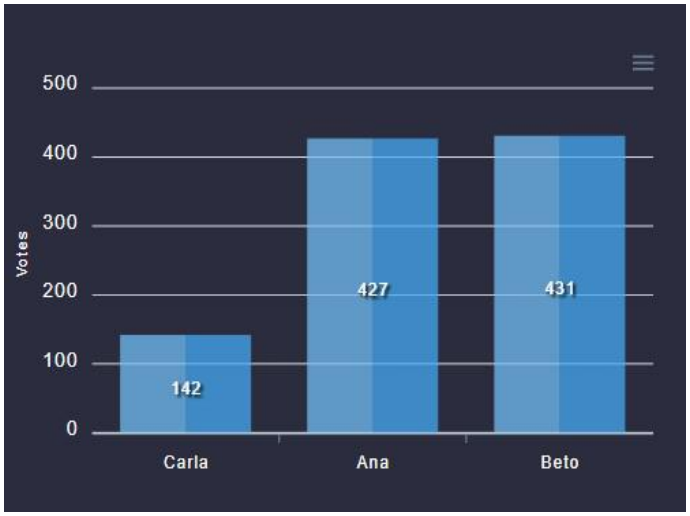


Figura 12 – IRV terceiro turno - Cenário 1

Figura 13 – AVS - Cenário 1 sem voto tático

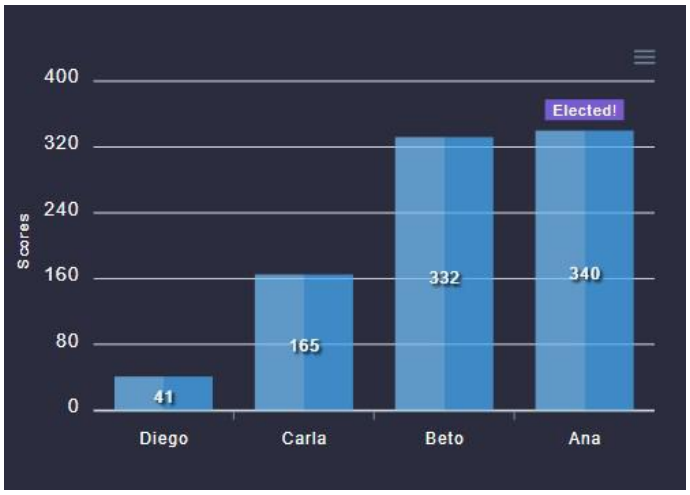


Figura 14 – BC - Cenário 1 sem voto tático

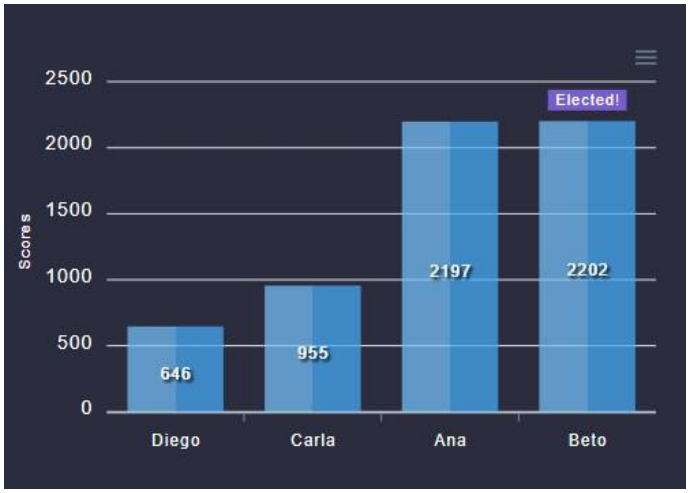


Figura 15 – SVS - Cenário 1 sem voto tático

Nesse cenário, sem "voto tático", o único sistema que não elege o Beto é o AVS. Todos os outros elegem o melhor candidato. Ou seja, apesar de Beto ter uma melhor média das notas dos eleitores, existem mais eleitores que ranqueiam Ana com uma nota acima de 0. No entanto, apenas 20% dos eleitores de Diego e Carla que preferem a Ana ao invés do Beto são o suficiente para fazer Beto não ser eleito se eles decidirem abandonar sua primeira opção e votar na Ana. Considerando a vantagem que os "votos táticos" proporcionam à Ana, esta é eleita em todos os sistemas menos o TRS e o IRV.

Figura 16 – FPTP - Cenário 1 - 20% de voto tático na Ana

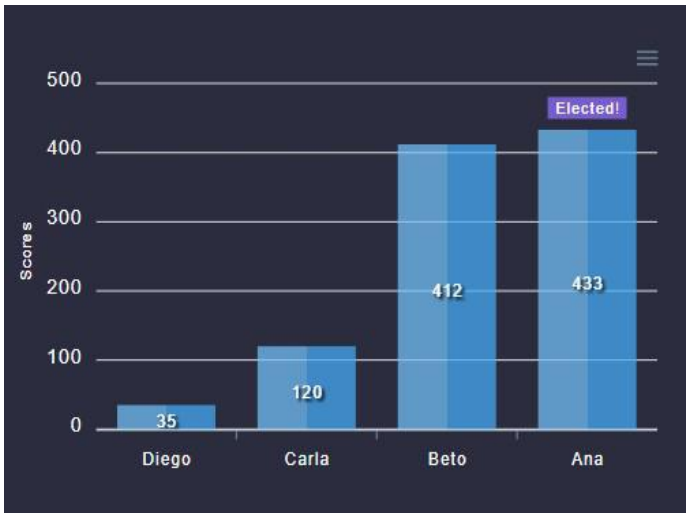


Figura 17 – TRS segundo turno - Cenário 1 20% de voto tático na Ana

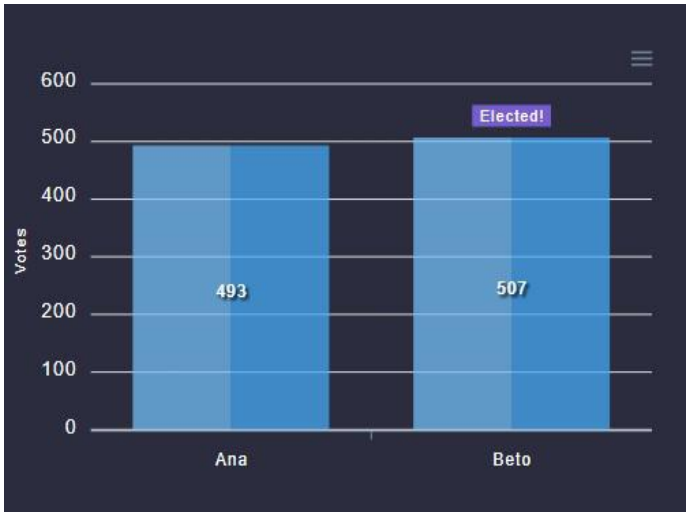


Figura 18 – IRV primeiro turno - Cenário 1 20% de voto tático na Ana

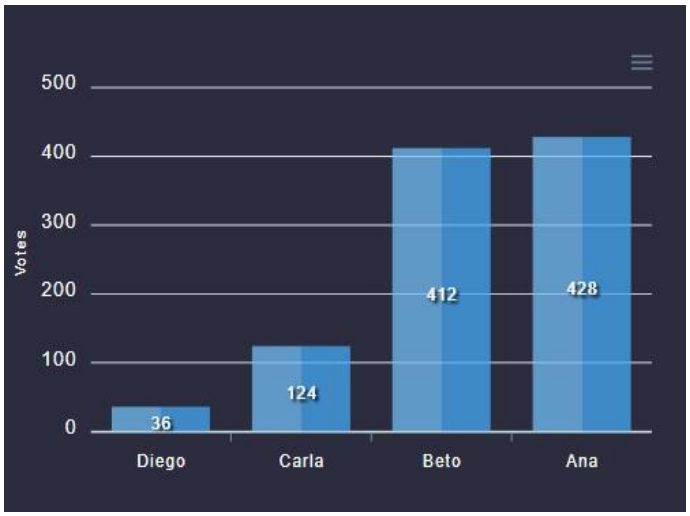


Figura 19 – IRV segundo turno - Cenário 1 20% de voto tático na Ana

Figura 20 – IRV terceiro turno - Cenário 1 20% de voto tático na Ana

Figura 21 – AVS - Cenário 1 20% de voto tático na Ana

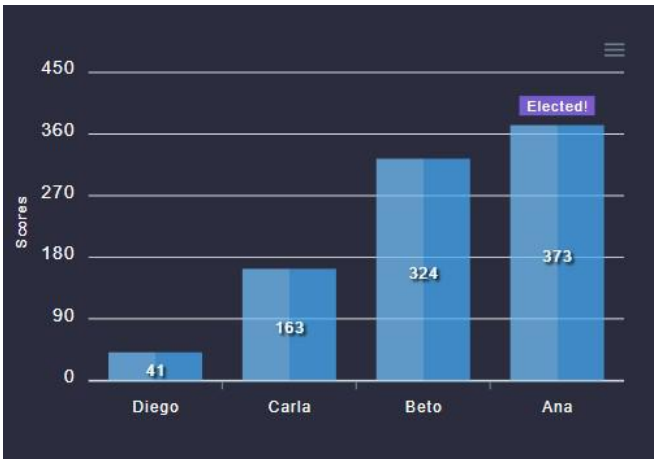


Figura 22 – BC - Cenário 1 20% de voto tático na Ana

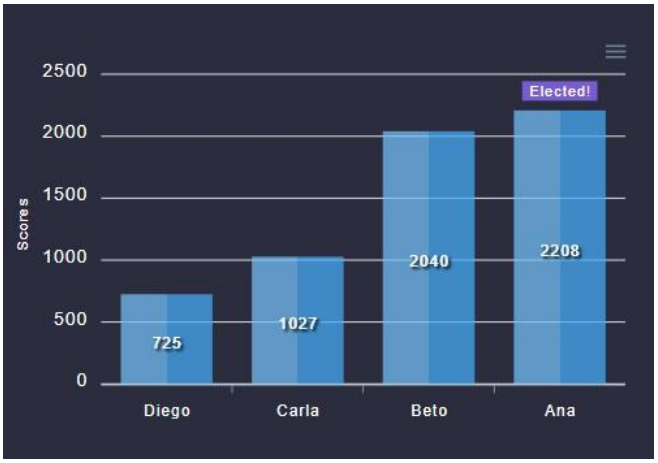
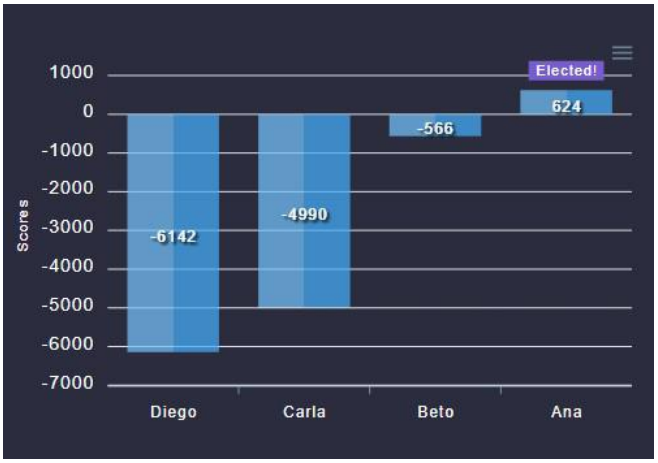
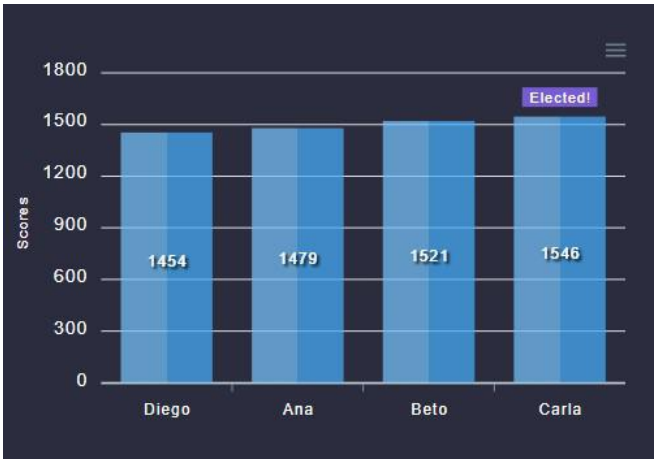


Figura 23 – SVS - Cenário 1 20% de voto tático na Ana



É claro que se os eleitores que preferem o Beto também votassem de maneira estratégica a situação se equilibraria e haveria o mesmo nível de concorrência entre os dois candidatos que ocorre quando não há "voto tático". Porém, no *Borda Count*, esse equilíbrio pode não acontecer e as estratégias de *compromising* e *burying* podem acabar saindo pela culatra se uma porcentagem elevada dos eleitores as adotarem. Abaixo encontram-se os resultados para esse sistema quando 100% dos eleitores de Carla e Diego se comportam maliciosamente.

Figura 24 – BC - Cenário 1 100% de voto tático na Ana e Beto



Por causa do uso demasiado da estratégia *burying*, Ana e Beto perdem muitos pontos e isso abre espaço para uma vitória da Carla. No entanto, para porcentagens menores que 100% o Beto ainda é o vencedor.

Essas duas estratégias também foram implementadas nesse simulador para o sistema IRV e é possível ver como ele é resistente a elas. Para que seja possível mudar os resultados de uma eleição sob o IRV é necessário mudar a ordem de eliminação dos candidatos para que o seu candidato preferido acabe enfrentando um candidato mais fraco nas rodadas finais. Isso não ocorre com *compromising* e *burying*. No próximo cenário será analisada como é possível modificar os resultados de uma eleição sob o IRV.

Esses resultados podem ser replicados acessando os links:

Sem "votos táticos": <http://electionsim.pythonanywhere.com/#/direct_res/1111110/-/-/[0,0,2,1]/[%22Ana%22,%22Beto%22,%20%22Carla%22,%22Diego%22]/-/-/-/-/-/-/100>

Com 20% de "voto tático": <http://electionsim.pythonanywhere.com/#/direct_res/1111110/-/-/[0,0,2,1]/[%22Ana%22,%20%22Beto%22,%22Carla%22,%22Diego%22]/1/-/[0.2,0,0,0]/-/-/-/100>

Com 100% de "voto tático"na Ana e Beto: <http://electionsim.pythonanywhere.com/#/direct_res/0000100/-/-/[0,0,2,1]/[%22Ana%22,%20%22Beto%22,%20%22Carla%22,%22Diego%22]/1/-/[1.0,1.0,0,0]/-/-/-/100>

4.2 CENÁRIO 2: VOTO TÁTICO NO IRV

Neste cenário teremos Ana, Carla e Beto concorrendo e três perfis distintos que apoiam cada candidato. As distribuições de notas dos perfis se encontram na tabela abaixo:

Tabela 12 – Distribuição de notas - Cenário 2

Perfil	Nota da Ana	Nota do Beto	Nota da Carla	Porcentagem de eleitores
1	10	5	0	42%
2	0	10	5	30%
3	5	0	10	28%

Com essas distribuições, Carla é eliminada na primeira rodada e seus 28% de votos são transferidos para a segunda opção de seus eleitores, Ana, que é eleita com 70% dos votos e uma média de 5.6.

Figura 25 – IRV primeiro turno - Cenário 2

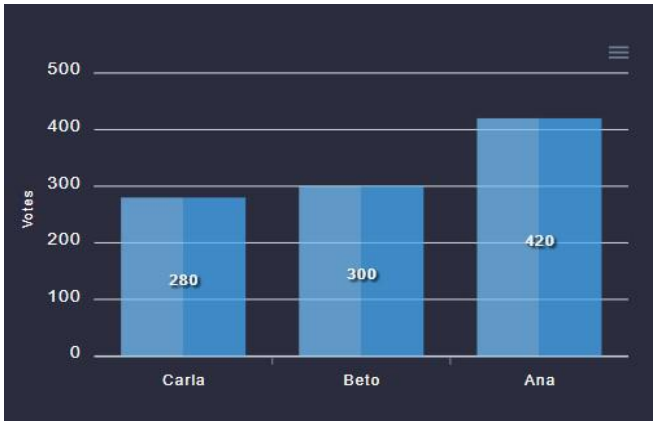
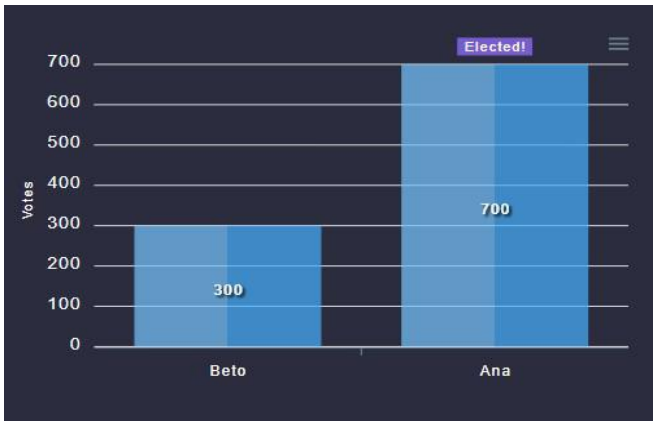


Figura 26 – IRV segundo turno - Cenário 2



Cientes desse provável futuro cenário onde Ana é eleita, uma pequena parte dos eleitores de Beto estrategicamente o abandona e decide apoiar Carla na esperança de leva-la ao segundo turno contra Ana. As novas distribuições de notas se encontram na tabela abaixo com essa porção de eleitores representada pelo perfil 4:

Tabela 13 – Distribuição de notas com voto tático - Cenário 2

Perfil	Nota da Ana	Nota do Beto	Nota da Carla	Porcentagem de eleitores
1	10	5	0	42%
2	0	10	5	28%
3	5	0	10	28%
4	0	8	10	2%

Essa mudança estratégica de voto altera a ordem de eliminações do sistema com Beto sendo o primeiro a ser eliminado. Com isso, seus votos são naturalmente transferidos para Carla que derrota Ana no segundo turno com 58% dos votos.

Figura 27 – IRV primeiro turno com voto tático - Cenário 2

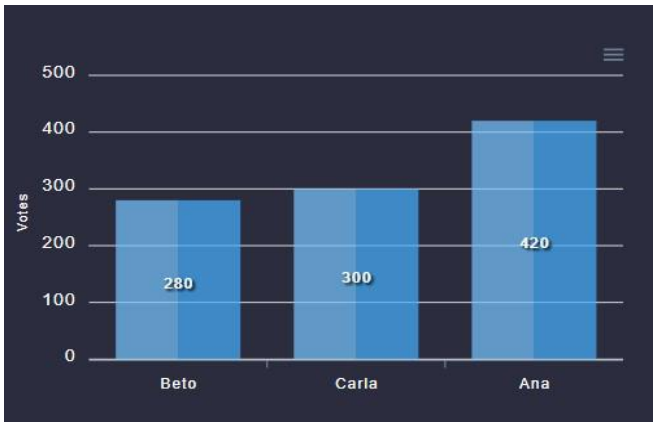
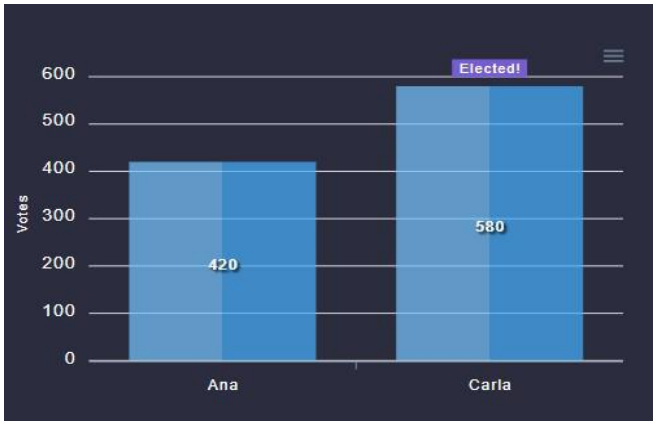


Figura 28 – IRV segundo turno com voto tático - Cenário 2



Percebendo que a eleição de Beto não era provável, seus eleitores conseguiram ao menos impedir a eleição do pior candidato para eles. Mas para isso, foi necessário que eles tivessem a informação de como os outros eleitores votariam. Uma aproximação dessas informações poderia ser obtida em um cenário real com poucos candidatos através de pesquisas eleitorais, porém se o número de candidatos é significativo, uma noção completa das distribuições dos rankings dos eleitores se torna muito menos viável e dificulta a formulação de estratégias para alterar a ordem em que os candidatos são eliminados.

Esses resultados podem ser replicados acessando os links:
Sem "voto tático":
<http://electionsim.pythonanywhere.com/#/direct_res/0010000/-/-/-/[%22Ana%22,%22Beto%22,%22Carla%22]/-/-/-/-/-/42,Voter%20Profile%200,10%205%200,30,Voter%20Profile%201,0%2010%205,28,Voter%20Profile%202,5%200%2010/100>

Com "voto tático":
<http://electionsim.pythonanywhere.com/#/direct_res/0010000/-/-/-/[%22Ana%22,%22Beto%22,%22Carla%22]/-/-/-/-/-/42,Voter%20Profile%200,10%205%200,28,Voter%20Profile%201,0%2010%205,28,Voter%20Profile%202,5%200%2010,2,Voter%20Profile%203,0%208%2010/100>

4.3 CENÁRIO 3: NÃO-MONOTONICIDADE

O interessante dos sistemas IRV e TRS é que os resultados do cenário anterior poderiam ter sido obtidos, ao invés do "voto tático", pela sua não-monotonicidade. Seria possível que a Carla derrotasse a Ana no segundo turno através do próprio aumento de popularidade da Ana. Assumindo que essa popularização ocorresse entre os candidatos do perfil 2, onde 3% passasse a apoiar a Ana ao invés do Beto, as novas distribuições seguiriam a tabela abaixo:

Tabela 14 – Distribuição de notas após popularização de Ana - Cenário 3

Perfil	Nota da Ana	Nota do Beto	Nota da Carla	Porcentagem de eleitores
1	10	5	0	45%
2	0	10	5	27%
3	5	0	10	28%

Dessa maneira, Beto, agora com uma porcentagem dos eleitores menor do que a de Carla, será eliminado na primeira rodada e, como seus eleitores têm a Carla como segunda opção, ela receberá todos os seus votos e será eleita na segunda rodada. As rodadas do IRV podem ser vistas nas figuras 29 e 30.

Figura 29 – IRV primeiro turno após popularização de Ana - Cenário 3

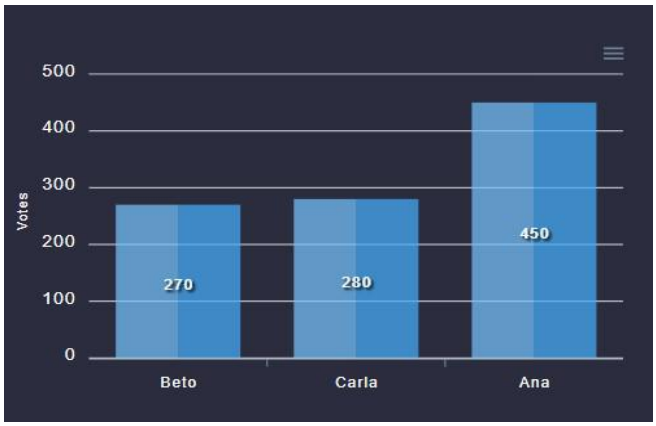


Figura 30 – IRV segundo turno após popularização de Ana - Cenário 3

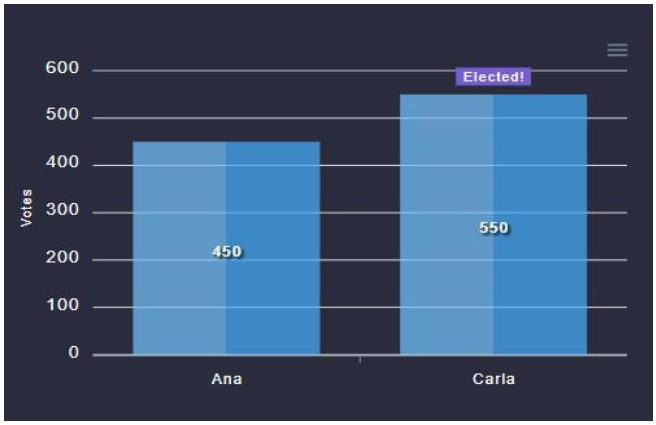


Tabela 15 – Resultados do Cenário 3

	Candidato eleito	Média das notas
Antes da popularização de Ana	Ana	5.6
Após popularização de Ana	Carla	4.15

Os efeitos dessa característica se refletem na satisfação média da população. A Ana é claramente a melhor candidata e mesmo com o seu ganho de popularidade (ou devido a ele) a Carla é eleita.

Esses resultados podem ser replicados acessando os links:

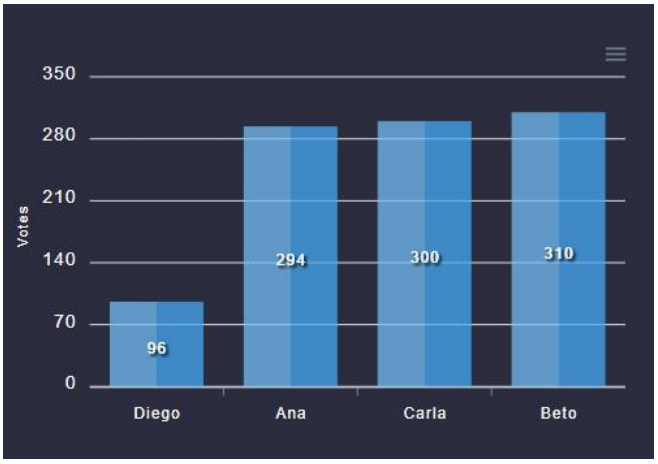
Antes da popularização de Ana: <http://electionsim.pythonanywhere.com/#/direct_res/0010000/-/-/-/[%22Ana%22,%22Beto%22,%22Carla%22]/-/-/-/-/-/42,Voter%20Profile%200,10%205%200,30,Voter%20Profile%201,0%2010%205,28,Voter%20Profile%202,5%200%2010/100>

Após popularização de Ana: <http://electionsim.pythonanywhere.com/#/direct_res/0010000/-/-/-/[%22Ana%22,%22Beto%22,%22Carla%22]/-/-/-/-/-/45,Voter%20Profile%200,10%205%200,27,Voter%20Profile%201,0%2010%205,28,Voter%20Profile%202,5%200%2010/100>

4.4 CENÁRIO 4: COALIZÕES

O TRS, por ser organizado em dois turnos separados, normalmente a semanas de distância um do outro, acaba abrindo brecha à mudança de opinião por parte dos eleitores e o desenvolvimento de novas estratégias por parte dos candidatos, tais como a coalizão com candidatos já eliminados. Com isso em mente, nesse cenário teremos novamente os quatro candidatos, Ana, Beto, Carla e Diego. Os três primeiros serão gerados da distribuição *Neutral* e Diego da distribuição *Disliked*.

Figura 31 – TRS turno 1 - Cenário 4



No TRS normal, Ana, Beto e Carla possuem, com essas distribuições, quantidades de votos muito próximas com Beto e Carla indo para o segundo turno. Carla está 10 votos atrás de Beto e decide aproveitar o espaço de tempo entre os dois turnos para tentar agregar mais votos. Com isso, Carla se aproxima de Diego e juntos formam uma coalizão. No entanto, com sua baixa popularidade, Diego acaba ferindo a reputação de Carla e torna o segundo turno muito mais fácil para Beto (Figura 33). Se Carla tivesse buscado o suporte de Ana, por outro lado, ela se veria vitoriosa no segundo turno e com uma ampla vantagem sobre o Beto (Figura 34). Ou até mesmo sem buscar uma coalizão, mesmo sendo uma corrida apertada, Carla teria naturalmente sido eleita (Figura 32).

Esses resultados podem ser replicados acessando os links:

Sem coalizões: <http://electionsim.pythonanywhere.com/#/direct_res/0100000/-/-/[0,0,0,2]/[%22Ana%22,%20%22Beto%22,%22Carla%22,%22Diego%22]/-/-/-/-/-/-/100>

Figura 32 – TRS turno 2 - Cenário 4

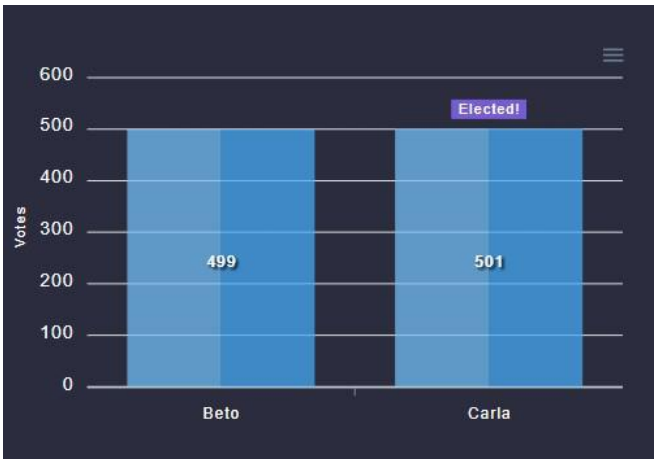
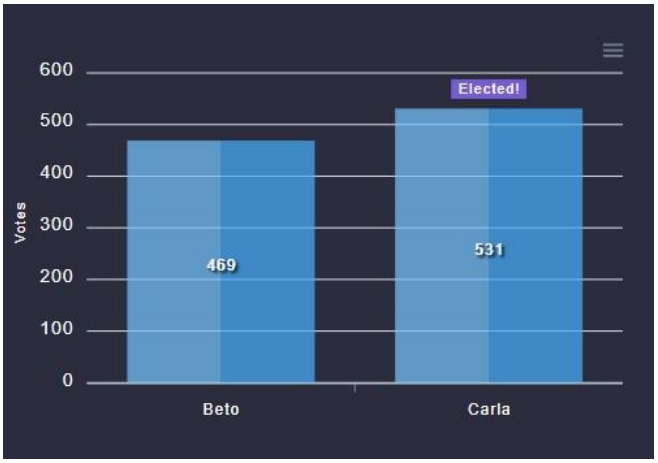


Figura 33 – TRS coalizão de Carla e Diego - Cenário 4

Figura 34 – TRS coalizão de Carla e Ana - Cenário 4



Coalizão Carla e Diego <http://electionsim.pythonanywhere.com/#/direct_res/0100000/-/-/[0,0,0,2]/[%22Ana%22,%22Beto%22,%22Carla%22,%22Diego%22]/-/-/-/-/2%203,Carla%20Diego/-/100>

Coalizão Carla e Ana <http://electionsim.pythonanywhere.com/#/direct_res/0100000/-/-/[0,0,0,2]/[%22Ana%22,%22Beto%22,%22Carla%22,%22Diego%22]/-/-/-/-/2%200,Carla%20Ana/-/100>

4.5 CENÁRIO 5: DILEMA DO PRISIONEIRO

Neste cenário temos uma eleição de uma vaga com três candidatos concorrentes e três perfis distintos de eleitores que seguirão as seguintes distribuições:

Tabela 16 – Distribuição de notas - Cenário 5

Perfil	Nota da Ana	Nota do Beto	Nota da Carla	Porcentagem de eleitores
1	10	-10	5	36%
2	-10	10	5	34%
3	-10	-10	5	30%

Esse contexto nos remete ao notável "dilema do prisioneiro". Nele dois prisioneiros cúmplices são interrogados individualmente sem poder se comunicar um com o outro e cada um enfrenta o dilema de delatar ou não o seu parceiro. Se os dois prisioneiros decidem se manter calados sobre o crime, ambos são sentenciados a um ano de prisão. Se apenas um deles dedurar o outro, o traidor sai livre enquanto seu parceiro é condenado a três anos. Se ambos dedurarem um ao outro, eles são condenados juntos a dois anos de prisão. Esse dilema é uma ideia clássica presente no estudo da teoria dos jogos sendo originalmente elaborada por Merrill Flood e Melvin Dresher em 1950 e mais tarde sendo nomeada de "dilema dos prisioneiros" por Albert W. Tucker (POUNDSTONE, 1992).

O que esse dilema nos mostra é a tentação que o indivíduo detém se sua racionalização estiver voltada para seus próprios interesses e somente eles, em oposição a pensar no bem do grupo como um todo. Neste cenário encontramos dois grupos de eleitores expressivamente opostos, os perfis 1 e 2. Se qualquer uma das preferências desses perfis fosse eleita, isso significaria uma alta rejeição de maior parte da população. No entanto, eles possuem a segunda opção em comum, a Carla, que por sua vez é a preferência do terceiro perfil. Portanto, a eleição da Carla seria logicamente o resultado ótimo.

Mesmo assim a Ana é eleita nos dois sistemas acima, o que não parece justo, pois pela tabela 16 é evidente que apenas 36% da população a aprova enquanto que os outros 64% a reprovam ao máximo. O sistema IRV se comporta de maneira idêntica ao TRS

Figura 35 – FPTP - Cenário 5

Figura 36 – TRS segundo turno - Cenário 5

nesta situação. Com esse resultado a média das notas é -2.8. No entanto, os três sistemas seguintes nos apresentam resultados diferentes.

Tabela 17 – Resultados do Cenário 5

Sistema	Candidato eleito	Média das notas
FPTP	Ana	-2.8
TRS	Ana	-2.8
IRV	Ana	-2.8
AVS	Carla	5.0
BC	Carla	5.0
SVS	Carla	5.0

Figura 37 – AVS - Cenário 5

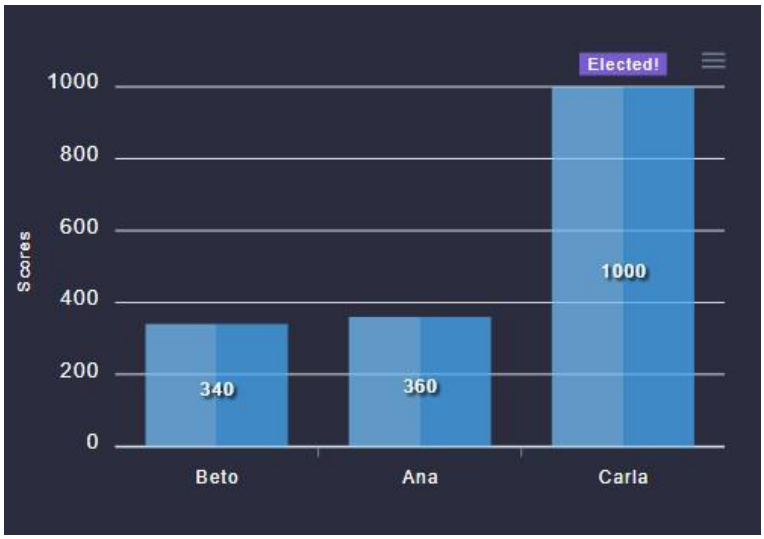
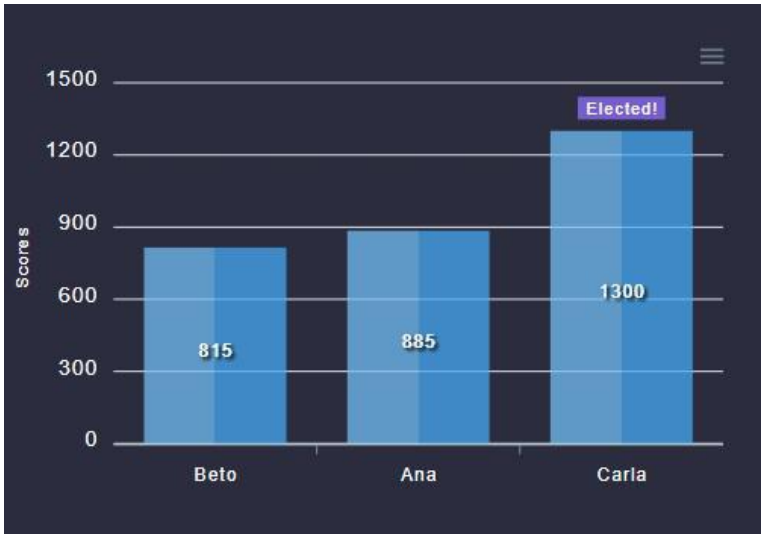


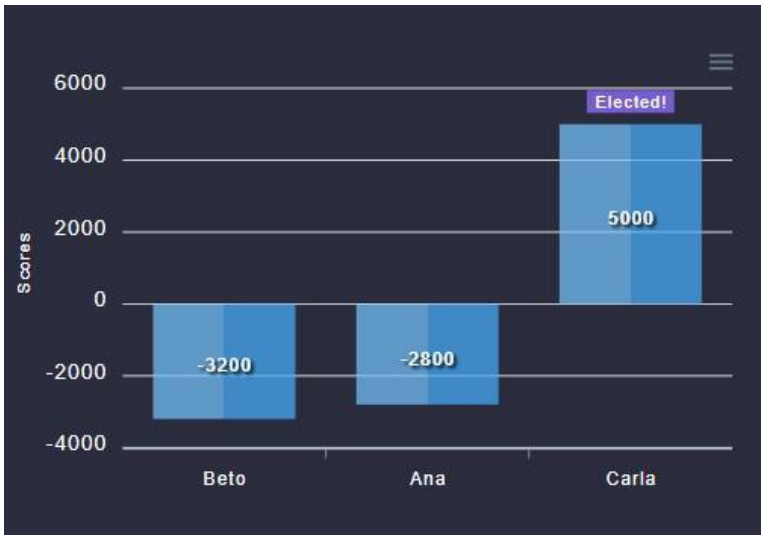
Figura 38 – BC - Cenário 5



Tanto o AVS quanto o *Borda Count* e o SVS são sistemas que consideram todas as opiniões de cada eleitor ao mesmo tempo e por isso são capazes de eleger o melhor candidato nesta situação, diferentemente do TRS que não absorve completamente todo sentimento do eleitor por todos os candidatos e o IRV que, apesar de ser bem expressivo, quebra esse processo em inúmeras rodadas, o que permite a perda de informação ao longo delas.

Esses resultados podem ser replicados acessando o link:
<http://electionsim.pythonanywhere.com/#/direct_res/1111110/-/-/-/[%22Ana%22,%20%22Beto%22,%20%22Carla%22]/-/-/-/-/-/36,Voter%20Profile%200,10%20-10%205,34,Voter%20Profile%201,-10%2010%205,30,Voter%20Profile%202,-10%20-10%205/100>

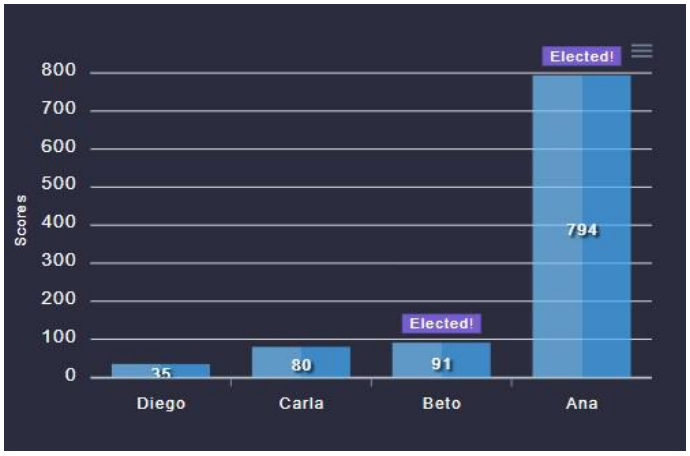
Figura 39 – SVS - Cenário 5



4.6 CENÁRIO 6: VOTO DE MINORIA

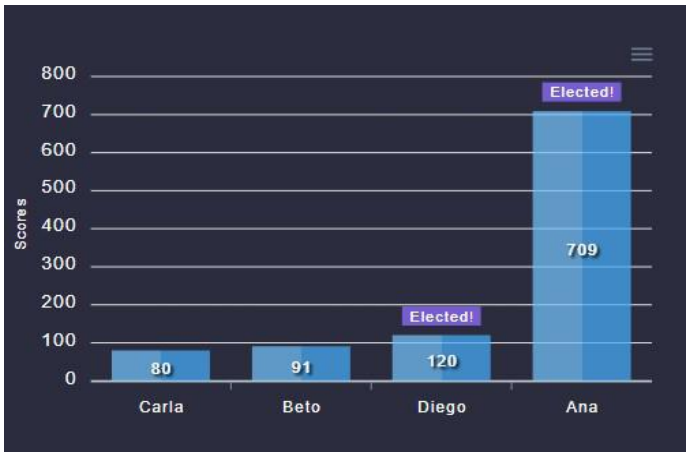
Neste cenário temos uma eleição com quatro candidatos e duas vagas. A Ana será a preferência de uma extensa parte da população (distribuição *Loved*, Figura 53), Beto e Carla serão candidatos razoavelmente populares (distribuição *Neutral*, Figura 47) e Diego será a preferência de apenas um pequeno grupo específico e rejeitado pela maioria (distribuição *Disliked*, Figura 51). É intuitivo esperar que os dois candidatos eleitos serão a Ana e ou o Beto ou a Carla, e isso é exatamente o que ocorre na figura 40, no sistema FPTP sem "voto de minoria".

Figura 40 – FPTP - Cenário 6 sem voto de minoria



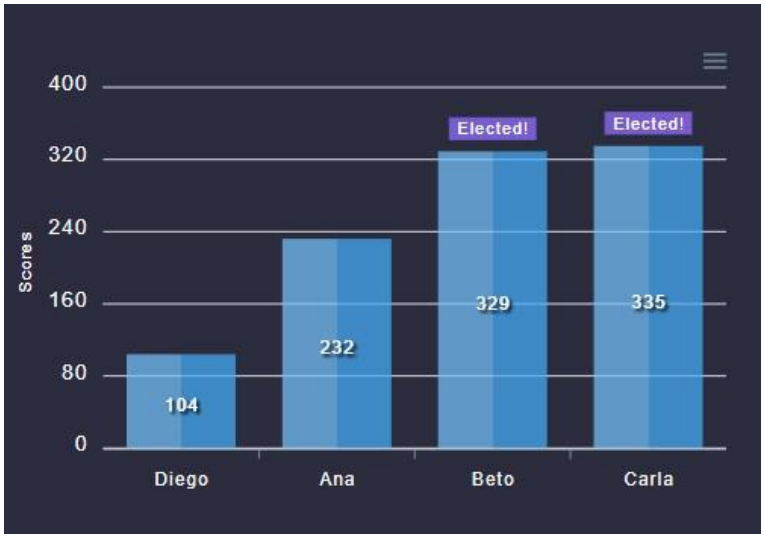
É evidente a discrepância da quantidade de votos da Ana para os outros candidatos e apesar de Beto e Carla serem mais populares que Diego, os três não se encontram muito afastados. Porém, se a pequena fração de candidatos que possuem a Ana como

Figura 41 – FPTP - Cenário 6 100% voto de minoria para Diego



preferência e Diego como segunda opção, optarem pelo "voto de minoria" a situação muda completamente (Figura 41). Assim, apesar de extensamente rejeitado, Diego conquista a segunda vaga e a média das notas despenca de 2.929 para 0.505. No entanto, ainda existe um cenário pior. Se os outros grupos de eleitores pensarem da mesma forma e evitarem votar na Ana, achando que sua eleição já está assegurada, apenas 80% de "voto de minoria" para esses grupos já é o suficiente para eliminá-la das eleições (Figura 42). Com a Ana eliminada, a média das notas cai ainda mais para 0.058.

Figura 42 – FPTP - Cenário 6 80% voto de minoria em Beto, Carla e Diego



O sistema *Bloc Vote* é uma das soluções para o "voto de minoria". Permitindo que os eleitores votem no mesmo número de candidatos que de vagas a serem preenchidas, não existe o ímpeto a esse tipo de voto estratégico. O resultado desse sistema, portanto, retorna a média das notas a casa dos 2.9 (Figura 43). Os sistemas AVS, *Borda Count* e SVS também impedem a presença do "voto de minoria" e atingem resultados semelhantes.

Figura 43 – BV - Cenário 6

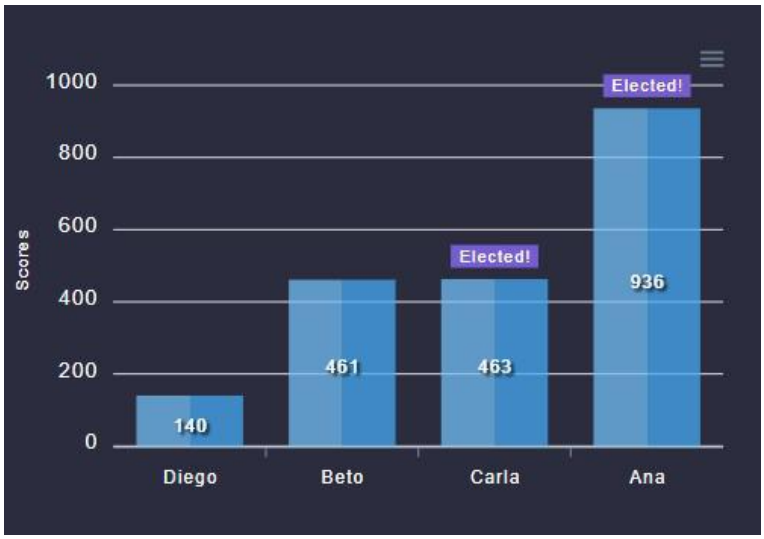


Tabela 18 – Resultados do Cenário 6

Sistema	Candidatos eleitos	Média das notas
FPTP sem voto de minoria	Ana e Beto	2.929
FPTP com 100% de voto de minoria no candidato 3	Ana e Diego	0.505
FPTP com 80% de voto de minoria	Beto e Carla	0.058
Bloc Vote	Ana e Carla	2.913

Esses resultados podem ser replicados acessando os links:

Sem "voto de minoria": <http://electionsim.pythonanywhere.com/#/direct_res/1000001/-/2/[4,0,0,2]/[%22Ana%22,%20%22Beto%22,%20%22Carla%22,%22Diego%22]/-/-/-/-/-/-/100>

100% "voto de minoria" para Diego: <http://electionsim.pythonanywhere.com/#/direct_res/1000001/-/2/[4,0,0,2]/[%22Ana%22,%20%22Beto%22,%20%22Carla%22,%22Diego%22]/-/1/-/[0,0,0,1.0]/-/-/100>

80% "voto de minoria" em Beto, Carla e Diego: <http://electionsim.pythonanywhere.com/#/direct_res/1000001/-/2/[4,0,0,2]/[%22Ana%22,%20%22Beto%22,%20%22Carla%22,%22Diego%22]/-/1/-/[0,0.8,0.8,0.8]/-/-/100>

5 COMO FUNCIONA O SIMULADOR

O programa foi implementado na linguagem *Python* e é composto por oito classes distintas sendo uma classe base e outras sete (uma para cada sistema eleitoral implementado) que estendem a classe base.

5.1 CLASSE `ELECTIONS`

Esta classe funciona como base para todas as outras. Ela contém todas as estruturas requisitadas e acessadas pela simulação de cada sistema eleitoral. Suas principais estruturas são:

- `candidates`: Dicionário composto por chave igual ao índice de cada candidato e valor igual ao número de votos recebidos por esse candidato.

Figura 44 – estrutura `candidates`

- `voters`: Lista de dicionários. Cada dicionário representa um eleitor e é composto por chave igual ao índice de cada candidato e valor igual à nota (-10 a 10) que o eleitor atribui àquele candidato.

- `votes`: Dicionário composto por chave igual ao índice de cada candidato e valor igual a um *set* contendo todos os índices dos eleitores que votaram nesse candidato.

Tais estruturas guardam as informações de todos os candidatos e eleitores gerados. Dois dos parâmetros recebidos pelo construtor da classe `Elections` é o número de candidatos e o número de eleitores a serem criados respectivamente pelos métodos `create_candidates` (apêndice B) e `create_voters` (apêndice A). O método `create_candidates` simplesmente inicializa cada índice das estruturas `candidates` e `votes`.

Eleitores podem ser gerados aleatória ou deterministicamente dependendo da intenção de uso do simulador. É possível gerar eleitores aleatoriamente onde a nota que esse eleitor

Figura 45 – estrutura `voters`

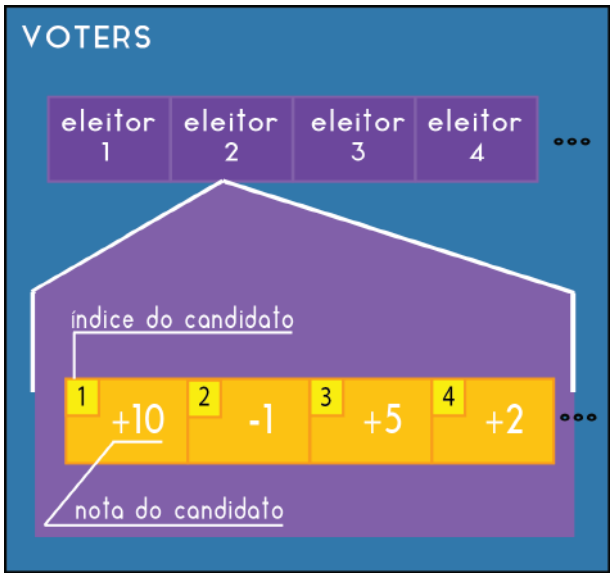
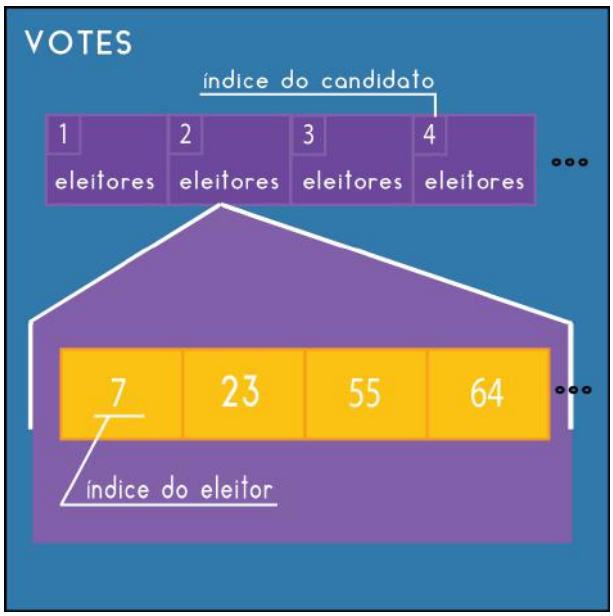


Figura 46 – estrutura `votes`



dará para cada candidato depende da distribuição selecionada para esse. A distribuição *neutral* possui maior probabilidade da nota gerada ser zero ou próxima de zero, a *liked* e *loved* geram notas mais próximas de 10 e de maneira oposta as distribuições *disliked* e *hated* geram notas mais próximas de -10 e por fim as distribuições *polarizer* e *strongly polirizer* geram notas em ambas extremidades e evitam notas próximas de 0. Porém, também é possível criar perfis de eleitores que definirão para determinada porcentagem da população as notas exatas dadas a cada candidato.

Com as listas das devidas notas de cada candidato por eleitor é preciso colocá-las na forma de um ranking para que fique evidente em qual candidato aquele eleitor votaria. Para isso, o método `sort_ranks` (apêndice H) ordena o ranking de cada eleitor

Figura 47 – PDF - Distribuição *Neutral*

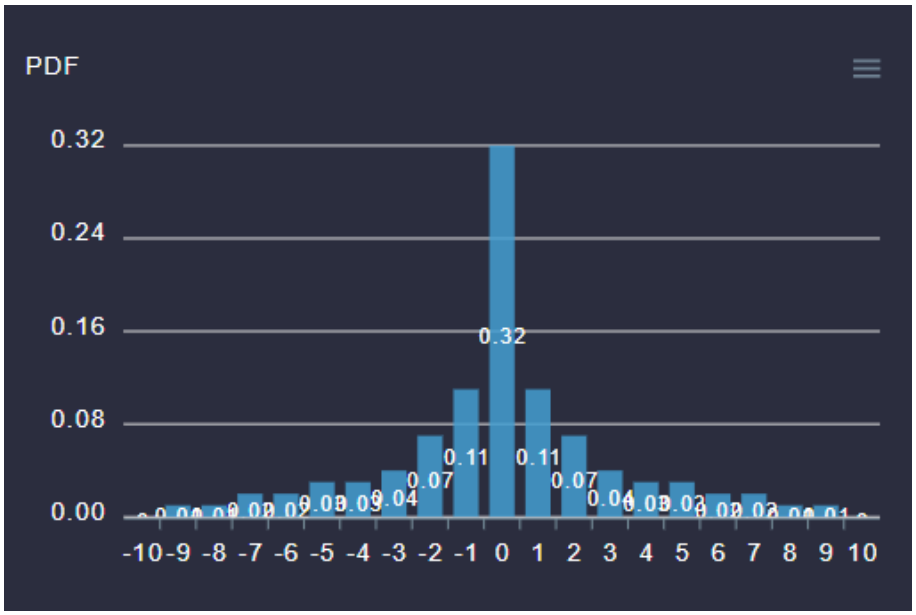
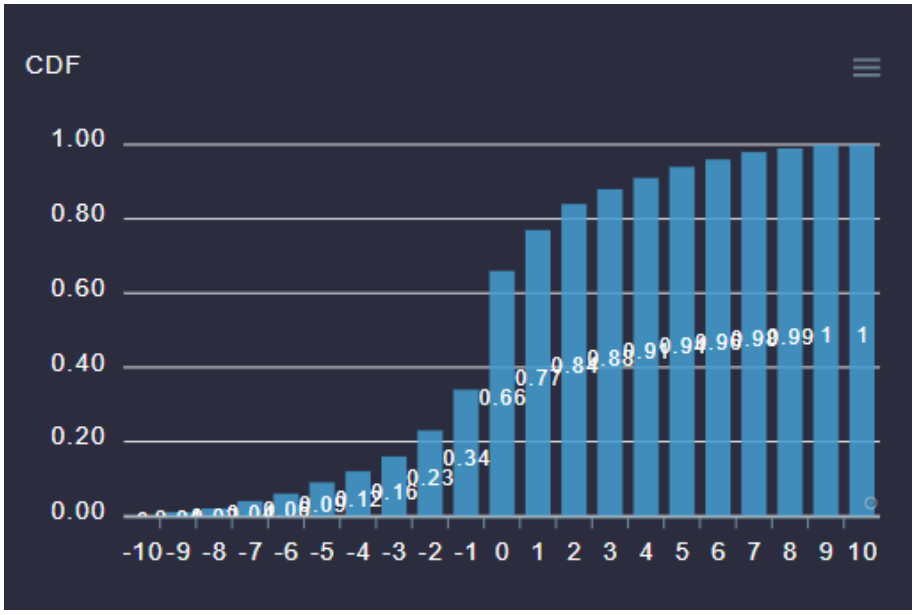


Figura 48 – CDF - Distribuição *Neutral*



na estrutura `voters`. Porém após a ordenação dos rankings é necessário lidar com a duplicidade de notas dentro de um mesmo ranking, pois, se dois ou mais candidatos possuem a mesma nota, ao chamarmos o método de ordenação do *python* o candidato com o maior índice sempre levará prioridade na votação o que não é condizente com a realidade. Então, a partir dos rankings ordenados se cria um dicionário para cada um, onde a chave é a nota e o valor é a lista dos candidatos com a mesma nota. Em seguida cada uma dessas listas é embaralhada dando a cada candidato com a mesma nota uma chance igual de ser escolhido. Esses dicionários são revertidos à estrutura `voters`, agora com uma ordenação justa. Para definir o voto de cada eleitor simplesmente são selecionados

Figura 49 – PDF - Distribuição *Liked*

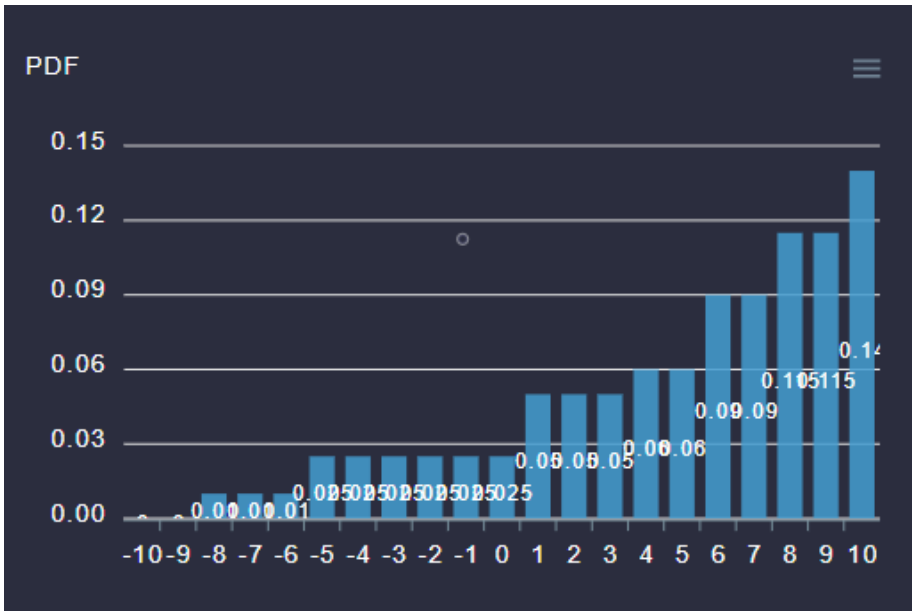
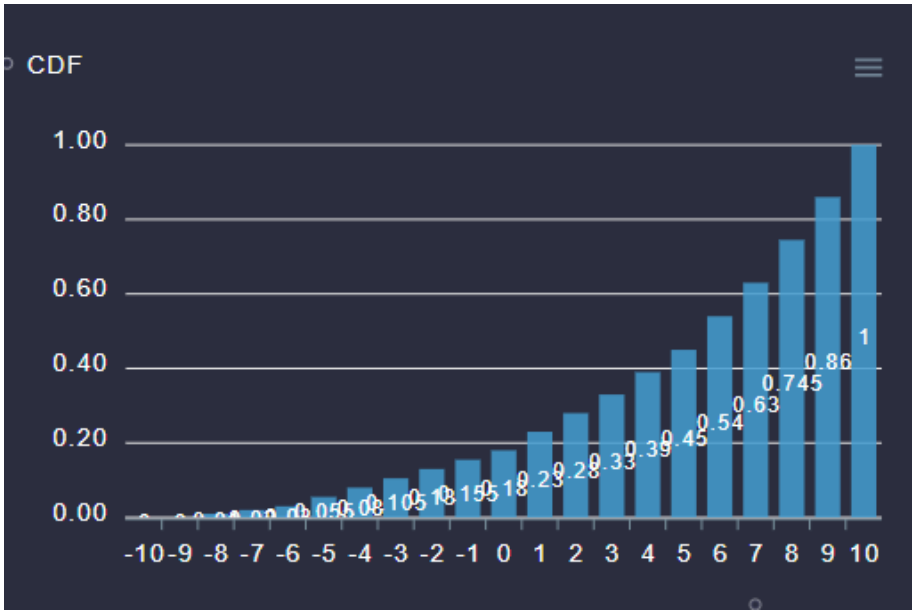


Figura 50 – CDF - Distribuição *Liked*



(dependendo do número de vagas da eleição) os candidatos no topo de cada ranking o que também é feito no método `sort_ranks`. Esses rankings, no entanto, não são fixos, ainda existe a possibilidade da alteração deles após a contabilização dos votos úteis, o que é feito de maneira diferente para cada sistema eleitoral e, portanto, cada um possui métodos próprios. É importante ressaltar também que os votos de todos os eleitores são contabilizados, ou seja, no simulador não existe a possibilidade de voto nulo ou voto em branco mesmo que um eleitor não aprove nenhum candidato.

Após essa contabilização é necessário definir os candidatos que estão na liderança das eleições. Isso é feito para definir quais candidatos serão os alvos dos "votos táticos". Os

Figura 51 – PDF - Distribuição *Disliked*

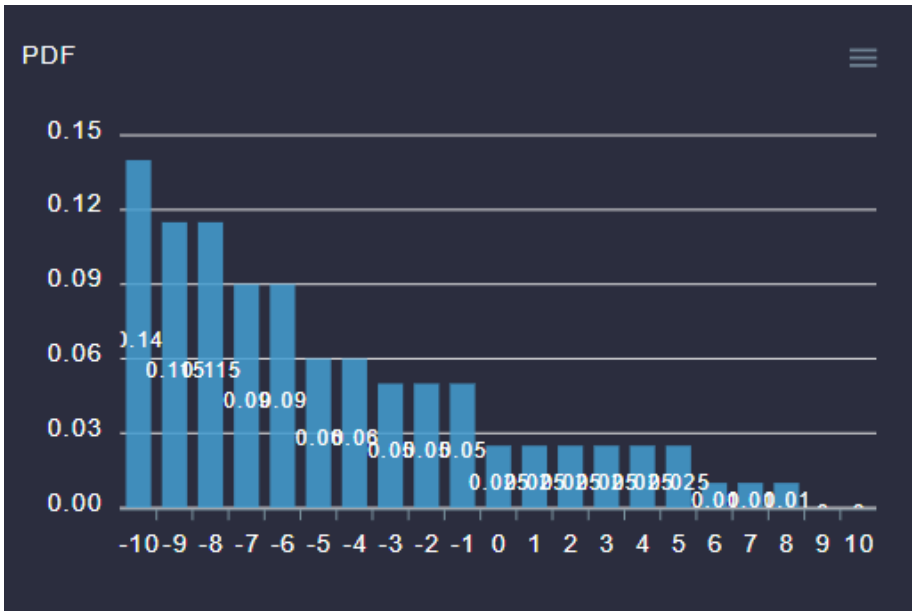
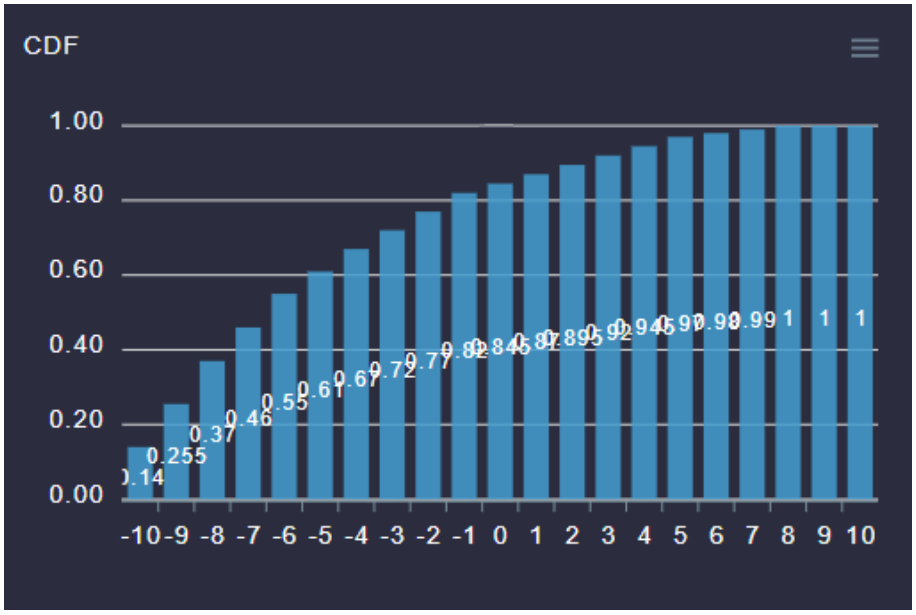


Figura 52 – CDF - Distribuição *Disliked*



candidatos que são considerados como líderes são iguais aos primeiros n+1 candidatos, onde n é o número de vagas ofertadas. A definição dos líderes é feita com o método `set_leading_candidates` (apêndice F). Com isso, a simulação do sistema FPTP está quase completa e a classe `FirstPastThePost` apenas lidará com os votos úteis.

Para cada sistema, o simulador notifica se ele elegeu o melhor candidato. Após a criação dos eleitores, o método `calculate_means` (apêndice C) calcula as média das notas de cada candidato e as guarda em um dicionário chamado `stats`. Quando um sistema escolhe um candidato, esse dicionário é checado e o método `get_mean` (apêndice D) retorna a própria média e `True` se esta é a maior média entre os candidatos. Essa

Figura 53 – PDF - Distribuição *Loved*

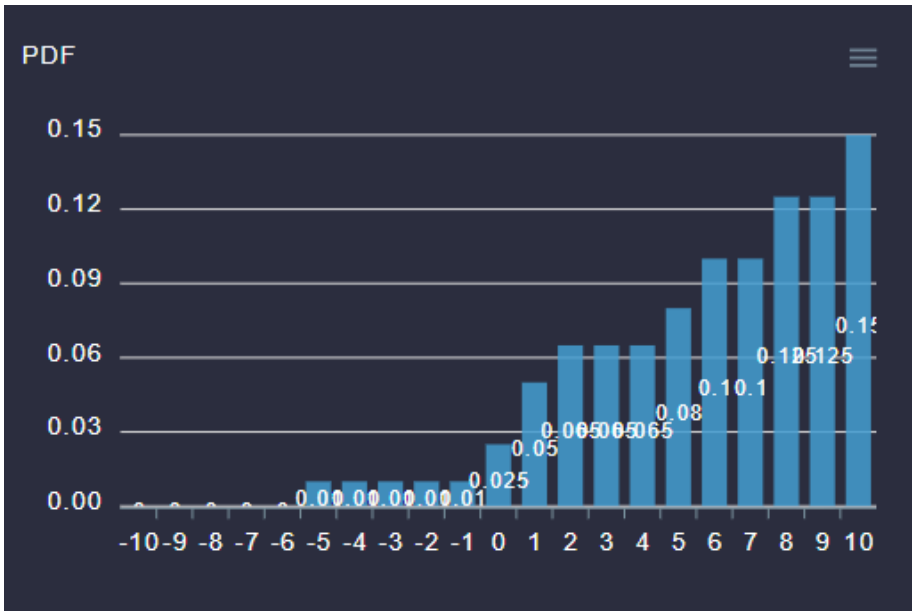
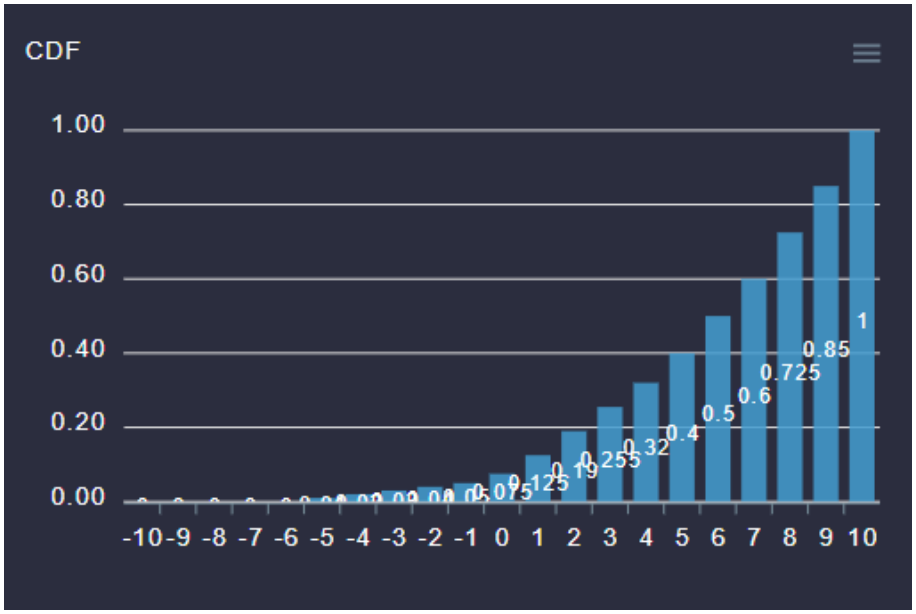


Figura 54 – CDF - Distribuição *Loved*



funcionalidade só existe para eleições com 10 candidatos ou menos e para eleições de 1 vaga senão ficaria custoso guardar todas as combinações de médias para candidatos vitoriosos. No entanto, para esses outros casos existe o método `calculate_mean` (apêndice E) que calcula apenas a média do(s) candidato(s) vitorioso(s). A média das notas é simplesmente o somatório das notas dos candidatos dividido pelo número de eleitores e pelo número de vagas da eleição.

Figura 55 – PDF - Distribuição *Hated*

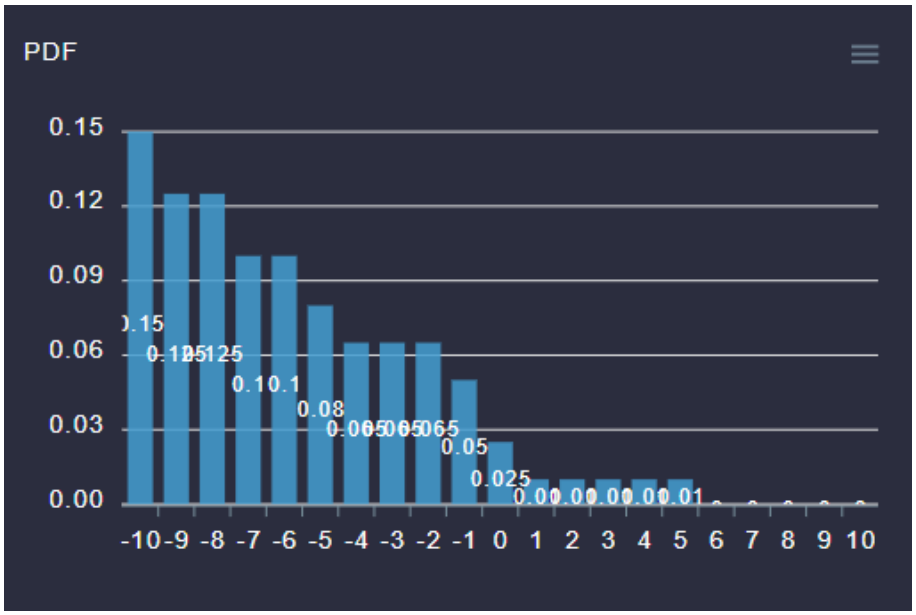
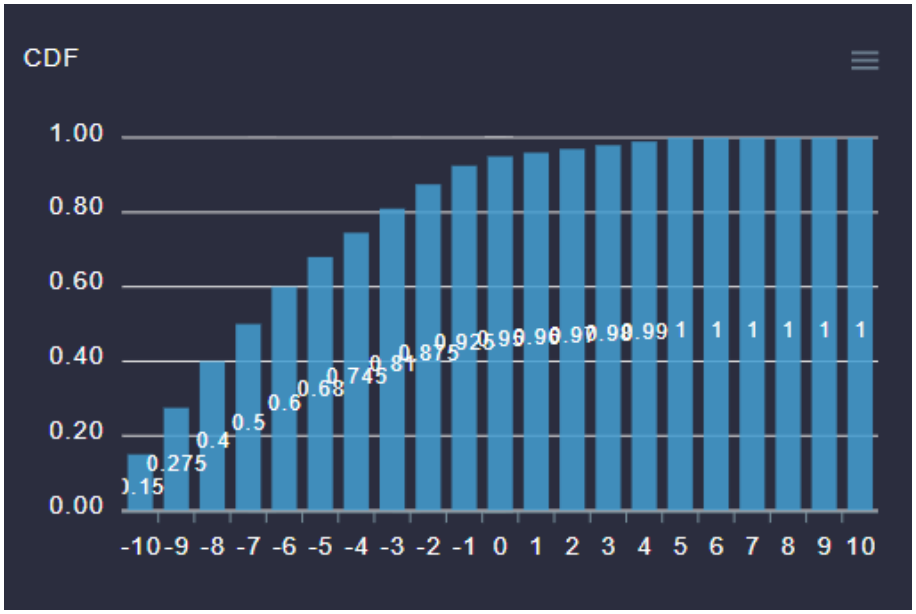


Figura 56 – CDF - Distribuição *Hated*



Média das Notas:

$$\frac{\sum_{i}^{e}\sum_{j}^{v} n_{ij}}{e \times v}$$

Sendo v igual ao número de vagas, e o número de eleitores e n_{ij} a nota dada pelo eleitor i ao candidato eleito para a vaga j.

Nas próximas seções serão comentados os códigos das classes que representam cada sistema eleitoral implementado.

Figura 57 – PDF - Distribuição *Polarizer*

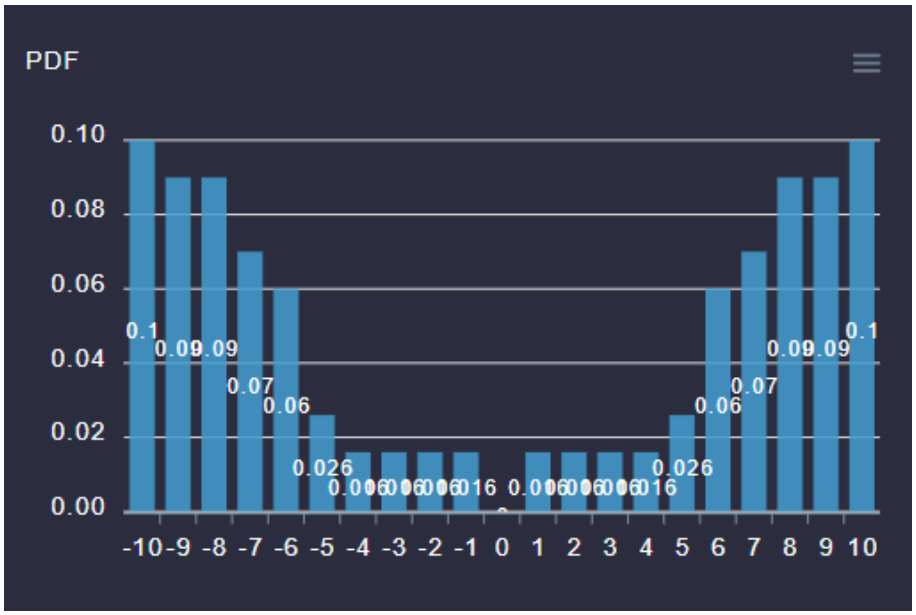
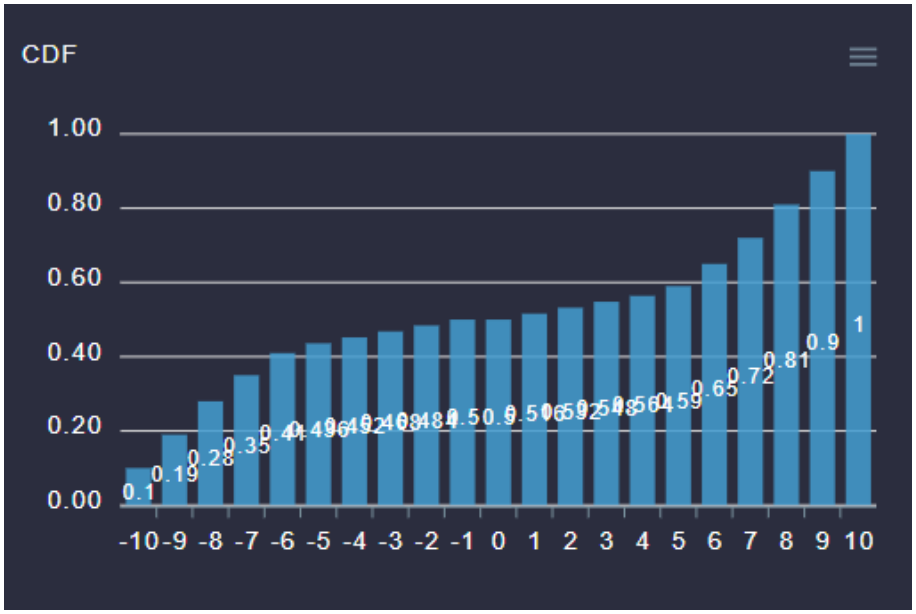


Figura 58 – CDF - Distribuição *Polarizer*



5.2 CLASSE `FIRSTPASTTHEPOST`

Existem dois tipos de votos úteis implementados neste simulador, o "voto tático" e o "voto de minoria". Em ambas as situações a decisão de um eleitor mudar de voto, no entanto, não é certa. A probabilidade da mudança tática de voto é definida por dois parâmetros por candidato (um para cada tipo de voto útil). Ou seja, para cada candidato é possível definir a porcentagem da população de seus eleitores que optará tanto pelo "voto tático" quanto pelo "voto de minoria".

Foi mencionado anteriormente que os rankings de cada eleitor não são fixos e que na contabilização dos votos úteis eles poderiam ser alterados, porém isso não é completa-

Figura 59 – PDF - Distribuição *Strongly Polarizer*

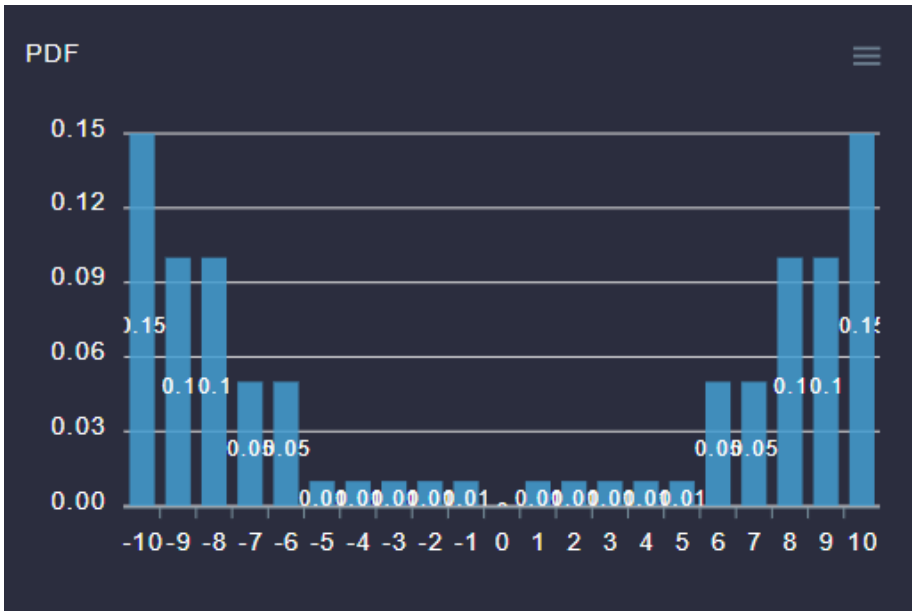
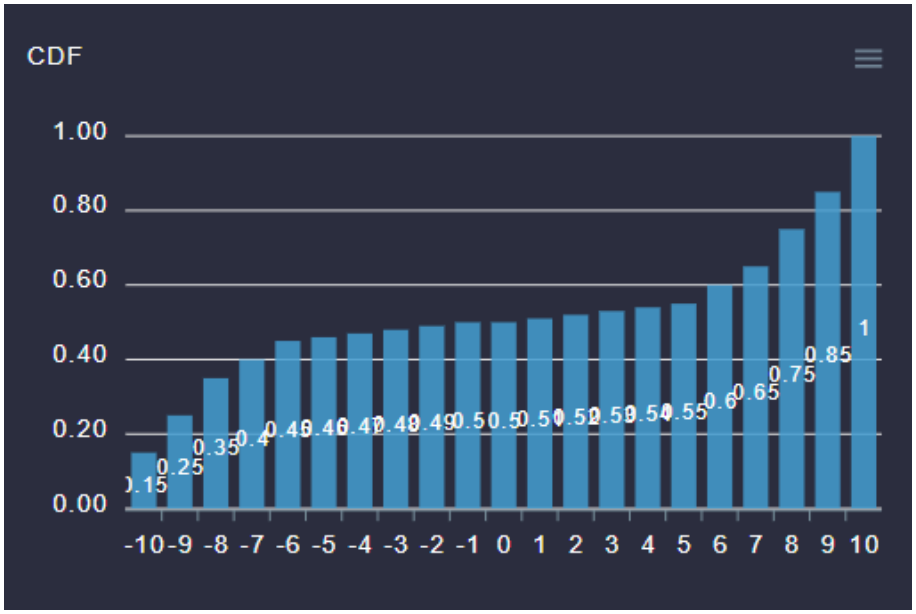


Figura 60 – CDF - Distribuição *Strongly Polarizer*



mente verdade. A estrutura que guarda esses rankings, denominada `sorted_voters`, é o resultado do método `sort_ranks` da classe `Elections`. Essa estrutura é fixa durante toda a simulação e os métodos correspondentes ao voto útil não a modificam, pois cada sistema eleitoral aplica suas mudanças de maneira distinta e para um número grande de eleitores fica muito custoso fazer uma cópia dessa estrutura para cada um deles. Ao invés disso, cada classe guarda uma lista separada contendo apenas os rankings modificados. Essa lista é verificada durante a contabilização dos "votos táticos" e dos "votos de minoria" que são feitas pelos métodos `fptp_count_tactical_votes` (apêndice I) e `fptp_count_minority_votes` (apêndice J) respectivamente.

5.3 CLASSE TWOROUNDSYSTEM

A primeira rodada do *Two-round System* é essencialmente a simulação da classe anterior, porém ao invés de selecionar diretamente o primeiro candidato como vencedor o sistema captura os dois primeiros. Durante essa primeira rodada os dois tipos de voto útil são contabilizados da mesma forma que no FPTP. Em seguida, é feita a checagem da proporção de votos do primeiro colocado, pois se for maior que 50% a segunda rodada não precisa ser simulada. No entanto, antes da simulação da segunda rodada é necessário fazer a contabilização das coalizões. Uma coalizão pode tanto melhorar a nota de um candidato quanto piorá-la dependendo das notas que o eleitor dá para os outros candidatos da coalizão. Para cada candidato de uma coalizão se soma à sua nota a parte inteira da metade das notas dos outros candidatos pertencentes àquela coalizão. Por exemplo, se um eleitor qualquer dá a nota 4 para um candidato C1 que pertence a uma coalizão com um outro candidato C2 a quem foi atribuída a nota 7, ao candidato C1 será somada a parte inteira da metade de 7, resultando em 4+3=7 e ao candidato C2 será atribuída a sua nota original mais a metade de 4, resultando em 7+2=9. Essa contabilização é feita pelo método `trs_account_for_coalitions` (apêndice L). Na segunda rodada todos os votos dos candidatos eliminados são redistribuídos segundo as segundas preferências determinadas pelos rankings dos eleitores. Essa contabilização final é feita com o método `trs_second_round` (apêndice K)

5.4 CLASSE INSTANTRUNOFFVOTING

A simulação do IRV consiste na chamada do método `irv_count_votes` (apêndice M) a cada rodada da eleição. Esse método checa se algum eleitor já possui mais de 50% dos votos ou se a rodada atual é a última rodada (o número de rodadas é sempre um a menos que o número de candidatos). Caso contrário, o candidato com menos votos é adicionado a uma lista de candidatos eliminados e o ranking de seus eleitores são analisados. Se itera por cada ranking, do candidato mais bem ranqueado ao menos, e o primeiro candidato que não estiver na lista de eliminados recebe o voto. Após a transferência dos votos do candidato eliminado a simulação continua para a próxima rodada.

Quando o "voto tático" está habilitado o sistema é simulado duas vezes. A primeira simulação ocorre sem "votos táticos" e seu propósito é definir quem são os líderes da eleição. Ela funciona como uma pesquisa eleitoral para os eleitores que optarem pelo "voto tático". Com isso, essa classe possui um método próprio para preencher a lista de candidatos líderes que simplesmente captura os dois candidatos da última rodada (apêndice G). A segunda simulação roda como a primeira, executando o método `irv_count_votes` a cada rodada, porém com o método `irv_apply_tactical_votes` (apêndice P e Q) sendo executado antes. Esse método simula as estratégias *compromising* e *burying* onde

cada eleitor que opta pelo "voto tático"ranqueia em primeiro lugar sua preferência entre os candidatos líderes e em último lugar o líder concorrente.

5.5 CLASSE `APPROVALVOTING`

Essa classe possui um método para a contagem de votos normal (apêndice N) e um para a contagem quando os "votos táticos" são habilitados no simulador (apêndice O). Na contagem normal se itera sobre todos os rankings contidos na estrutura `sorted_voters`, já mencionada anteriormente, e para cada um deles todos os candidatos com notas acima de 0 recebem um voto, ou seja, são aprovados. O tipo de "voto tático" presente nesse sistema é o *bullet voting*, ele é simulado da seguinte maneira: com os "votos táticos" habilitados, para uma porcentagem dos eleitores (fornecida pelo usuário) de cada candidato, apenas o voto em seu candidato preferido será contabilizado, não importando se outros candidatos também possuam notas maiores que 0. Ou seja, se o usuário definir 50% de "voto tático" para certo candidato, por volta de 50% dos eleitores que o tiverem como primeira opção apenas votam nele e em mais nenhum outro.

5.6 CLASSE `BORDACOUNT`

Neste sistema de votação os candidatos são pontuados de acordo com suas posições nos rankings dos eleitores. O pior colocado recebe zero e a pontuação aumenta de um em um até o mais bem colocado. Essa pontuação é feita no método `bc_sum_candidates_scores` (apêndice R). O "voto tático" para essa classe é exatamente o mesmo utilizado no IRV, sendo simuladas as estratégias de *compromising* e *burying* (apêndice P e Q). Como no IRV, quando os "votos táticos" estão habilitados, duas simulações ocorrem com a primeira delas funcionando como uma pesquisa eleitoral para os eleitores. Essa classe também possui um método próprio para a seleção dos candidatos líderes, o qual é idêntico ao método na classe `Elections`, porém pode ter resultados diferentes da simulação no FPTP e por isso precisa executar separadamente.

5.7 CLASSE `SCOREVOTING`

De maneira similar à classe `ApprovalVoting`, nesta classe também foi implementada uma forma de *bullet voting*. Isso é feito no método `svs_apply_tactical_votes` (apêndice S). Para uma porcentagem da população que opta pelo "voto tático" seus rankings são adulterados onde o candidato de sua preferência receberá a nota máxima(+10) e todos os outros a nota mínima(-10). Logo, é como se eles estivessem restringindo seu voto a apenas um candidato apesar da liberdade presente neste sistema. A contagem final dos pontos é feita no método `svs_sum_candidates_scores` (apêndice T) e é meramente o somatório dos pontos atribuídos a cada candidato.

5.8 CLASSE `blocvote`

O sistema *Bloc Vote* elimina o "voto de minoria" encontrado no sistema FPTP e, portanto, sua classe correspondente apenas possui um método para contagem dos votos e não lida com nenhum tipo de "voto tático". A contagem é feita com o método `bv_count_votes` (apêndice U), nele, todos os n candidatos mais bem colocados no ranking de um eleitor recebem um voto, onde n é igual ao número de vagas da eleição.

5.9 TECNOLOGIAS UTILIZADAS

A implementação foi feita com uma combinação dos frameworks *Django* para a linguagem *Python*, utilizado no *backend* da aplicação, e *React* para *Javascript* no *frontend*. O *React* é uma biblioteca para o desenvolvimento de interfaces de usuário. Sua vantagem é a possibilidade de criação de componentes, ou classes, em javascript que podem ser multiplicados e reutilizados na estruturação da aplicação. Django segue o modelo de arquitetura MTV que separa a aplicação em três camadas, *Models*, *Templates* e *Views* que é uma interpretação própria do modelo MVC. A integração desses dois frameworks é possível com o *module bundler Webpack* que por sua vez gera um arquivo estático que é interpretado pelo *Django* a partir dos componentes do *React*.

Em relação à parte estética do site foi incorporado o tema *Lux* do *Bootstrap* e os gráficos apresentados na sessão de resultados são fornecidos pela biblioteca *Apex Charts*.

O gerenciamento de estados no *frontend* da aplicação é feito com o uso de uma biblioteca chamada *Redux*, a criação de testes unitários com a biblioteca *Nose*, o versionamento do projeto foi feito com o auxílio do *Github* e por fim a aplicação está sendo hospedada no site *pythonanywhere*.

Todo o código se encontra no link: <https://github.com/gsbevilaqua/elections>

6 CONCLUSÃO

Dentre os sistemas abordados neste trabalho, é nítido que o FPTP é o sistema que menos se esforça em satisfazer frações significativas da população, não apenas por ser um sistema de pluralidade, mas também por ser elevadamente suscetível a mais de um tipo de voto estratégico. O TRS, apesar de ser um avanço em relação ao FPTP, por impor a necessidade da maioridade, seus resultados ainda podem ser manipulados através da formação de coalizões e sua estrutura é muito mais complacente ao voto tático do que o IRV.

Mesmo assim, diante dos resultados obtidos no Capítulo 4 existe um impasse a ser considerado. Por um lado temos o IRV que se comporta melhor do que suas alternativas mais comumente usadas, o FPTP e o TRS, e que possui certa resistência aos riscos envolvidos com os votos táticos, mas que nem sempre retorna os melhores resultados e possui a estranha característica da não-monotonicidade. Por outro lado, temos sistemas como o *Borda Count* e o SVS que parecem ser consistentes em retornar o melhor candidato, mas que podem ser completamente desajustados se um grupo razoável de indivíduos resolver votar de maneira estratégica ao invés de votar em suas reais preferências.

Em relação às eleições com mais de uma vaga, as dificuldades também emergem da votação estratégica. Os denominados neste trabalho "votos de minoria", quando presentes, afetam profundamente o FPTP, enquanto as alternativas Bloc Vote, *Borda Count*, AVS e SVS são mais eficientes porque os evitam.

É evidente ver que o sistema ideal seria um que ao mesmo tempo que, em sua estrutura, restringisse ao máximo qualquer tentativa de pensamento estratégico, como no IRV, mas que também permitisse o maior grau de expressividade possível ao eleitor e absorvesse toda a intenção do mesmo, assim como ocorre no SVS. Este sistema híbrido, formado por esses outros dois, talvez já seja o próprio *Borda Count*, que possui um tipo de voto tático, mas que, quando presente, pode ser tão desastroso e frívolo que talvez seja o necessário para fazer com que a população votante perceba que votar de maneira verdadeira é sim de seu interesse.

Portanto, se não for possível o desenvolvimento de um sistema que permita, ao mesmo tempo, alta expressividade aos eleitores e a anulação de qualquer tipo de voto estratégico, talvez a melhor alternativa seja uma maior educação eleitoral sobre os malefícios deste tipo de pensamento e com isso a utilização de sistemas eficientes como o *Borda Count* e o SVS.

REFERÊNCIAS

AMY, D. J. **Behind the Ballot Box: A Citizen's Guide to Voting Systems**. 88 Post Road West, Westport, CT 06881: Praeger Publishers, 2000.

Bartholdi, John J., III and Orlin, James B. Single transferable vote resists strategic voting. 1990. Disponível em: <https://www2.isye.gatech.edu/~jjb/papers/stv.pdf>. Acesso em: 29 jul.2019.

Electoral Reform Society. Bloc vote. 2017. Disponível em: <https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/first-past-the-post/block-vote/>. Acesso em: 29 mai.2019.

Electoral Reform Society. Borda count. 2017. Disponível em: <https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/borda-count/>. Acesso em: 28 mai.2019.

HAMLIN, A. Who uses approval voting? 2015. Disponível em: <https://www.electionscience.org/voting-methods/approval-voting-progress/>. Acesso em: 27 mai.2019.

LIPPMAN, D. Borda count. 2012. Disponível em: <https://courses.lumenlearning.com/waymakermath4libarts/chapter/borda-count/>. Acesso em: 28 mai.2019.

ORNSTEIN, J. T. Monotonicity failure under irv. 2018. Disponível em: <https://joeornstein.github.io/MonotonicityFailure.html>. Acesso em: 29 mai.2019.

Pew Research Center. Nearly six-in-ten governments are democracies. 2017. Disponível em: <https://www.pewresearch.org/fact-tank/2017/12/06/despite-concerns-about-global-democracy-nearly-six-in-ten-countries-are-now-democratic/ft_17-11-10_demo_auto_trend/>. Acesso em: 07 abr.2019.

POUNDSTONE, W. **Prisoner's Dilemma**. 1745 Broadway, New York, NY 10019: Doubleday, 1992.

The Center for Election Science. **Bullet Voting**. 2015. Disponível em: <https://www.electionscience.org/library/bullet-voting/>. Acesso em: 29 mai.2019.

APÊNDICES

APÊNDICE A – MÉTODO CREATE_VOTERS.

Código A.1 – Método create_voters

```
def create_voters(self):
  if(not len(self.voter_profiles)):
        for voter in range(self.N_VOTERS):
        candidates_rank = dict()
          for candidate in reversed(range(self.N_CANDIDATES)):
                if(self.BIAS_VECTOR[candidate]==4):
                  candidates_rank[candidate] = self._sortear(self.loved)
                elif(self.BIAS_VECTOR[candidate]==3):
                  candidates_rank[candidate] = self._sortear(self.liked)
                elif(self.BIAS_VECTOR[candidate]==2):
                  candidates_rank[candidate] = self._sortear(self.disliked)
                elif(self.BIAS_VECTOR[candidate]==1):
                  candidates_rank[candidate] = self._sortear(self.hated)
                elif(self.BIAS_VECTOR[candidate]==-1):
                  candidates_rank[candidate] = self._sortear(self.polarizer
                      )
                elif(self.BIAS_VECTOR[candidate]==-2):
                  candidates_rank[candidate] = self._sortear(self.
                     more_polarizer)
                else:
                  candidates_rank[candidate] = self._sortear(self.neutral)
          self.voters.append(candidates_rank)
  else:
        ranges = []
        ranks = []
        for prof in self.voter_profiles:
          ranges.append(int(prof["pop_percentage"]))
          rank = {}
          for index, score in enumerate(prof["scores"]):
                rank[index] = score
          ranks.append(rank)
        for index, _range in enumerate(ranges):
          for _ in range(int(self.N_VOTERS*(_range/100))):
                self.voters.append(ranks[index])
```

APÊNDICE B – MÉTODO CREATE_CANDIDATES.

Código B.1 – Método create_candidates

```
def create_candidates(self):
  for i in range(self.N_CANDIDATES):
    self.candidates[i] = 0
    self.votes[i] = set()
```

APÊNDICE C – MÉTODO CALCULATE_MEANS.

Código C.1 – Método calculate_means

```
def calculate_means(self):
  for candidate in range(self.N_CANDIDATES):
        rating_sum = 0
        for voter in self.voters:
          rating_sum += voter[candidate]

        self.stats['means'][candidate] = rating_sum/self.N_VOTERS
```

APÊNDICE D – MÉTODO GET_MEAN.

Código D.1 – Método get_mean

```
def get_mean(self, winners):
        if len(winners) > 1 or self.N_CANDIDATES > 10:
                return self.calculate_mean(winners)
        else:
                chose_best = True if winners[0] == self.best_candidate else
                    False
                return self.stats['means'][winners[0]], chose_best
```

APÊNDICE E – MÉTODO CALCULATE_MEAN.

Código E.1 – Método calculate_mean

```
def calculate_mean(self):
  rating_sum = 0
  for voter in self.voters:
        for candidate in winners:
                rating_sum += voter[candidate]
  return rating_sum/(self.N_VOTERS*self.N_VACANCIES), False
```

APÊNDICE F – MÉTODO SET_LEADING_CANDIDATES.

Código F.1 – Método set_leading_candidates

```
def set_leading_candidates(self):
  for i in range(1, self.N_VACANCIES + 2):
        self.leading_candidates.append(self.sorted_candidates[-i][self.
           CANDIDATE_INDEX])
```

APÊNDICE G – MÉTODO IRV_SET_LEADING_CANDIDATES.

Código G.1 – Método irv_set_leading_candidates

```
def _set_leading_candidates(self):
  for leader in self.elec.rounds[-1]:
        self.leading_candidates.append(leader[0])
```

APÊNDICE H – MÉTODO SORT_RANKS.

Código H.1 – Método sort_ranks

```
def sort_ranks(self):
  self._account_for_coalitions()

  for voter in self.voters:
    self.sorted_voters.append(sorted(voter.items(), key=operator.itemgetter
        (1)))

  temp = []

  for voter in self.sorted_voters:
    new_dict = defaultdict(list)
    for k in voter:
      new_dict[k[1]].append(k[0])
    temp.append(new_dict)

  self.sorted_voters = []

  for voter_index, current_voter in enumerate(temp):
    new_voter = []
          for rating, candidate_indexes in current_voter.items():
            if len(current_voter[rating]) > 1:
              shuffle(current_voter[rating])
              for e in candidate_indexes:
                new_voter.append((e, rating))
            else:
              new_voter.append((candidate_indexes[0], rating))
    self.sorted_voters.append(new_voter)
    self.candidates[new_voter[-1][0]] += 1
    self.votes[new_voter[-1][0]].add(voter_index)
```

APÊNDICE I – MÉTODO FPTP_COUNT_TACTICAL_VOTES.

Código I.1 – Método fptp_count_tactical_votes

```
def count_tactical_votes(self):
  for candidate in self.votes_copy:
        if candidate not in self.elec.leading_candidates:
          for voter_index in self.votes_copy[candidate]:
                for index, _candidate in enumerate(reversed(self.elec.
                    sorted_voters[voter_index])):
                  if _candidate[self.elec.CANDIDATE_INDEX] in self.elec.
                      leading_candidates
                    if random.random() < self.elec.
                        tactical_vote_percentages[_candidate[self.elec.
                        CANDIDATE_INDEX]]:
                          self.rankings_changed[voter_index] = copy.
                              deepcopy(self.elec.sorted_voters[voter_index])
                          self.rankings_changed[voter_index][-(index + 1)],
                               self.rankings_changed[voter_index][-1] = self
                              .rankings_changed[voter_index][-1], self.
                              rankings_changed[voter_index][-(index + 1)]
                          self.candidates[candidate] -= 1
                          self.candidates[_candidate[self.elec.
                              CANDIDATE_INDEX]] += 1
                          self.votes[candidate].remove(voter_index)
                          self.votes[_candidate[self.elec.CANDIDATE_INDEX
                              ]].add(voter_index)
                          break
                      break
```

APÊNDICE J – MÉTODO FPTP_COUNT_MINORITY_VOTES.

Código J.1 – Método fptp_count_minority_votes

```
def count_minority_votes(self):
  half_vacancies = math.floor(self.elec.N_VACANCIES/2)
  for candidate in self.elec.leading_candidates:
        if self.elec.leading_candidates.index(candidate) >= half_vacancies:
          continue
        for voter_index in self.votes_copy[candidate]:
          if voter_index in self.rankings_changed:
                ranking = self.rankings_changed[voter_index]
          else:
                ranking = copy.deepcopy(self.elec.sorted_voters[voter_index
                   ])
          for index, _candidate in enumerate(reversed(ranking)):
                if _candidate[self.elec.CANDIDATE_INDEX] not in self.elec.
                   leading_candidates or (_candidate[self.elec.
                   CANDIDATE_INDEX] in self.elec.leading_candidates and
                   self.elec.leading_candidates.index(_candidate[self.elec.
                   CANDIDATE_INDEX]) >= half_vacancies):
                  if _candidate[self.elec.CANDIDATE_RANK] >= 0:
                        if random.random() < self.elec.
                           minority_vote_percentages[_candidate[self.elec.
                           CANDIDATE_INDEX]]:
                          self.candidates[candidate] -= 1
                          self.candidates[_candidate[self.elec.
                             CANDIDATE_INDEX]] += 1
                          break
                        break
                  break
```

APÊNDICE K – MÉTODO TRS_SECOND_ROUND.

Código K.1 – Método trs_second_round

```
def trs_second_round(self):
  self._account_for_coalitions()
  for candidate in self.votes:
        if candidate != self.winner and candidate != self.second_place:
          for voter_index in self.votes[candidate]:
                if voter_index in self.rankings_changed:
                  ranking = self.rankings_changed[voter_index]
                else:
                  ranking = copy.deepcopy(self.elec.sorted_voters[
                      voter_index])
                for index2, candidate in enumerate(reversed(ranking)):
                  if candidate[self.elec.CANDIDATE_INDEX] == self.winner:
                        self.candidates[self.winner] += 1
                        self.votes[self.winner].add(voter_index)
                        break
                  elif candidate[self.elec.CANDIDATE_INDEX] == self.
                      second_place:
                        self.candidates[self.second_place] += 1
                        self.votes[self.second_place].add(voter_index)
                        break

  self.sorted_candidates = self.elec.sort_candidates(self.candidates)
  winners = []
  vacancies = self.elec.N_VACANCIES
  for candidate in reversed(self.sorted_candidates):
        if vacancies == 0:
          break
        winners.append(candidate[0])
        vacancies -= 1
  mean, chose_best = self.elec.get_mean(winners = winners)
```

APÊNDICE L – MÉTODO TRS_ACCOUNT_FOR_COALITIONS.

Código L.1 – Método trs_account_for_coalitions

```
def account_for_coalitions(self):
  for voter_index, voter in enumerate(self.elec.sorted_voters):
        voter_dict = dict()
        og_voter_dict = dict()
        for tup in voter:
          voter_dict[tup[0]] = tup[1]
          og_voter_dict[tup[0]] = tup[1]
        for coalition in self.elec.coalitions:
          for candidate in coalition:
                add_to_score = 0
                for candidate2 in coalition:
                  if candidate == candidate2:
                        continue
                  half = og_voter_dict[candidate2['value']]/2
                  if half < 0:
                        add_to_score += math.ceil(half)
                  else:
                        add_to_score += math.floor(half)
                if voter_dict[candidate['value']] + add_to_score > 10:
                  voter_dict[candidate['value']] = 10
                elif voter_dict[candidate['value']] + add_to_score < -10:
                  voter_dict[candidate['value']] = -10
                else:
                  voter_dict[candidate['value']] += add_to_score
        self.rankings_changed[voter_index] = []
        for candidate in voter_dict:
          self.rankings_changed[voter_index].append((candidate, voter_dict[
             candidate]))
        self.rankings_changed[voter_index] = sorted(self.rankings_changed[
           voter_index], key=lambda x: x[1])
```

APÊNDICE M – MÉTODO IRV_COUNT_VOTES.

Código M.1 – Método irv_count_votes

```
def irv_count_votes(self, _round):
  self.elec.rounds.append(self.sorted_candidates[_round:])
  if(self.sorted_candidates[self.elec.N_CANDIDATES - 1][1]/self.elec.
     N_VOTERS == 0.5 and _round == self.elec.N_CANDIDATES - self.elec.
     N_VACANCIES - 1):
        return 2
  elif(self.sorted_candidates[self.elec.N_CANDIDATES - 1][1]/self.elec.
     N_VOTERS > 0.5):
        return 1
  else:
        self.elec.excluded.add(self.sorted_candidates[_round][self.elec.
           CANDIDATE_INDEX])
        for voter_index in self.votes[self.sorted_candidates[_round][self.
           elec.CANDIDATE_INDEX]]:
          if voter_index in self.rankings_changed:
                for candidate in reversed(self.rankings_changed[voter_index
                   ]):
                  if candidate[self.elec.CANDIDATE_INDEX] not in self.elec.
                     excluded:
                        self.candidates[candidate[self.elec.CANDIDATE_INDEX
                           ]] += 1
                        self.votes[candidate[self.elec.CANDIDATE_INDEX]].
                           add(voter_index)
                        break
                  else:
                        continue
          else:
                for candidate in reversed(self.elec.sorted_voters[
                   voter_index]):
                  if candidate[self.elec.CANDIDATE_INDEX] not in self.elec.
                     excluded:
                        self.candidates[candidate[self.elec.CANDIDATE_INDEX
                           ]] += 1
                        self.votes[candidate[self.elec.CANDIDATE_INDEX]].
                           add(voter_index)
                        break
                  else:
                        continue

        self.sorted_candidates = self.elec.sort_candidates(self.candidates)
        return 0
```

APÊNDICE N – MÉTODO AVS_COUNT_VOTES.

Código N.1 – Método avs_count_votes

```
  def avs_count_votes(self):
    for index in range(self.elec.N_CANDIDATES):
      self.candidates[index] = 0
    for voter in self.elec.sorted_voters:
      for candidate in reversed(voter):
        if candidate[self.elec.CANDIDATE_SCORE] > 0:
          self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += 1
        else:
          break
```

APÊNDICE O – MÉTODO AVS_COUNT_VOTES_WITH_TACTICAL.

Código O.1 – Método avs_count_votes_with_tactical

```
def _count_votes_with_tactical(self):
  for index in range(self.elec.N_CANDIDATES):
    self.candidates[index] = 0
  for voter in self.elec.sorted_voters:
    if random.random() < self.elec.tactical_vote_percentages[voter[-1][self
       .elec.CANDIDATE_INDEX]]:
      self.candidates[voter[-1][self.elec.CANDIDATE_INDEX]] += 1
    else:
      for candidate in reversed(voter):
        if candidate[self.elec.CANDIDATE_SCORE] > 0:
          self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += 1
        else:
          break
```

APÊNDICE P – MÉTODO IRV_APPLY_TACTICAL_VOTES.

Código P.1 – Método irv_apply_tactical_votes

```
def irv_apply_tactical_votes(self):
  pos = set()
  for index, perc in enumerate(self.elec.tactical_vote_percentages):
    if perc > 0:
      pos.add(index)
  for candidate in self.elec.candidates:
    if candidate not in self.leading_candidates:
      for voter_index in self.votes[candidate]:
        raised = None
        for candidate_tuple in reversed(self.elec.sorted_voters[voter_index
            ]):
          if candidate_tuple[self.elec.CANDIDATE_INDEX] in self.
              leading_candidates:
            if candidate_tuple[self.elec.CANDIDATE_INDEX] in pos:
              if random.random() < self.elec.tactical_vote_percentages[
                  candidate_tuple[self.elec.CANDIDATE_INDEX]]:
                ranking = [None]*self.elec.N_CANDIDATES
                ranking[-1] = (candidate_tuple[self.elec.CANDIDATE_INDEX],
                    self.elec.voters[voter_index][candidate_tuple[self.elec.
                    CANDIDATE_INDEX]])
                raised = candidate_tuple[self.elec.CANDIDATE_INDEX]
                break
              break
            break
        if raised != None:
          for _candidate in self.leading_candidates:
            if _candidate != ranking[-1][0]:
              ranking[0] = (_candidate, self.elec.voters[voter_index][
                  _candidate])
              buried = _candidate
              break
          i = 1
          for candidate_tuple in self.elec.sorted_voters[voter_index]:
            if candidate_tuple[self.elec.CANDIDATE_INDEX] in [raised,
                buried]:
              continue
            else:
              ranking[i] = (candidate_tuple[self.elec.CANDIDATE_INDEX],
                  self.elec.voters[voter_index][candidate_tuple[self.elec.
                  CANDIDATE_INDEX]])
              i += 1
          self.rankings_changed[voter_index] = ranking
```

APÊNDICE Q – MÉTODO IRV_APPLY_TACTICAL_VOTES (CONTINUAÇÃO).

Código Q.1 – Método irv_apply_tactical_votes (continuação)

```
    elif candidate in pos:
      for voter_index in self.votes[candidate]:
        if random.random() < self.elec.tactical_vote_percentages[candidate
            ]:
          for _candidate in self.leading_candidates:
            if _candidate != candidate:
              ranking = [None]*self.elec.N_CANDIDATES
              ranking[0] = (_candidate, self.elec.voters[voter_index][
                  _candidate])
              buried = _candidate
              break
          i = 1
          for candidate_tuple in self.elec.sorted_voters[voter_index]:
            if candidate_tuple[self.elec.CANDIDATE_INDEX] == buried:
              continue
            else:
              ranking[i] = (candidate_tuple[self.elec.CANDIDATE_INDEX],
                  self.elec.voters[voter_index][candidate_tuple[self.elec.
                  CANDIDATE_INDEX]])
              i += 1
          self.rankings_changed[voter_index] = ranking
```

APÊNDICE R – MÉTODO BC_SUM_CANDIDATES_SCORES.

Código R.1 – Método bc_sum_candidates_scores

```
def bc_sum_candidates_scores(self):
  for index in range(self.elec.N_CANDIDATES):
    self.candidates[index] = 0
  for voter_index, voter in enumerate(self.elec.sorted_voters):
    score = 0
    if voter_index in self.rankings_changed:
      for candidate in self.rankings_changed[voter_index]:
        self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += score
        score += 1
    else:
      for candidate in voter:
        self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += score
        score += 1
```

APÊNDICE S – MÉTODO SVS_APPLY_TACTICAL_VOTES.

Código S.1 – Método svs_apply_tactical_votes

```
def svs_apply_tactical_votes(self):
  for voter_index, voter in enumerate(self.elec.sorted_voters):
    if random.random() < self.elec.tactical_vote_percentages[voter[-1][self
       .elec.CANDIDATE_INDEX]]:
      self.rankings_changed[voter_index] = copy.deepcopy(self.elec.
         sorted_voters[voter_index])
      for candidate_index, candidate in enumerate(reversed(voter)):
        if candidate_index == 0:
          self.rankings_changed[voter_index][-(candidate_index + 1)] = (
             voter[-(candidate_index + 1)][self.elec.CANDIDATE_INDEX], 10)
        else:
          self.rankings_changed[voter_index][-(candidate_index + 1)] = (
             voter[-(candidate_index + 1)][self.elec.CANDIDATE_INDEX], -10)
```

APÊNDICE T – MÉTODO SVS_SUM_CANDIDATES_SCORES.

Código T.1 – Método svs_sum_candidates_scores

```
def svs_sum_candidates_scores(self):
  for index in range(self.elec.N_CANDIDATES):
    self.candidates[index] = 0
  for voter_index, voter in enumerate(self.elec.sorted_voters):
    if voter_index in self.rankings_changed:
      for candidate in self.rankings_changed[voter_index]:
        self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += candidate[
            self.elec.CANDIDATE_RANK]
    else:
      for candidate in voter:
        self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += candidate[
            self.elec.CANDIDATE_RANK]
```

APÊNDICE U – MÉTODO BV_COUNT_VOTES.

Código U.1 – Método bv_count_votes

```
def bv_count_votes(self):
  for index in range(self.elec.N_CANDIDATES):
    self.candidates[index] = 0
  for voter in self.elec.sorted_voters:
    votes = self.elec.N_VACANCIES
    for candidate in reversed(voter):
      if votes == 0:
        break
      self.candidates[candidate[self.elec.CANDIDATE_INDEX]] += 1
      votes -= 1
```