Aplicação Web Classificadora de Operadoras de Telefonia Móvel em Áreas de Interesse

VICTOR ANDRADE FULGÊNCIO[1], MARCOS TOMIO KAKITANI [2] (Orientador)
[1] Universidade Federal de São João del-Rei, Campus Alto Paraopeba (e-mail: victorandradefulgencio@aluno.ufsj.edu.br)
[2] Departamento das Engenharias de Telecomunicações e Mecatrônica, Universidade Federal de São João del-Rei, Campus Alto Paraopeba (e-mail: marcos.kakitani@ufsj.edu.br)

RESUMO Nem sempre é fácil definir qual operadora de telefonia móvel contratar. Por isso, neste trabalho, propomos um algoritmo *fuzzy*, ou difuso, e uma aplicação *web* para apoiar esse tipo de tomada de decisão. Para alimentar o algoritmo, dados reais foram utilizados, provenientes do plano de dados abertos da Anatel. Desta forma, com parâmetros de reclamações, qualidade de serviço, percentual de cobertura das redes móveis e lógica *fuzzy*, podemos classificar as operadoras e ajudar os usuários da telefonia móvel.

Palavras-chave Algoritmo. Anatel. Classificar. Lógica Fuzzy. Lógica Difusa. Operadoras. Telefonia Celular. www.classificadoroperadoras.com.br.

I. INTRODUÇÃO

O crescente número de acessos à telefonia móvel no Brasil, combinado com o aumento da necessidade dos brasileiros em estarem cada vez mais conectados [1], torna a decisão de escolher a melhor operadora muito importante, porém complexa. Visando diminuir a dificuldade de encontrar uma boa operadora de telefonia celular para cada consumidor, propomos um algoritmo *fuzzy*, que apresenta a classificação das melhores operadoras para um determinado usuário como saída.

O algoritmo *fuzzy* possui as seguintes entradas:

1) A cidade de um usuário.
2) As áreas dessa cidade consideradas mais importantes pelo usuário.
3) O preço dos planos disponibilizados pelas operadoras.
4) Percentual de cobertura das operadoras, proveniente da Anatel.
5) Métricas de qualidade de serviço, proveniente da Anatel.

Também foi criada uma aplicação web de código aberto como forma de acessar o algoritmo, disponível no endereço www.classificadoroperadoras.com.br.

Ao final do trabalho, também serão propostas algumas melhorias para, potencialmente, tornar esta aplicação mais escalável, rápida e comercial.

A. LÓGICA FUZZY

Lógica *fuzzy*, também conhecida por Lógica Nebulosa, ou Lógica Difusa, é um tipo de interpolação que permite que sistemas lidem com a subjetividade de várias situações do nosso dia a dia [2]. Podemos notar que a escolha de uma operadora de telefonia é bastante subjetiva, já que é muito dependente de parâmetros sociais e experiências pessoais de cada indivíduo, um grande motivo da escolha de uma abordagem *fuzzy* ao algoritmo.

Uma abordagem *fuzzy* também é conveniente quando temos diversos parâmetros de entrada, e não temos uma regra da saída do sistema totalmente determinística, que também é o caso da nossa tomada de decisão. Com a lógica *fuzzy*, temos a possibilidade de, através de regras sociais, obter resultados exatos, ou aproximações escalares muito próximas da realidade.

B. PRINCÍPIOS BÁSICOS DO ALGORITMO

Primeiramente, é necessário compreender todas as entradas do sistema e, posteriormente, associá-las a *Fuzzy Sets*. *Fuzzy Sets*, ou conjuntos difusos, são entidades que possuem elementos com diferentes níveis de pertencimento, relacionados à uma função denominada *Membership Functions*, ou funções de pertencimento [3]. No caso desta aplicação temos os conjuntos listados pela Tabela 1:

TABELA 1. Conjuntos difusos, ou *Fuzzy Sets* usados na aplicação.

Papel	Nome	Níveis
Entrada	Percentual de cobertura da rede 2G na cidade	bom, regular ou ruim. $(0\% \rightarrow 100\%)$
Entrada	Percentual de cobertura da rede 3G na cidade	bom, regular ou ruim. $(0\% \rightarrow 100\%)$
Entrada	Percentual de cobertura da rede 4G na cidade	bom, regular ou ruim. $(0\% \rightarrow 100\%)$
Entrada	Percentual de cobertura da rede 2G nas áreas da cidade destacadas pelo usuário	bom, regular ou ruim. $(0\% \rightarrow 100\%)$
Entrada	Percentual de cobertura da rede 3G nas áreas da cidade destacadas pelo usuário	bom, regular ou ruim. $(0\% \rightarrow 100\%)$
Entrada	Percentual de cobertura da rede 4G nas áreas da cidade destacadas pelo usuário	alto, regular ou baixo. $(0\% \rightarrow 100\%)$
Entrada	Taxa de reclamações por acesso	alto, regular ou baixo. $(0 \rightarrow 1)$
Entrada	Custo do plano oferecido pela operadora	muito barato, barato, regular, caro ou muito caro. $R\$5,00 \rightarrow R\$500,00$
Entrada	Tamanho da franquia de dados oferecida pelo plano	alto, regular, baixo. $1GB \rightarrow 500GB$
Saída	Nota para a operadora	ruim, medíocre, mediano, bom ou muito bom. $(0 \rightarrow 10)$

Com os conjuntos difusos definidos, é possível agora obter funções de pertencimento, ou *Membership Functions*, para cada item da Tabela 1. Com essas funções obtemos entradas difusas, uma cobertura que seja um pouco boa e um pouco regular, por exemplo.

Para fazer uso dos dados de entrada "*fuzzyficados*", temos também um conjunto de regras que irá definir qual saída será ativada para uma determinada entrada:

1) **Regra 1**: Se as coberturas 4G e 3G da cidade e das áreas destacadas forem boas. Se o plano for barato, ou muito barato. Se o plano tiver uma alta franquia de dados e se a taxa de reclamações por acesso for baixa. Então, a nota da operadora será muito boa;
2) **Regra 2**: Se as coberturas 4G da cidade e das áreas destacadas forem boas. Se o plano for barato ou de custo regular. Se o plano tiver uma franquia de dados regular e se a taxa de reclamações por acesso for baixa. Então a nota da operadora será boa;
3) **Regra 3**: Se as coberturas 4G e 3G da cidade e das áreas destacadas forem medianas e se o plano tiver um custo regular, então a nota da operadora será regular. A nota da operadora também será regular caso o plano tenha um custo mediano, uma franquia de dados mediana e uma taxa de reclamações mediana;
4) **Regra 4**: Se as coberturas 4G e 3G da cidade **não** forem boas e se a cobertura 2G da cidade for boa, então a nota da operadora será medíocre. A nota também será medíocre se as coberturas 3G e 4G das áreas destacadas não forem boas e a cobertura 2G das áreas destacadas for boa.
5) **Regras 5**: Se a cobertura 4G ou 3G da cidade for ruim, ou se a taxa de reclamações for alta, ou se a cobertura 4G ou 3G das áreas destacadas for ruim, então a nota da operadora será ruim.

TABELA 2. Operadores lógicos equivalentes entre as lógicas booleana e fuzzy.

Lógica Booleana	Lógica Fuzzy
OR(X, Y)	MAX(X, Y)
AND(X, Y)	MIN(X,Y)
NOT(X)	1-X

Será explicado, posteriormente, como esse conjunto de regras foi obtido, bem como formas de aprimorar essas regras e tornar a saída do algoritmo mais factível com a realidade.

Na lógica *fuzzy*, substituímos os operadores booleanos de acordo com a relação mostrada na Tabela 3. Desta forma, cada regra tem uma saída que respeita o intervalo:

$$0 \leq x \leq 1$$

O resultado de cada regra define uma região na função de pertencimento da saída, como mostrado no exemplo da Figura 1, em que a reta vertical representa o centro de massa da região destacada, ou seja o valor escalar representado por essa região. O processo para obter um valor escalar que representa região obtida é chamado de *defuzzification*. Neste trabalho usamos o centro de massa da região como sendo este valor escalar. Desta forma, temos uma nota para cada operadora, notas estas que são baseadas na resposta das regras e funções de pertencimento de acordo com os dados de entrada.

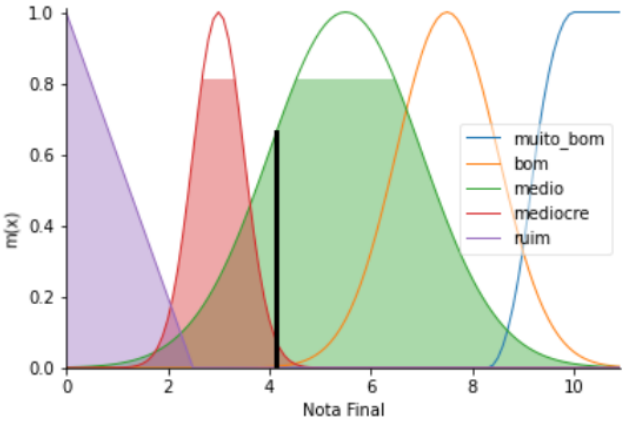

FIGURA 1. Região formada na função de pertencimento da saída.

C. DEFINIÇÃO DAS FUNÇÕES DE PERTENCIMENTO

As figuras a seguir mostram o comportamento de cada função de pertencimento presente no controlador *fuzzy*. No eixo Y, temos valores contínuos de 0 até 1, onde o valor 0 representa a ausência de pertencimento e o 1 representa o pertencimento por completo. Já no eixo X, temos o domínio de cada variável de entrada.

A Figura 2 mostra como foi modelada a função de pertencimento $m(x)$ para a variável "percentual de cobertura da rede". O eixo X varia de 0% até 100%, como indicado

na Tabela 3. Se a cobertura for de 100%, a única curva que apresentará um resultado diferente de 0 será a curva "bom", com o valor igual a 1, representando pertencimento máximo ao subconjunto "bom". Neste caso, temos que a cobertura é boa e não possui nenhuma componente nos subconjuntos "médio" e "ruim". Entretanto, se o valor da cobertura for de 70%, teremos componentes não nulas para os subconjuntos "médio" e "bom", cada subconjunto com um peso variando de 0 até 1.

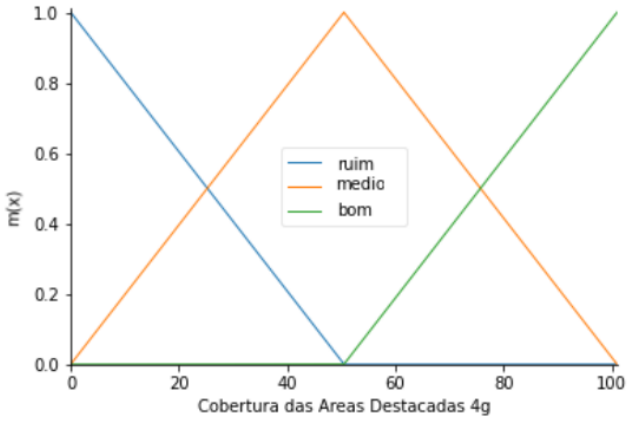

FIGURA 2. Função de pertencimento para o *Fuzzy Set* "Percentual de cobertura da rede 4G nas áreas da cidade destacadas pelo usuário".

Já a Figura 3 mostra a função de pertencimento para a variável "custo de um plano de telefonia celular". Diferente da função para o percentual de cobertura, as curvas dessa função são diversas, e não são somente triangulares. Usando curvas gaussianas podemos suavizar e relativizar ainda mais o nível de pertencimento de um dado, cujo comportamento será discutido posteriormente neste trabalho.

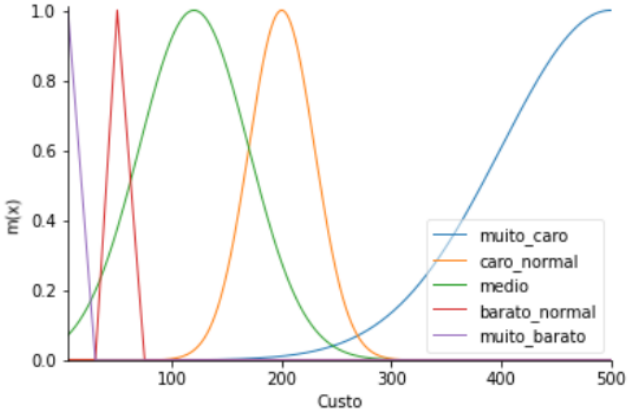

FIGURA 3. Função de pertencimento para o *Fuzzy Set* "Custo do plano oferecido pela operadora".

Para a variável "tamanho da franquia de dados oferecida pelo plano", temos a função de pertencimento $m(x)$ mostrada na Figura 4. Nesta função, o subconjunto "baixo" permanece totalmente ativado ($m(x) = 1$) no intervalo de 0 GB até 40 GB, que funciona quase como um pulso. Poderíamos transformar essa curva quadrada em uma condicional no código, como mostrada no pseudo-código abaixo:

```
1  if(0 <= franquia_de_dados.gigabytes <= 40):
2      franquia_de_dados = low()
```

É importante notar que como há uma intersecção entre as curvas "baixo" e "médio", então o pseudo-código mostrado já não seria suficiente. É justamente esse comportamento complexo que torna o uso da lógica *fuzzy* interessante. Evitamos milhares de condicionais e deixamos a manutenção do sistema bem mais agradável.

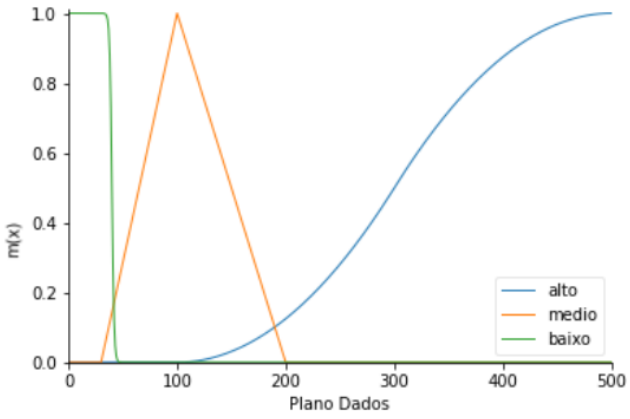

FIGURA 4. Função de pertencimento para o *Fuzzy Set* "Tamanho da franquia de dados oferecida pelo plano".

A variável "taxa reclamações por acesso" da Figura 5, como o nome sugere, representa a razão entre o número de reclamações de uma operadora em uma determinada cidade, dividido pelo número de chips ativos naquela cidade. Percebemos que se essa razão se aproxima de 1, temos uma taxa de reclamação muito ruim.

A curva "médio" é uma gaussiana, que representa a distribuição normal, com média e desvio padrão de alguns dados reais disponibilizados pela Anatel.

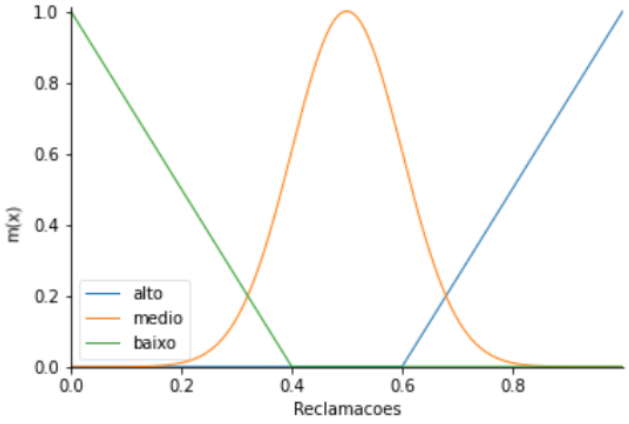

FIGURA 5. Função de pertencimento para o *Fuzzy Set* "Taxa de reclamações por acesso".

A saída do sistema é dada pela função ilustrada na Figura 6. As regras aplicadas formam uma área, que será a resposta *fuzzy* do sistema. Porém, como comentado anteriormente, usaremos o centro de massa da área obtida para ter uma resposta escalar. Desta forma, o escalar obtido será a nota recebida pela operadora em questão.

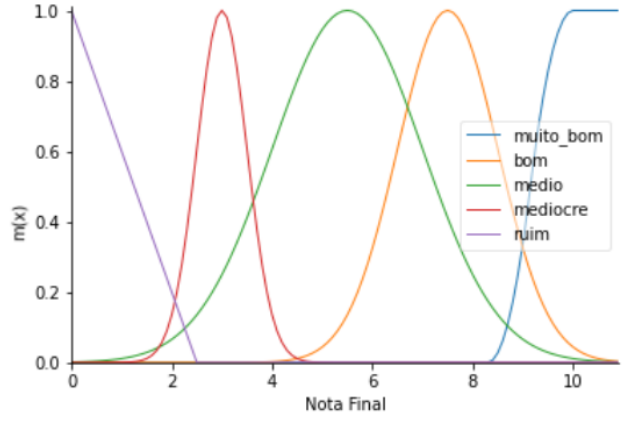

FIGURA 6. Função de pertencimento para o *Fuzzy Set* de saída "Nota para a operadora".

II. AQUISIÇÃO DOS DADOS

Todos os dados usados para alimentar a aplicação são reais. As informações de cobertura, reclamações e acessos SMP ("Serviço Móvel Pessoal") são advindas da política de dados abertos da Anatel [4] e foram todas inseridas em um banco de dados para serem consumidas em tempo real pela aplicação.

Infelizmente, não encontramos uma fonte de dados abertos sobre planos de cada operadora SMP, sendo assim, adicionamos manualmente preços e franquia de dados de planos similares de cada operadora para algumas cidades-chave, sendo elas Belo Horizonte, Ouro Branco e São João del-Rei, todas do estado de Minas Gerais.

É importante notar que as informações de percentual de área coberta são divididas em setores censitários. Para tornar a aplicação usável, inserimos um mapa no qual o usuário pode navegar e selecionar as regiões mais importantes. Usamos uma compilação de setores censitário do IBGE para obter os limites de cada setor censitário do país, separado por cidade [5].

Os arquivos obtidos diretamente do IBGE são do formato KML ("Keyhole Markup Language") e podem ser utilizados juntamente com a biblioteca do Google Maps. Entretanto, os arquivos KML não são facilmente manipuláveis em tempo de execução, tornando a tarefa de simplesmente colorir as regiões selecionadas pelo usuário extremamente árdua. Desta forma, usamos uma biblioteca chamada ToGeoJson para NodeJS [6] e convertemos todos os arquivos KML disponibilizados pelo IBGE para um formato chamado GeoJson, que é muito mais amigável para ser manipulado em tempo de execução pela API do Google Maps. Os arquivos foram convertidos e disponibilizados na internet, de modo que possam ser consumidos sob-demanda pela aplicação. Sem a conversão para GeoJson, os usuários teriam que informar o código do setor censitário manualmente, tornando a aplicação inviável em termos de experiência do usuário.

III. APLICAÇÃO WEB

Uma aplicação web foi criada para que o algoritmo seja acessível publicamente. Utilizando a API do Google Maps [7], podemos exibir as regiões da cidade selecionada e interagir com o mapa, colorindo as regiões escolhidas pelo usuário. Cruzando os dados do IBGE , que fornece o mapa com os limites das regiões das cidades, com os dados da Anatel, podemos buscar a cobertura das áreas destacadas.

Inserimos os dados provenientes da Anatel em um banco de dados, que é consumido por uma API (*"Application Programming Interface"*) construída em C#. Esta API é então disponibilizada aos usuários que acessam a GUI (*"Graphical User Interface"*) construída com Angular *Framework*. O código de ambas as aplicações podem ser acessados no endereço a seguir: https://github.com/VictorAndrade4/tcc.

Além disso, também é necessário expor o *script* feito em Python e a biblioteca *skfuzzy* na internet. Isso foi feito através de um serviço chamado "AWS *Lambda Functions*" [8], o qual nos possibilita expor a função que calcula a nota final de uma operadora na forma de um

endereço exposto na internet. O código desta aplicação também é público e pode ser acessado no link a seguir: https://github.com/VictorAndrade4/tcc-fuzzy-handler. Através desse repositório de código podemos ver todas os conjuntos difusos, regras e operações *fuzzy* sendo feitas com o auxílio da biblioteca *skfuzzy*, que facilita a aplicação e visualização de programas com lógica difusa [9].

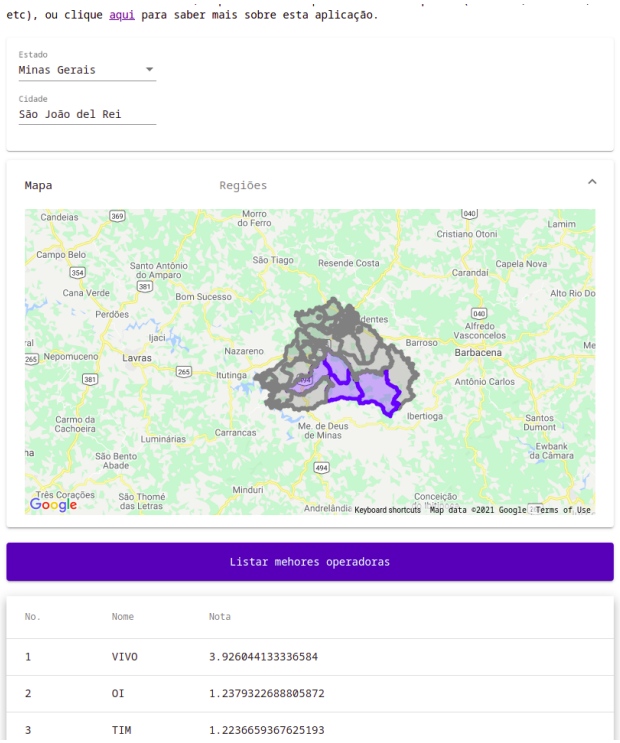

FIGURA 7. Captura de tela da interface gráfica aplicação para áreas da cidade de São João del-Rei.

IV. METODOLOGIA

Os parâmetros para a classificação influenciam muito no resultado final. Nesta sessão, discutimos mais sobre quais são esses parâmetros e seus impactos na tomada de decisão.

A. REGRAS DO ALGORITMO

Para construir o algoritmo *fuzzy*, responsável pela classificação, criaram-se as regras *fuzzy*. Essas regras podem ser definidas de diversas maneiras e usamos a forma mais simples delas, baseando no nosso senso de realidade. As regras atuais são extremamente simples, porém objetivas, o que torna a análise muito menos penosa. É claro que esse tipo de abordagem traz também muito viés pessoal, além de possíveis ruídos.

B. FUNÇÕES DE PERTENCIMENTO

De forma análoga às regras, as *membership functions* (funções de pertencimento) também foram definidas com base em experiências pessoais de forma arbitrária. Desta forma, estão extremamente sujeitas à ruídos e vieses. Estas funções foram e são ajustadas com o uso e de forma manual.

Também é interessante notar que algumas funções de pertencimento possuem curvas mais suavizadas, que relativizam essas funções ainda mais, tendo um significado de "geralmente pertence" [2].

Poderíamos também ter mais subconjuntos *fuzzy*, tornando nossos modelos mais confiáveis e reais. O percentual de cobertura, por exemplo, pode ser horrível, ruim, razoável ruim, mediano, razoável bom, bom e excelente, totalizando sete subconjuntos. Na verdade, o número de subconjuntos é indefinido para cada conjunto (*fuzzy set*) e, quanto maior o número de subconjuntos, maior será a definição do resultado final.

Entretanto, inserir mais subconjuntos *fuzzy* deixaria as regras mais complexas, além de aumentar a necessidades de recursos computacionais, deixando o sistema mais caro e complexo de se manter. A escolha do número de subconjuntos também foi feita de forma arbitrária, levando em conta a complexidade das regras e necessidade de cada *fuzzy set*.

C. DEFUZZIFICATION

Para obter o resultado final do algoritmo, usamos o centro de massa, como destacado visualmente na Figura 1. Poderíamos ter utilizado qualquer outra forma de *defuzzification*, já que a abordagem *fuzzy* nos dá essa liberdade. Para os cenários observados, o centro de massa se mostrou muito fiel, sendo uma ótima representação escalar dos planos *fuzzy* encontrados.

D. PLANOS DE CADA OPERADORA

As informações sobre os planos das operadoras não estão disponíveis na página de dados abertos da Anatel. Também não obtivemos respostas nas tentativas de contato com a Anatel sobre o tema. Como seria inviável inserir informações sobre todos os planos de todas as operadoras, para todas as cidades do Brasil, escolhemos um conjunto de cidades-chave (Belo Horizonte, Ouro Branco e São João del-Rei) e empregamos os planos medianos de cada operadora. Foi feita a média dos preços e franquia de dados para cada operadora e utilizamos esses dados para alimentar nosso controlador *fuzzy*.

V. RESULTADOS

Ao longo do desenvolvimento notamos e resolvemos o problema de várias operadoras obterem notas similares. O comportamento foi mudado ao ajustar as funções de pertencimento e as regras do controlador *fuzzy*. Agora, as operadoras tendem a obterem pesos distintos para as regras existentes, fazendo com que as notas se tornem mais diversas.

A. MAIOR NOTA POSSÍVEL

Na Figura 8 vemos a representação gráfica para a saída com a melhor entrada de dados possíveis:

1) Cobertura Total da Cidade 2G igual a 100%.
2) Cobertura Total da Cidade 3G igual a 100%.
3) Cobertura Total da Cidade 4G igual a 100%.
4) Cobertura das Áreas Destacadas 2G igual a 100%.

5) Cobertura das Áreas Destacadas 3G igual a 100%.
6) Cobertura das Áreas Destacadas 4G igual a 100%.
7) Custo do plano igual a 5 reais (menor valor possível).
8) Taxa de reclamações por acessos igual a 0.

Neste cenário, a nota obtida para a operadora fictícia foi de 9,5573 e não a nota 10, como deveria ser. Como se trata de um cenário controlado, já sabemos o resultado desejado, então esse foi um dos casos usados como controle durante o desenvolvimento. A diferença entre o valor encontrado pelo algoritmo e o valor real se dá pelas funções de pertencimento e às regras criadas.

Esse comportamento acontece porque, apesar da regra que define o resultado ótimo ser igual a 1 (valor máximo), outras regras também são ativadas, causando a nota a ser um pouco menor do que o ideal. É um dos problemas de se tratar regras e funções de pertencimento de forma manual. Quanto mais regras complexas e funções diversas, mais difícil é prever o comportamento do sistema.

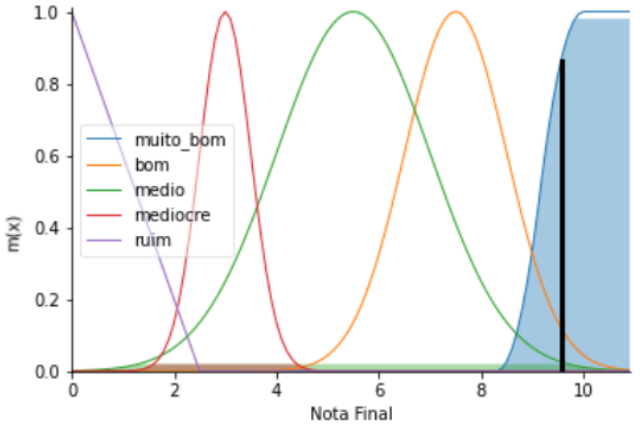

FIGURA 8. Resultado do algoritmo *fuzzy* para o melhor cenário. Nota igual a 9,5573

B. MENOR NOTA POSSÍVEL

De forma análoga, temos a menor nota possível, com entradas opostas às selecionadas para o maior resultado possível. A Figura 9 mostra a representação gráfica da saída deste cenário, obtendo a nota de 0,8333. Neste caso podemos ver que apenas a região "*poor*" ou "ruim" é ativada. O problema, neste caso, é que como representamos a região pelo centro de massa, a nota nunca será igual 0, a menos que essa região seja infinitesimalmente pequena. Neste caso, poderíamos usar mais subconjuntos *fuzzy* para aproximar ainda mais esse valor de 0.

C. ALGUNS CENÁRIOS REAIS

Através do estado e cidade informados pelo usuário, colocamos um mapa com os setores censitários daquela cidade para que o usuário selecione as regiões que considera mais importantes. Com as regiões informadas, usamos nossa API para buscar todos os parâmetros necessários para o algoritmo, no nosso banco de dados. A API aciona o controlador *fuzzy*

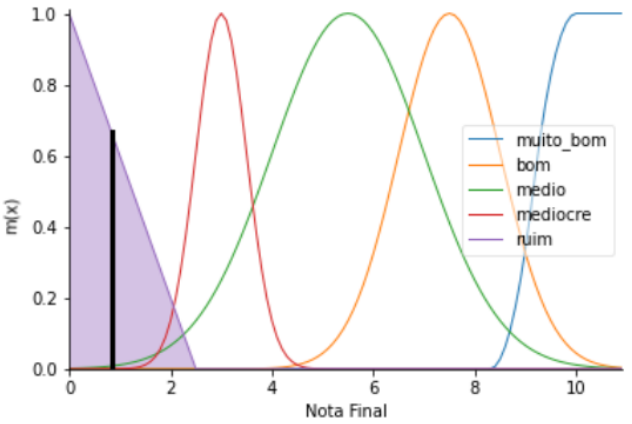

FIGURA 9. Resultado do algoritmo *fuzzy* para o pior cenário. Nota igual a 0,833333

escrito em Python e retorna os resultados para a interface gráfica.

A aplicação, disponível em classificadoroperadoras.com.br, foi utilizada para realizar os testes mostrados nas figuras abaixo. O resultado se mostra bastante factível, com notas maiores para as operadoras em Belo Horizonte (Figura 10), que conta com maior infraestrutura de telefonia móvel se comparado com Ouro Branco (Figura 11).

Na Tabela 3, podemos observar a razão entre as notas das áreas selecionadas em Belo Horizonte e notas para áreas selecionadas em Ouro Branco. Neste cenário, as notas de Belo Horizonte foram, em média, 1,78715 vezes maior que as notas obtidas em Ouro Branco.

TABELA 3. Comparativo entre as notas obtidas para as mesmas operadoras Ouro Branco e Belo Horizonte.

Cidade	Claro	OI	TIM	Vivo
Belo Horizonte	8,9144	5,4515	8,7578	8,5392
Ouro Branco	4,9650	3,0000	4,8217	5,4152
Razão	1,7954	1,8171	1,8163	1,7198

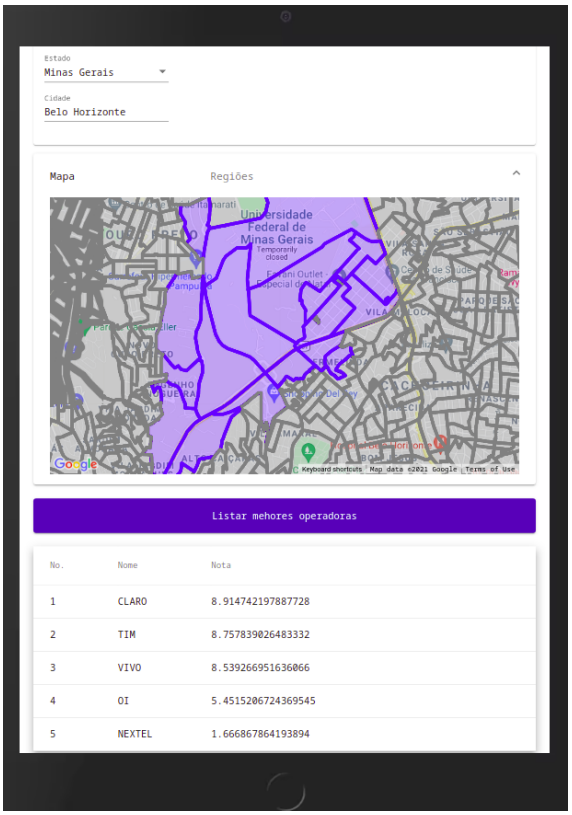

FIGURA 10. Usando a aplicação em um cenário real para Belo Horizonte, Minas Gerais.

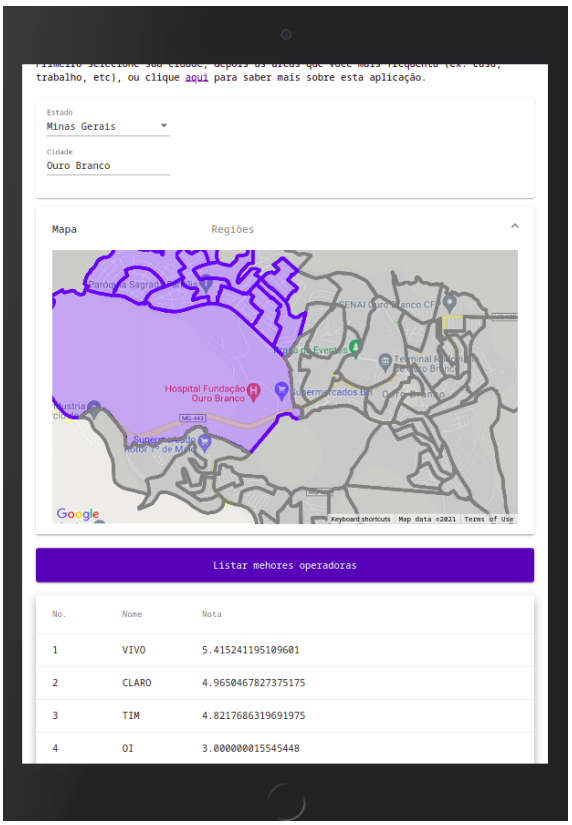

FIGURA 11. Usando a aplicação em um cenário real para Ouro Branco, Minas Gerais.

VI. CONCLUSÃO

Controladores *fuzzy* são muito adaptáveis, nos dando graus de liberdade para adaptar o controle à necessidade real, tão flexível e subjetivo quanto precisamos para resolver problemas complexos, flexíveis e subjetivos.

Como dito anteriormente, a definição das regras está feita de maneira simples, desta forma, ainda tem-se muito espaço para melhorar a classificação feita hoje. Como a lógica *fuzzy*, por si só, não tem nenhum mecanismo para aprender, poderíamos usar algoritmos evolucionários para otimizar as regras do sistema. Uma outra possível melhoria na modelagem seria a construção das funções de pertencimento, ou *membership*, com maior apoio à estatística e técnicas de *big data*.

Com métodos mais eficientes de se modelar as regras do sistema, poderíamos obter mais conjuntos *fuzzy*, como diferenciar tipos de reclamações diferentes e atribuir diferentes pesos à elas. Também poderíamos criar adaptar as funções

Por fim, para tornar a aplicação mais escalável e mais robusta, é necessário um zelo maior com o armazenamento dos arquivos GeoJson, responsáveis por carregar o mapa das cidades selecionadas. Implementar um sistema de cache poderia resolver vários problemas de lentidão, nesse sentido. Melhorar também a modelagem do banco de dados, para evitar consultas em tabelas excessivamente longas. Além de criar rotinas automáticas, que visitariam os repositórios de dados da Anatel e IBGE para inserir dados atualizados no banco de dados a cada período de tempo.

REFERÊNCIAS

[1] Anatel, “Telefonia móvel - serviço móvel pessoal (smp) - 2º semestre de 2020,” 2020. [Online]. Disponível em: https://bit.ly/3pdZujN

[2] L. Zadeh, “Fuzzy logic,” 1988. [Online]. Disponível em: https://ieeexplore.ieee.org/abstract/document/53

[3] L. Zadeh, “Fuzzy sets,” 1979. [Online]. Disponível em: https://www.worldscientific.com/doi/abs/10.1142/9789814261302_0021

[4] Anatel, “Plano de dados abertos da anatel.” [Online]. Disponível em: https://www.gov.br/anatel/pt-br/dados/dados-abertos

[5] IBGE, “Malha de setores censitários.” [Online]. Disponível em: https://www.ibge.gov.br/geociencias/organizacao-do-territorio/malhas-territoriais/26565-malhas-de-setores-censitarios-divisoes-intramunicipais.html?=&t=o-que-e

[6] “Convert kml and gpx to geojson.” [Online]. Disponível em: https://github.com/mapbox/togeojson

[7] Google, “Google maps platform documentation.” [Online]. Disponível em: https://developers.google.com/maps/documentation

[8] AWS, “Aws lambda.” [Online]. Disponível em: https://aws.amazon.com/pt/lambda/

[9] “Skfuzzy documentation.” [Online]. Disponível em: https://pythonhosted.org/scikit-fuzzy/