Online Monitoring of Buses Information Using KNN, ATR and DMC Algorithms

J. Sá, V. Castro, E. Coelho, and F. Farias

Abstract— **In developing countries such as Brazil, it is still hard to know what time a bus will reach a bus stop. This means that a lot of people have to wait for long periods of time without knowing whether the bus is near or far away, and it raises serious problems of traffic mobility that affect millions of public transport users every day. One way out is to seek solutions based on software and artificial intelligence that can help the public when making their everyday journey. However, most of them still need to be approved by private companies or the local government. This article sets out an online monitoring strategy for bus routes on a collaborative basis, i.e., among the users, as an alternative system that does not need to be authorized by private companies of the local government. It involves adopting the K-Nearest Neighbors algorithm and creating two new algorithms called Activation of the Regressive Timer and Deactivating Collaborative Mode. The algorithms were implemented in an application called "Olha o Ônibus", in English Tracking the Buses, and five tests were conducted to test the effectiveness of the scheme. The results showed that the algorithms are able to ensure the reliability of the information that is shared among the users.**

Index Terms—**K-nearest neighbors, Artificial intelligence, Online monitoring, Bus routes.**

I. Introdução

PROBLEMAS relacionados à mobilidade urbana podem afetar na qualidade de vida da população nos centros urbanos. Os fatores que podem contribuir com a baixa qualidade da mobilidade urbana estão relacionados ao transporte público ruim, como o péssimo estado de conservação dos veículos e ausência de informações para os usuários [1]. A indisponibilidade de informações sobre as rotas, tempo de chegada e atrasos são problemas que tornam o sistema de transporte público coletivo pouco atrativo para muitos usuários [2].

Como forma de buscar soluções alternativas para os problemas, diversos desenvolvedores de *softwares* têm apresentado sistemas que visam auxiliar usuários do transporte público durante o processo de mobilidade urbana. Neste contexto, soluções como Moovit [3] e Cadê o Ônibus? [4], buscam disponibilizar informações sobre linhas de ônibus como horários, itinerários, previsão de chegada, informações sobre o trânsito e em alguns casos a localização dos veículos de forma *online*.

Entre os *softwares* citados, alguns permitem que os usuários colaborarem com informações sobre os coletivos. No entanto, nenhum destes permite a colaboração através do envio da localização *online* dos ônibus para outros usuários.

Na literatura existem diversos trabalhos que visam solucionar problemas de rotas de transportes urbanos, por exemplo, em [5] os autores propõem um sistema denominado de *Security System For Bus* (SSBus), composto por um sistema *web* e dois *softwares* para plataformas móveis, denominados SSBus Cliente e SSBus Ônibus, que tem por objetivo fazer a colaboração da localização e do estado de segurança dos ônibus, além disso, no trabalho são apresentados resultados fundamentados em experimentos que avaliam as respostas do sistema usando métricas de avaliação de precisão que comprovam que os *softwares* propostos são eficientes na tarefa de informar a localização e *status* de segurança dos veículos. Já em [6], os autores apresentam um aplicativo móvel denominado de TrafficInfo que é parte de um sistema que transmite informações de ônibus de forma colaborativa entre usuários. Além disso, este sistema é composto por uma solução que monitora eventos de parada do veículo. A solução utiliza o algoritmo de árvore de decisão *J48*, o qual é alimentado por dados provenientes de recursos do *smartphon*e como *Global Positioning System* (GPS), *WiFi* e acelerômetro.

Em [7], foi proposto a utilização de Inteligência Artificial (IA) para aumentar a precisão de informações dos ônibus, onde os autores desenvolveram e apresentaram a utilização de uma Rede Neural Artificial (RNA) aprimorada para realizar previsões de tempo de chegada do ônibus. Neste estudo, foram feitas simulações com 24 ônibus, e a análise de confiabilidade mostrou resultados precisos da utilização do modelo. Já em [8], os autores apresentam um modelo matemático baseado em algoritmos de predição com o intuito de obter a previsão do fluxo de tráfegos de veículos em rodovias. Comprovou-se a viabilidade do modelo através de experimentos, onde foram obtidas informações precisas de taxas de fluxo de tráfego.

Os autores em [9] apresentaram e analisaram algoritmos de busca *Anytime* na plataforma Raspberry Pi com o objetivo de planejar rotas otimizadas de redes veiculares. Experimentos foram feitos e os resultados obtidos são considerados satisfatórios em relação ao desempenho do algoritmo proposto e da viabilidade da utilização do Raspberry Pi como um sistema embarcado para o planejamento de rotas. Em [10] os autores propuseram e compararam três algoritmos, um baseado em *branch-and-bound*, outro em programação *mixed-integer* e o

J. S. Sá, Universidade Federal do Pará, Cametá, Pará, Brasil (e-mail: joinersa@hotmail.com).

V. C. Castro, Universidade Federal do Pará, Cametá, Pará, Brasil (e-mail: vicente_soad@yahoo.com.br).

E. M. Coelho, Universidade Federal do Pará, Cametá, Pará, Brasil (e-mail: eliezermirandacoelho@hotmail.com).

F. S. Farias, Universidade Federal do Pará, Cametá, Pará, Brasil (e-mail: fabriciosf@ufpa.br).

último em arvore cinética, para gerenciamento de solicitações de compartilhamento de viagens em uma rede rodoviária. Em experimentos feitos com vários dados de taxis, o algoritmo de árvore cinética superou os outros dois algoritmos propostos e outras abordagens que eram comumente usadas.

Em [11] foi proposto um modelo que prevê o tempo de chegada de ônibus usando dados de um *Automatic Vehicle Location* (AVL). Testes com dados reais de uma rota de ônibus foram realizados em três diferentes modelos de algoritmos, um baseado em dados históricos, outro em regressão e o último em RNA. Como resultado, constatou-se que o modelo de RNA superou os outros modelos em termos de precisão da previsão. Já, os autores em [12] propuseram uma metodologia para formular e resolver o problema de projeto de redes de trânsito de ônibus referente ao tempo aleatório de viagens. Neste trabalho, como solução do problema, foi formulado um modelo de solução heurístico baseado nos algoritmos *K-Shortest Path, Simulated Annealing, Monte Carlo Simulation* e *Probit-Type Discrete Choice Model*, e para mostrar a viabilidade da proposta, experimentos foram realizados aplicando o modelo proposto à rede de localidades da Suiça. Os resultados demonstraram que esta solução supera métodos anteriores relatados na literatura.

Os autores em [13] propuseram o uso de um algoritmo genético como ferramenta para lidar com a complexidade do problema de projeto de redes de ônibus. Experimentos foram realizados em um estudo de caso de uma rede real de larga escala na cidade de Mashhad, no Irã. O método proposto alcançou resultados melhores do que outras redes projetadas e relatadas em outros estudos que apresentaram resultados relacionados a medidas de desempenho e função objetivo. Já no trabalho de [14], foi proposto um sistema de rastreamento inteligente de ônibus, onde qualquer usuário pode visualizar os tempos estimados de chegada, localização atual e as rotas dos ônibus em um mapa. No sistema proposto, foi utilizado o algoritmo C4.5 para estimar os tempos de chegada dos ônibus e minimizar o tempo de espera dos passageiros. Como resultado, foi apresentado um *software* robusto que faz o uso de tecnologias como GPS, *Global System for Mobile Communications* (GSM), codificação *Quick Response* (QR) e recursos dos mapas do Google.

Em [15], os autores propuseram um modelo dinâmico de previsão de tempo de viagem para ônibus abordando casos em estrada com múltiplas rotas de ônibus, baseado em Máquina de Vetores de Suporte (SVM) e algoritmo baseado em Filtro de Kalman. O SVM bem treinado prevê os tempos de viagem de ônibus da linha de base a partir dos dados históricos da viagem. O algoritmo dinâmico baseado em filtro de Kalman ajusta o tempo de viagem com as últimas informações sobre operações de ônibus e estima os tempos de viagem das linhas. Para validar o modelo proposto, foram realizados experimentos com dados do mundo real em uma estrada com múltiplas rotas de ônibus em Shenzhen na China. Os resultados mostram que o modelo é viável e aplicável para predição do tempo de viagem de ônibus, além disso, o modelo tem o melhor desempenho entre cinco modelos apresentados no estudo, em termos de precisão de previsão em estradas com múltiplas rotas de ônibus. Já em [16], foi apresentado um algoritmo não paramétrico que proporciona previsões precisas de tempo de chegada dos ônibus nos pontos de parada, com base em medições de dados de GPS em tempo real, essa ideia usa o modelo de Regressão de Kernel. O algoritmo proposto foi avaliado através de dados reais de frotas de ônibus de Dublin, na Irlanda. Através dos experimentos, constatou-se que a proposta supera os métodos paramétricos.

Os autores em [17], propuseram um sistema embarcado de rastreamento de objetos utilizando componentes de baixo custo. Para demonstrar a eficácia do projeto, os autores construíram um protótipo do sistema proposto e um *software* rudimentar para demonstrar o rastreamento do sistema embarcado. Através dos testes, a proposta foi considerada útil para inúmeras aplicações de rastreamento de objetos.

Em [18] foi proposto um sistema de rastreamento de ônibus em tempo real. A proposta apresenta ao usuário final, através de *smarphone* Android e de uma página *web*, a localização dos veículos de forma precisa e em tempo real. O projeto utilizou GPS, *Radio Frequency Identification* (RFID) e GSM para acessar a localização do ônibus em tempo real. Testes de simulação foram realizados e resultados mostraram a eficiência da proposta. O sistema economiza tempo e aumenta a eficiência, pois reduz os esforços do usuário em viagens e evita o desperdício de tempo de espera dos ônibus.

Os trabalhos apresentados se propuseram a utilizar *softwares* e algoritmos, em sua maioria de IA, como abordagem para definição de rotas otimizadas, previsão do tempo de chegada de ônibus, previsão do fluxo de tráfego de veículos, compartilhamento de viagens, detecção de eventos de parada e colaboração da localização do veículo. No entanto, os trabalhos apresentados se concentram no uso de dados fornecidos por equipamentos instalados nos ônibus [5][9][14][17][18], ou utilizam algoritmos complexos que necessitam de uma etapa de treinamento que limita a expansão para novas rotas sem a prévia realização de novas simulações [6][7][11][15], ou estão baseados em modelos de otimização que limitam-se aos cenários sob investigação [8][9][10][12][13], ou ainda não realizam o tratamento de informações falsas enviadas por parte de usuários mal-intencionados [6]. Diante deste contexto, este artigo tem como objetivo principal propor e testar uma solução para o monitoramento *online* e colaborativo de rotas de ônibus usando o algoritmo de IA dos *K-Nearest Neighbors* (KNN) [19], em conjunto com dois novos algoritmos denominados de Ativador do Temporizador Regressivo (ATR) e Desativação do Modo Colaborativo (DMC). Esta abordagem também tem como objetivo garantir a confiabilidade da informação fornecida por qualquer usuário que deseja colaborar a rota do ônibus enviando sua geolocalização, isto é, corrigindo erros provocados por imprecisão de GPS ou por usuários mal-intencionados. Para validação, os algoritmos foram implementados, em um *software* denominado de Olha o Ônibus (OoO) e testados através de cinco testes de campo.

O restante do trabalho é organizado nas seguintes seções. A seção II expõe os detalhes sobre os algoritmos e modelos matemáticos propostos. A seção III apresenta a implementação dos algoritmos propostos, no *software* OoO. A seção IV explica o caso de estudo deste trabalho. A seção V apresenta os testes e

resultados da utilização dos algoritmos propostos. Por fim, a seção VI apresenta as conclusões.

II. Algoritmos e Modelagem Matemática

Esta seção apresenta os algoritmos e seus modelos matemáticos que visam garantir a confiabilidade das informações sobre rotas de ônibus que são colaboradas entre usuários. O primeiro o algoritmo apresentado é o KNN adaptado para o problema; em seguida, é apresentado o algoritmo responsável pela ativação do temporizador regressivo; por fim é apresentado o algoritmo responsável em acionar de modo automático a desativação do modo colaborativo.

A. Algoritmo K-Nearest Neighbors

O algoritmo KNN é utilizado para classificação de objetos e seu funcionamento está relacionado a determinação dos k vizinhos mais próximos de um dado objeto para fins de classificação [19]. Essa abordagem possui três elementos-chave: um conjunto de registros rotulados na base de treinamento; uma métrica para calcular a distância entre registros; e a quantidade de vizinhos mais próximos, o k. Para classificação de um novo registro não rotulado, são calculadas as distâncias entre o novo registro e os registros rotulados da base de treinamento, os k vizinhos mais próximos ao novo registro serão identificados e seus rótulos servirão de parâmetro para rotular o novo elemento [20]. Em uma abordagem mais simples, o novo registro é classificado sendo comparado ao mais relevante entre seus k vizinhos mais próximos.

Neste trabalho, uma adaptação do algoritmo KNN é usada para classificar e verificar a distância entre pontos geográficos referentes a rota do ônibus e o ponto fornecido pelo usuário.

O funcionamento matemático do algoritmo baseia-se na seleção de determinado ponto da rota que representa a menor distância (1-NN) para o ponto geográfico enviado pelo usuário. No uso de k = 1, o KNN determina que o ponto com a menor distância representa a classe de saída ótima.

Para isso, uma base de dados de treinamento composta pelos atributos nome da rota, latitude e longitude é apresentada ao algoritmo. O número de registros armazenados varia de acordo com o tamanho da rota cadastrada, quantidade de rotas cadastradas, e número de pontos geográficos de latitude e longitude armazenados. Deste modo, podemos organizar a base de dados de treinamento de determinada rota como um vetor (V_r) formado por um conjunto de registros, isto é, $V_r = \{V_{r1}(\varphi_{r1}, \lambda_{r1}), V_{r2}(\varphi_{r2}, \lambda_{r2}), \dots, V_{rn}(\varphi_{rn}, \lambda_{rn})\}$, sendo este vetor composto por pares ordenados contendo a latitude do ponto da rota (φ_r) e longitude do ponto da rota (λ_r) que representam de modo discreto determinado ponto da rota $V_r(\varphi_r, \lambda_r)$ a ser percorrida pelo ônibus.

Ao selecionar uma rota para colaboração, o usuário recebe do GPS a sua localização geográfica que por sua vez é apresentada ao algoritmo KNN para correção da localização capturada. Durante a execução do KNN, os pontos de latitude e longitude obtidos pelo usuário são comparados aos pontos armazenados na base de dados de treinamento de uma rota pré-definida, em seguida, o algoritmo classifica o ponto com menor distância. Para este caso, os registros que dão suporte a classificação são compostos por atributos de latitude (φ_u) e longitude (λ_u) do usuário, isto é, $P_u(\varphi_u, \lambda_u)$, os quais são coletados e comparados com pontos da rota V_r. Na etapa de classificação, as coordenadas $\varphi_u, \varphi_r, \lambda_u$ e λ_r são representadas em unidade de radianos. Após a realização dos cálculos, os pontos originais e corrigidos são enviados para armazenamento no servidor de banco de dados. Por fim, a localização do ônibus corrigida pelo KNN passa a ser disponibilizada para os demais usuários.

Dentre as formas de realização dos cálculos, a distância Euclidiana se destaca como uma das formas mais comuns empregadas no KNN, porém esta pode não ser sempre melhor solução para todos os casos [21]. Para medir distâncias geográficas, a fórmula de Haversine é uma solução viável, pois com ela é possível calcular a distância d entre dois pontos na superfície terrestre a partir dos atributos de latitude e longitude [22]. Esta é a solução empregada no algoritmo KNN para este trabalho. A equação (1) representa o modelo matemático de Haversine.

$$d(V_r(\varphi_r, \lambda_r), P_u(\varphi_u, \lambda_u)) = 2.R.\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_u - \varphi_r}{2}\right) + \cos(\varphi_r).\cos(\varphi_u).\sin^2\left(\frac{\lambda_u - \lambda_r}{2}\right)}\right) \quad (1)$$

onde d e R são respectivamente a distância entre dois pontos e o raio do planeta terra.

A equação (2) representa o modelo matemático de classificação de um ponto com o KNN:

$$P_{u\_novo} = V_{r\left(\underset{x \in [1,n]}{\operatorname{argmin}}\, d(V_{rx}(\varphi_{rx}, \lambda_{rx}), P_u(\varphi_u, \lambda_u))\right)} \quad (2)$$

onde P_{u_novo}, n, x e argmin são respectivamente o novo ponto ajustado do usuário, a quantidade de pontos de uma rota V_r, a posição de um ponto na rota V_r, e o valor de x para o menor valor da função de distância d. Esta representação classifica o ponto do usuário P_{u_novo} com o valor do ponto da rota V_r na posição x. Neste caso, o ponto P_{u_novo} é sempre atualizado em função do valor do ponto $V_{r(x)}$.

B. Algoritmos de Ativação do Temporizador Regressivo e o de Desativação do Modo Colaborativo

Os algoritmos ATR e DMC tem o objetivo de garantir que usuários colaboradores compartilhem informações confiáveis da geolocalização de ônibus.

As métricas de colaboração da geolocalização de ônibus são dependentes: da velocidade do colaborador, da distância entre o colaborador e rota, e do tempo em que o agente colaborador pode se manter em algum ponto fora da rota prevista, ou seja, sendo considerado como um estado de erro. Para este modelo, o tempo é definido em minutos, a distância em metros e a velocidade em quilômetros por hora (km/h). A equação (3) representa a métrica ATR que é acionada pelo estado de erro.

$$M_1(v,d) = \begin{cases} -1, v \geq v_{min} \; e \; d \leq d_{max} \\ 0, v < v_{min} \; e \; d \leq d_{max} \\ 1, v < v_{min} \; e \; d > d_{max} \\ 2, v \geq v_{min} \; e \; d > d_{max} \end{cases} \quad (3)$$

onde v, v_{min}, d e d_{max} são respectivamente a velocidade do colaborador, a velocidade mínima para colaboração, a menor distância entre o colaborador e a rota, e a distância máxima de colaboração (distância entre o colaborador e a rota).

O usuário colaborador só pode acionar o estado de erro quando $M_1 \neq -1$. Se $v < v_{min}$ e $d \leq d_{max}$, logo $M_1 = 0$ e o temporizador para o desligamento do modo colaborativo será iniciado com uma contagem decrescente de τ_1 minutos. Caso $v < v_{min}$ e $d > d_{max}$, logo $M_1 = 1$ e o temporizador será habilitado com uma contagem regressiva de τ_2 minutos. Se $v \geq v_{min}$ e $d > d_{max}$, logo $M_1 = 2$ e o temporizador será ativado com uma contagem decrescente de τ_3 minutos. Já a métrica DMC pode ser representada pela equação (4):

$$M_2(t_a) = \begin{cases} 0, t_a = 0 \\ 1, t_a > 0 \end{cases} \quad (4)$$

onde t_a é o valor atualizado do temporizador regressivo.

Quando $t_a = 0$, logo M_2 = 0 e o usuário é desabilitado do modo colaborativo. Caso o usuário retorne uma $v \geq v_{min}$ e uma $d \leq d_{max}$ durante o período de decréscimo do tempo, a variável M_1 retornará para o valor $M_1 = -1$ e o temporizador regressivo será desabilitado.

A Fig. 1 ilustra na forma de fluxograma o funcionamento dos algoritmos ATR, em Fig. 1 (a), e DMC, em Fig. 1 (b). A representação em fluxograma é uma forma objetiva para entendimento das métricas matemáticas propostas para operação e controle dos algoritmos.

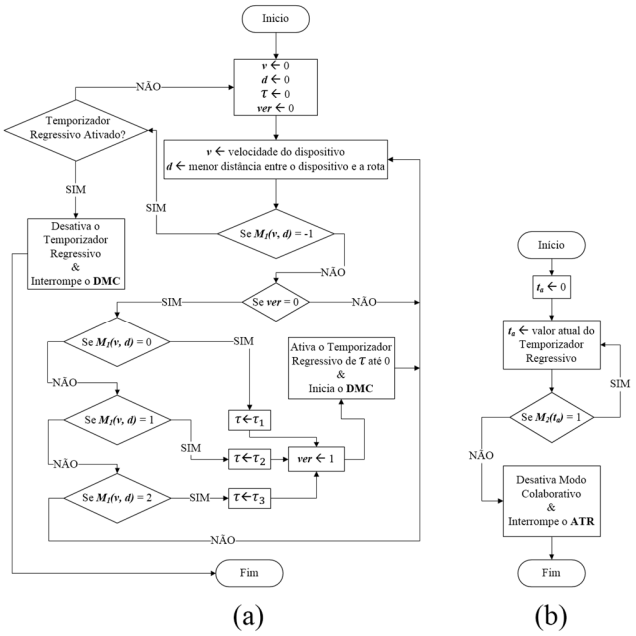


Fig. 1. (a) Fluxograma do algoritmo de Ativação do Temporizador Regressivo. (b) Fluxograma do algoritmo de Desativação do Modo Colaborativo.

A Fig. 1 (a) representa o algoritmo ATR, o fluxo inicia e segue para o processo de criação das variáveis e atribuição de seus valores iniciais, em seguida atribui-se valores de entrada às variáveis de velocidade v e distância d, o passo seguinte é verificar a primeira estrutura de decisão, se a condição for satisfeita para $M_1 = -1$, o fluxo segue para outra estrutura de decisão onde é verificado se o temporizador regressivo está ativado, se sim, ele será desativado, o funcionamento do DMC será interrompido e o fluxo atual será encerrado, senão o fluxo retorna para o processo de criação de variáveis e atribuição de seus valores. Ainda na primeira estrutura de decisão, caso $M_1 \neq -1$ o usuário estará se comportando em estado de erro durante o ato de colaborar, logo o fluxo avança para verificação da variável ver, se $ver \neq 0$ então será executado novamente o processo de atribuição de novos valores às variáveis v e d, caso contrário a próxima estrutura condicional será verificada. Se $M_1 = 0$, então serão executados os processos que definem os valores de τ e ver. A partir destes processos, será ativado o temporizador regressivo por um período τ e será iniciado de forma síncrona o algoritmo DMC descrito na Fig. 1 (b), o qual executará em paralelo ao algoritmo ATR que seguirá efetuando repetições de verificações. Caso $M_1 \neq 0$, outras estruturas de decisão serão verificadas e caso sejam satisfeitas, o cronômetro será ativado com um outro valor de τ.

Para iniciar a colaboração, os usuários acessam a funcionalidade "Colaborar" e selecionam o ônibus que desejam compartilhar a localização. Neste momento, o algoritmo ATR é inicializado e passa para o estado ativo e cíclico, onde realiza a verificação em tempo real das condições de velocidade e distância do usuário colaborador. Quando o usuário decide finalizar a colaboração ou o algoritmo identifica inconsistências nas informações fornecidas pelos usuários, o algoritmo ATR é automaticamente finalizado e o modo colaborativo é desligado.

Deste modo, se no ciclo atual o temporizador já estiver ativado, a variável ver garantirá que o temporizador não seja iniciado novamente com a mesma ou com outra condição em que se foi iniciado em algum ciclo anterior.

A Fig. 1 (b) representa o fluxograma do algoritmo DMC. Este algoritmo tem início com o processo de criação e inicialização da variável t_a. No processo seguinte a variável t_a é atualizada com o valor atual do temporizador regressivo (o qual foi ativado no ATR), o próximo passo é verificar a condição da função M_2, caso $M_2 = 1$, o fluxo retorna para o processo de atualização de t_a, caso contrário o modo colaborativo será desativado, o algoritmo ATR será interrompido e o usuário deixará de colaborar.

Como os algoritmos propostos são dependentes um do outro, pois trocam dados de forma síncrona, o ATR pode encerrar a execução do algoritmo DMC (se $M_1 = -1$), mesmo antes que o contador regressivo seja finalizado e o modo colaborativo seja desativado, assim como o DMC pode interromper a execução do ATR.

III. Implementação dos Algoritmos Propostos

Os algoritmos apresentados na seção II, foram implementados em um *software,* para sistemas operacionais

Android, denominado de Olha o Ônibus (OoO). O aplicativo OoO tem como objetivo possibilitar a colaboração *online* da geolocalização de rotas de ônibus entre usuários.

A Fig. 2 representa a arquitetura de funcionamento do *software* OoO, onde o usuário colaborador envia ao servidor as coordenadas do ônibus ajustadas pelo KNN e os usuários finais que são favorecidos pela colaboração recebem do servidor a localização do ônibus de modo *online*.

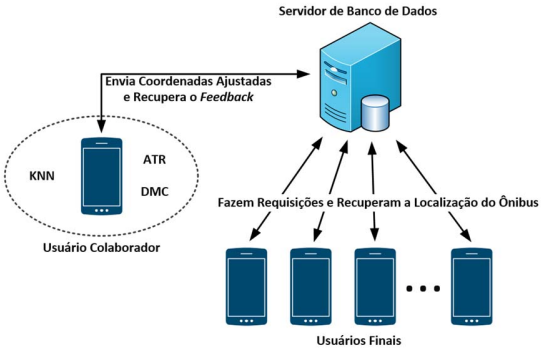
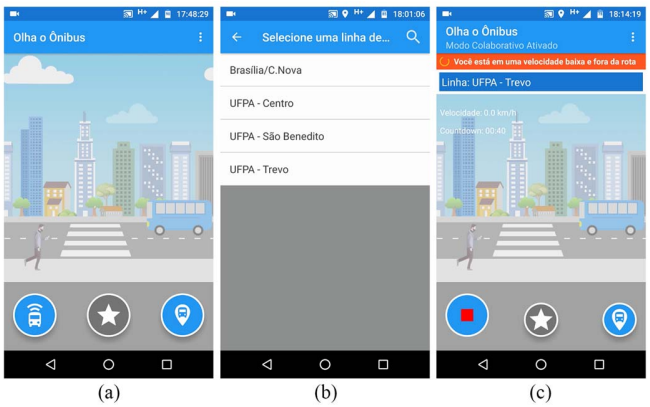


Fig. 2. Arquitetura de funcionamento do *software* OoO.

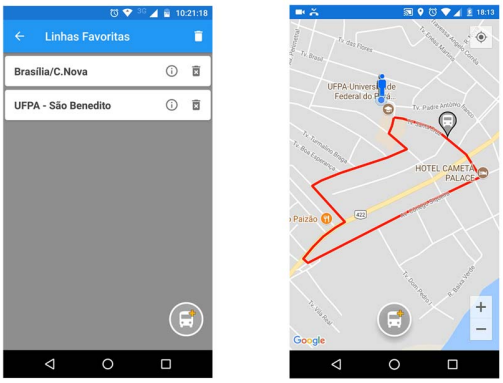
A Fig. 3 apresenta três telas do *software,* onde a Fig. 3 (a) representa a tela principal do aplicativo, nesta, estão os botões para o acesso à ativação do modo colaborativo, linhas de ônibus favoritas e mapa de monitoramento. A Fig. 3 (b) representa a tela onde estão as linhas de ônibus disponíveis para serem colaboradas por usuários. Já a Fig. 3 (c), é a representação da tela que é exibida ao usuário que está colaborando, nesta, estão os botões para interrupção da colaboração, visualização de linhas de ônibus favoritas e mapa de monitoramento, além de informações de *status*, nome da linha colaborada, velocidade e o contador regressivo.



(a) (b) (c)

Fig. 3. Telas de apresentação do software OoO. (a) Tela principal do aplicativo. (b) Tela de linhas de ônibus disponíveis para colaboradores. (c) Tela do modo colaborativo.

O *software* OoO dá suporte ao monitoramento *online* dos ônibus colaborados através de um mapa. As linhas monitoradas podem ser guardadas em uma lista de favoritos no aplicativo. A Fig. 4 apresenta duas telas de funções do *software,* onde a Fig. 4 (a) representa a tela de gerenciamento de linhas de ônibus favoritas. Já a Fig. 4 (b) representa a tela do mapa onde contém a rota e os marcadores de localização do usuário e ônibus.



(a) (b)

Fig. 4. Telas de apresentação de funções do *software* OoO. (a) Tela de linhas de ônibus favoritas. (b) Tela de mapa de monitoramento de ônibus.

IV. Caso de Estudo

Essa seção apresenta informações de rotas e parâmetros de configuração dos algoritmos e modelos propostos na seção II. Neste estudo, foram definidas duas rotas, R1 e R2, para os testes realizados na cidade de Belém-Pará. A Fig. 5 representa as trajetórias de R1 e R2, que são respectivamente rotas das linhas de ônibus UFPA – Alcindo Cacela e UFPA – Centro Histórico [23]. Em R1 foram definidos 1069 pontos de pares ordenados do vetor V_r que alimentam a base de dados que representa a rota R1. Os pontos de R1 estão distribuídos em uma rota com 19,40 km de comprimento, a qual compreende aos bairros Guamá, Condor, Cremação, São Brás, Reduto, Campina, Jurunas, Batista Campos e Umarizal. Já para rota R2, foram definidos 599 pontos de pares ordenados do vetor V_r que alimentam a base de dados a ser percorrida. Os pontos de R2 estão distribuídos em 15,77 km, os quais alcançam os bairros Guamá, Condor, Jurunas, Campina e Batista Campos.

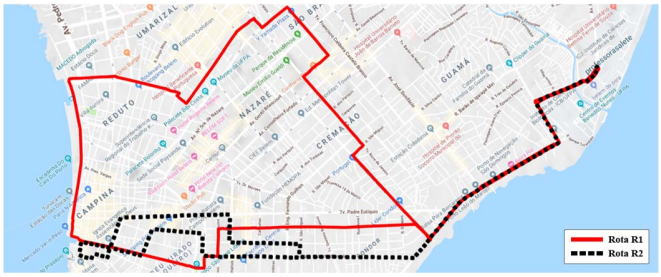


Fig. 5. Rotas de testes do caso de estudo na cidade de Belém-Pará.

Para os testes realizados nas rotas R1 e R2 foram considerados os parâmetros dispostos na Tabela I. Estes parâmetros são hipóteses, onde a distância máxima (d_{max}) foi definida levando em consideração erros de GPS em *smartphones* [24]. Já a escolha da velocidade mínima (v_{min}) de colaboração foi definida de acordo com velocidade média dos ônibus em grandes cidades [25]. Por fim, os parâmetros τ_1, τ_2 e τ_3, referentes ao temporizador regressivo, são defomodps de acordo com os autores [24][26][27].

Para τ_1 foi levado em consideração usuários que precisam parar em semáforos, engarrafamentos, acidentes, entre outros fatores que atrasam o percurso do ônibus dentro da rota [26]. Para τ_2 foi considerado a distância entre o usuário e a rota, seja

ela por grandes erros de GPS ou por usuários que saem dos veículos ou da rota [24], assim como a velocidade média ao caminhar [27]. Por fim, a definição de τ_3 levou em consideração situações onde pode haver desvios de rotas, sejam causados por bloqueios de vias ou por outras situações que atrasam o percurso durante a viagem dos coletivos [26].

TABELA I
PARÂMETROS DE CONFIGURAÇÃO DOS ALGORITMOS

Parâmetro	Valor
d_{max}	50 metros
v_{min}	10 km/h
τ_1	5 minutos
τ_2	1 minuto
τ_3	3 minutos

Para a tomada de decisão *online*, foi utilizado o algoritmo KNN, DMC e ATR apresentados na seção II. Além disso, foi adotada como métrica de avaliação o número de saltos dentro e fora a rota, isto é, um salto representa uma coleta discreta de $P_u(\varphi_u, \lambda_u)$.

V. TESTES E RESULTADOS

Esta seção apresenta os testes e resultados obtidos. O objetivo dos testes foi validar a efetividade dos algoritmos propostos em relação a correção de pontos coletados fora da rota pelo GPS para pontos dentro da rota. Os testes foram realizados em um veículo particular de forma a simular condições de viagem dos ônibus. No experimento, foram utilizados dois *smartphones* com o *software* OoO instalado. Um foi utilizado para simular o usuário colaborador, enquanto que o outro foi utilizado para acompanhar o monitoramento usando o mapa disposto no *software*. Para a comunicação entre os *softwares* e o servidor foi utilizado o serviço de Internet de 4ª geração (4G).

Durante os testes foi avaliada a eficiência dos algoritmos em relação a diferentes. Para a rota R1 foram realizados testes de (A) verificação de cumprimento da rota (com o usuário completando sua viagem e desabilitando manualmente a colaboração); (B) cumprimento da rota com desvio ocasionado por bloqueio de via; e desconexão por tempo parado dentro da rota (com $M_1 = 0$); e (C) cumprimento da rota com desvio ocasionado por bloqueio de via. Já para R2 foram realizados os testes de (D) desconexão por tempo parado fora da rota (com $M_1 = 1$); e (E) desconexão por tempo em movimento fora da rota (com $M_1 = 2$).

A. Resultados e Teste de Verificação de Cumprimento da Rota R1

O experimento foi realizado em 17 minutos e 8 segundos com a velocidade média de 21,7 km/h. O resultado demonstra que o usuário colaborador cumpriu sua trajetória de viagem de 6,2 km, não ultrapassou o limite definido de distância (d_{max}), manteve sua velocidade acima da mínima definida (v_{min}) e ao final desabilitou o modo colaborativo. No entanto, devido as imprecisões de geolocalização apresentadas pelo GPS, houve 38 saltos fora da rota, em uma trajetória realizada com 164 saltos.

Embora o número de saltos fora da rota tenha sido 38, antes de enviar os pontos geográficos ao servidor, o algoritmo KNN foi acionado para ajustar cada salto considerado fora da rota, possibilitando uma precisão de 100% na informação fornecida pelo usuário colaborador ao usuário final. A Fig. 6 ilustra a trajetória do usuário na rota R1, assim como os saltos reais e os saltos ajustados pelo algoritmo KNN. Com a utilização da IA, o usuário final recebeu como informação a rota ajustada.

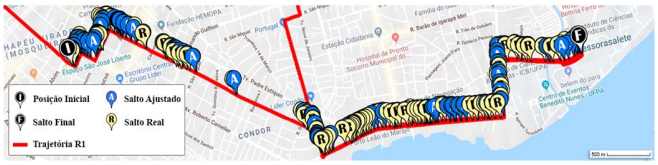


Fig. 6. Trajetória do veículo no teste de Verificação do Cumprimento da Rota R1.

B. Resultados e Teste de Verificação de Cumprimento da Rota R1 com Desvio Ocasionado por Bloqueio de Via

O experimento foi realizado com 467 saltos, no tempo de 40 minutos e 48 segundos e com uma velocidade média de 15 km/h. Durante a viagem do usuário dentro da rota, houve 134 saltos com erros causados pela imprecisão do GPS. Quando o colaborador realizou o desvio de trajeto foram contabilizados 22 saltos fora da rota. Todos os saltos fora da rota causados por erros de GPS e pelo desvio de via foram ajustados para dentro da rota. Durante o experimento, o usuário colaborador percorreu 10,25 km e manteve a velocidade maior que v_{min}.

Dos 10,25 km, 440 metros foram percorridos fora da rota devido a um bloqueio de via que obrigou o usuário colaborador desviar sua rota. Na Fig. 7 é possível visualizar o desvio de trajeto através dos saltos reais dispostos no *zoom*. A Fig. 7 também apresenta o ponto onde o temporizador regressivo (cronômetro) foi iniciado com τ_3 após o ATR ter identificado o desvio de rota. Antes de finalizar a contagem regressiva e o algoritmo DMC desabilitar a colaboração, o usuário colaborador voltou para a rota e cumpriu o restante de sua viagem dentro de R1.

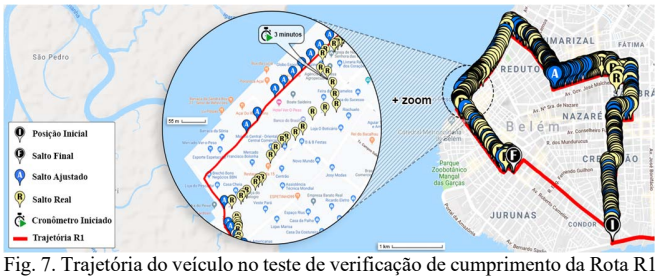


Fig. 7. Trajetória do veículo no teste de verificação de cumprimento da Rota R1 com desvio ocasionado por bloqueio de via.

Embora o KNN seja eficaz para ajustar todos os pontos fora da rota, para esse experimento não houve eficácia em ajustar pontos referentes ao desvio de rota feito pelo usuário, pois os saltos ajustados não correspondem com a localização real do agente colaborador. Além disso, é importante ressaltar que o algoritmo ATR teve sua importância em identificar o desvio de trajeto e que o tempo τ_3 estipulado no caso de estudo foi satisfatório para este caso de estudo, pois o usuário teve tempo suficiente para retornar para a rota após um desvio.

C. Resultados e Teste de Desconexão por Tempo Parado Dentro da Rota R1

Durante este experimento foram percorridos 2,92 km com duração de 8 minutos e 59 segundos. No primeiro quilômetro percorrido o automóvel atingiu a velocidade média de 19,67 km/h, após isso a velocidade foi reduzida e permaneceu abaixo de v_{min}. Desta forma, o temporizador regressivo foi ativado por um período de τ_1. Como a velocidade do veículo se manteve abaixo de v_{min} durante o tempo τ_1, a conexão do colaborador com o servidor *web* foi interrompida através do algoritmo DMC. Este teste teve 99 saltos, sendo 19 fora da rota.

Apesar de existirem saltos fora da rota causados por imprecisões de GPS, todos foram ajustados pelo algoritmo KNN para o posterior envio ao servidor *web*, desta forma a informação compartilhada pelo usuário colaborador ao usuário final teve a precisão de 100%. Já o algoritmo ATR foi eficiente em ativar o contador regressivo após identificar que a velocidade do veículo estava abaixo 10 km/h, considerada a velocidade mínima para colaboração (v_{min}), e o algoritmo DMC se mostrou eficaz em desconectar do servidor o usuário colaborador após o termino do tempo no contador regressivo. A Fig. 8 ilustra o experimento realizado, onde é possível visualizar a localização a qual o temporizador regressivo foi iniciado.

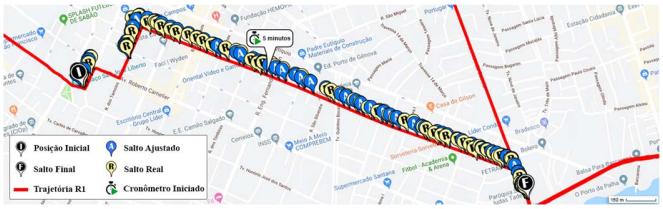


Fig. 8. Trajetória do veículo no teste de Desconexão por Tempo Parado Dentro da Rota R1.

D. Resultados e Teste de Desconexão por Tempo Parado Fora da Rota R2

Neste experimento foi percorrida a distância total de 10,43 km, sendo 10,15 km dentro da rota, isto é, $d \leq d_{max}$, com velocidade média de 15,1 km/h, e 280 metros fora da rota e com a distância acima de d_{max}.

Para desconectar o usuário, o veículo ultrapassou a distância d_{max} e em seguida reduziu sua velocidade para baixo de 10 km/h (v_{min}), neste momento M_1 assumiu o valor 1 e pelo algoritmo ATR foi ativado o contador regressivo com um tempo de τ_2. Após o período τ_2, o algoritmo DMC desconectou do servidor o usuário colaborador.

Neste teste houve 375 saltos do usuário, sendo que 108 foram identificados fora da rota. Dentre os saltos foram da rota, 78 ocorreram devido a imprecisão do GPS, enquanto que os demais ocorreram devido ao deslocamento do usuário para fora da rota. Embora o usuário tenha saído da rota, foi preservado para o usuário final o último ponto ajustado na rota até que o usuário colaborador seja totalmente desconectado. A Fig. 9 ilustra o teste em R2 onde o usuário colaborador desvia de trajetória, mas os pontos ajustados pelo KNN permanecem no ponto mais próximo da rota.

A mudança de trajetória e velocidade do veículo no teste, foram identificados de forma eficiente pelo algoritmo ATR. Apesar da mudança de trajeto pelo automóvel do colaborador, o algoritmo também DMC foi eficiente em permitir a colaboração do usuário apenas pelo tempo definido de τ_2. Vale ressaltar que estes resultados apresentam a importância dos algoritmos em identificar usuários mal-intencionados.

Fig. 9. Trajetória do veículo no teste de Desconexão por Tempo Parado For a da Rota R2.

E. Resultados e Teste de Desconexão por Tempo em Movimento Fora da Rota R2

Neste experimento o usuário colaborador se locomoveu por 3,65 km. Após 1,89 km este se deslocou para fora da rota e ultrapassou o d_{max} com uma velocidade superior a v_{min}, isto é 22,3 km/h. Satisfeita a condição $M_1 = 2$, foi iniciado o temporizador regressivo com o tempo τ_3. Como o usuário colaborador não retornou à rota durante o período de decréscimo de τ_3, o algoritmo DMC interrompeu a colaboração do usuário para evitar o compartilhamento de informações erradas. Durante o teste houve no mapa 119 saltos do usuário, onde 31 foram fora da rota e causados por erros de GPS, e 49 causados pelo desvio de trajeto feito pelo colaborador.

A Fig. 10 representa a trajetória feita durante o experimento, com os saltos reais dentro e fora da rota, e os saltos ajustados pelo algoritmo KNN. Conforme pode ser observado, embora o usuário colaborador tenha a intenção de informar pontos fora da rota, o algoritmo de KNN corrige todas as possibilidades de saídas da rota desejada. No entanto, a utilização deste algoritmo para este caso não é a melhor solução, pois a distância do veículo para rota é consideravelmente grande. Desta forma a informação de localização real do automóvel é considerada bastante diferente da localização ajustada e apresentada no mapa do aplicativo OoO para o usuário final.

Como resultado, para este teste, a confiabilidade das informações colaboradas foi garantida pelos algoritmos ATR e DMC, onde o algoritmo ATR identificou a saída do usuário colaborador da rota e o algoritmo DMC interrompeu o envio constante de informações incorretas ao servidor.

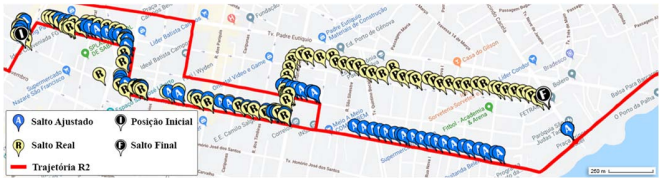


Fig. 10. Trajetória do veículo no teste de Desconexão por Saída da Rota R2.

VI. Conclusões

Este trabalho apresentou a viabilidade técnica do monitoramento *online* e colaborativo de rotas de ônibus através da utilização do algoritmo KNN em conjunto com os algoritmos ATR e DMC. Para provar a eficiência dos algoritmos propostos, experimentos foram realizados em duas rotas de linhas de ônibus da cidade de Belém, onde foram avaliados a eficácia em

termos de verificação de cumprimento da rota com e sem desvio ocasionado por bloqueio de via, desconexão por tempo parado dentro da rota, desconexão por tempo parado fora da rota e desconexão do usuário colaborador por tempo em movimento fora da rota. É importante ressaltar que nos experimentos realizados foi percorrida uma distância total de 33,48 km.

Os resultados demonstram que os algoritmos conseguem retornar para o usuário final informações de localização com 100% de acerto para testes feitos com a verificação de cumprimento da rota e desconexão por tempo parado dentro da rota, pois o algoritmo KNN ajusta todas as imprecisões provenientes do GPS. Já para os testes onde o usuário colaborador se posiciona fora da rota, o algoritmo KNN ajusta a localização deste usuário em 100% para dentro da rota. No entanto, a informação para o usuário final não é 100% confiável, neste caso, a margem de erro vai depender do tempo τ_2 ou τ_3 para o algoritmo DMC desconectar o agente colaborador. Para este caso, onde há imprecisão das informações quando o colaborador sai da rota, cabe um ajuste de calibragem dos parâmetros τ_2 e τ_3, para que se possa diminuir o tempo de erro de informações compartilhadas. Em todo o caso, os algoritmos ATR e DMC foram eficazes em realizar as tarefas de identificar e desconectar usuários mal-intencionados.

Pode-se concluir a partir dos resultados apresentados que é possível realizar o monitoramento *online* e colaborativo de rotas de ônibus usando o *software* OoO em conjunto com os algoritmos KNN, ATR e DMC.

REFERÊNCIAS

[1] C. H. R. Carvalho, "Políticas de melhoria das condições de acessibilidade do transporte urbano no Brasil", *Instituto de Pesquisa Econômica Aplicada (Ipea)*. 2015.

[2] D. A. M. O. Cruz, "Problemas do transporte público coletivo em Presidente Prudente/SP", *Revista Percurso*, vol. 5, no. 1, pp. 179-196, 2013.

[3] Moovit Company Overview. (2014). [Online]. Available: http://moovitapp.com/wp-content/uploads/2014/07/Moovit-At-A-Glance-USA-4-14.pdf.

[4] Cadê o Ônibus?. (2019). [Online]. Available: http://www.cadeoonibus.com.br.

[5] V. C. Castro, J. S. Sá, M. A. R. Mocbel, D. Acatauassu, F. J. B. Barros and F. S. Farias, "Online Monitoring For Public Transport Using Mobile Applications", in *IEEE Latin America Transactions,* vol. 16, no. 05, pp. 1780-1786, 2018.

[6] K. Farkas, G. Feher, A. Benczur and C. Sidlo, "Crowdsending based public transport information service in smart cities", in *IEEE Communications Magazine*, vol. 53, no. 8, pp. 158-165, August 2015.

[7] S. I. Chien, Y. Ding and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks", *Journal of Transportation Engineering*, vol. 128, no. 5, pp. 429-438, 2002.

[8] A. Abadi, T. Rajabioun and P. A. Ioannou, "Traffic Flow Prediction for Road Transportation Networks With Limited Traffic Data", in *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 653-662, April 2015.

[9] M. L. Braga, A. J. Santos, A. C. P. Pedroza and L. H. M. K. Costa, "Planejamento de Rotas com Algoritmos Anytime em Redes Veiculares na Plataforma Raspberry Pi", in *IV Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pp. 1-6, 2014.

[10] Y. Huang, R. Jin, F. Bastani and X. S. Wang, "Large scale real-time ridesharing with service guarantee on road networks", in *Proceedings of the VLDB Endowment*, vol. 7, no. 14, pp. 2017-2028, 2014.

[11] R. Jeong and L. R. Rilett, "Bus arrival time prediction using artificial neural network model", *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 988-993, 2004.

[12] Y. Yan, Z. Liu, Q. Meng and Y. Jiang, "Robust optimization model of bus transit network design with stochastic travel time", in *Journal of Transportation Engineering*, vol. 139, no. 6, pp. 625-634, 2013.

[13] S. M. M. Amiripour, A. S. Mohaymany, A. Ceder, "Optimal modification of urban bus network routes using a genetic Algorithm", *Journal of Transportation Engineering*, vol. 141, no. 3, pp. 04014081, 2014.

[14] S. Eken, A. Sayar, "A smart bus tracking system based on location-aware services and QR codes", in *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, pp. 299-303, 2014.

[15] C. Bai, Z. R. Peng, Q. C. Lu, and J. Sun, "Dynamic bus travel time prediction models on road with multiple bus routes", *Computational intelligence and neuroscience*, vol. 2015, pp. 1-9, 2015.

[16] M. Sinn, J. W. Yoon, F. Calabrese and E. Bouillet, "Predicting arrival times of buses using real-time GPS measurements", in *15th International IEEE Conference on Intelligent Transportation Systems* IEEE, pp. 1227-1232, 2012.

[17] A. P. S. Thengal, N. Rastogi, A. Medhi, R. Srivastava, K. Datta, "Parameter sensing and object tracking using global positioning system", in *2016 Sixth International Symposium on Embedded Computing and System Design (ISED), IEEE*, pp. 289-293, 2016.

[18] D. Ingle and A. B. Bagwan, "Real-Time Analysis and Simulation of Efficient Bus Monitoring System", in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE*, pp. 128-133, 2018.

[19] K. Faceli, A.C. Lorena, J. Gama, and A. C. P. D. L. Carvalho, "Inteligência Artificial: Uma abordagem de aprendizado de máquina", 2011.

[20] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. H. Zhou, M. Steinbach, D. J. Hand and D. Steinberg, "Top 10 algorithms in data mining", *Knowledge and information systems*, vol. 14, no. 1, pp. 1-37, 2008.

[21] J. Machaj and P. Brida. "Performance comparison of similarity measurements for database correlation localization method", in *Asian Conference on Intelligent Information and Database Systems*. Springer, Berlin, Heidelberg, pp. 452-461, 2011.

[22] N. R. Chopde and M. K. Nichat, "Landmark based shortest path detection by using A* and Haversine formula", in *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 298-302, 2013.

[23] Semob. "Itinerários das Linhas de Ônibus". (2019). [Online]. Available: http://www.belem.pa.gov.br/semob/site/wp-content/uploads/2018/02/ITINER%C3%81RIOS-%C3%94NIBUS.pdf.

[24] E. C. Lima, "Proposta de metodologia para melhora do posicionamento obtido através de receptores GPS de baixo custo", Tese de doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, BR, 2018.

[25] R. Rolnik and D. Klintowitz, "(I) Mobilidade na cidade de São Paulo", *Estudos Avançados*, vol. 25, no. 71, pp. 89-108, 2011.

[26] M. C. M. Ladeira, F. D. Michel, L. A. S. Senna, "Estratégias de controle da operação de linhas de ônibus", *Associação Nacional de Pesquisa e Ensino em Transportes–ANPET*, (2013).

[27] W. D. Monteiro, C. G. S. Araújo, "Transição caminhada-corrida: considerações fisiológicas e perspectivas para estudos futuros", in *Rev Bras Med Esporte*, vol. 7, no. 6, pp. 207-222, 2001.

Joiner S. Sá possui Especialização em Gestão de Projetos e Desenvolvimento de *Softwares* e graduação em Bacharelado em Sistemas de Informação, ambos pela UFPA. Membro colaborador do Laboratório de Programação Extrema (LABEX) desde 2015.

Vicente C. Castro possui Especialização em Gestão de Projetos e Desenvolvimento de *Softwares* pela UFPA. Possui graduação em Bacharelado em Sistemas de Informação pela UFPA. Membro colaborador do LABEX desde 2015.

Eliezer M. Coelho possui Especialização em Gestão de Projetos e Desenvolvimento de *Softwares* pela UFPA. Graduação em Bacharelado em Sistemas de Informação pela UFPA. Atua como membro colaborador para o LABEX.

Fabricio S. Farias possui Doutorado em Engenharia Elétrica, com ênfase em Computação Aplicada, pela UFPA (2016). Professor Adjunto na UFPA e Coordenador do LABEX.