Planejamento de Rotas com Algoritmos Anytime em Redes Veiculares na Plataforma Raspberry Pi

Marcus de L. Braga, Alyson de J. dos Santos, Aloysio C. P. Pedroza and Luís H. M. K. Costa
Universidade Federal do Rio de Janeiro (UFRJ) - PEE/COPPE/GTA - DEL/POLI
Email:{marcus,alyson,aloysio,luish}@gta.ufrj.br

***Abstract*—Mobility became a major challenge in urban areas around the world, with frequent congestions and ever growing commuting time. Even with recent advances in the monitoring of road infrastructure through Intelligent Transportation Systems (ITS), the dynamics of such systems make it difficult to forecast and manage the traffic of vehicles. One key issue is how, given traffic monitoring, a vehicle decides to dynamically change its route. In this context, we analyze algorithms of the anytime class to calculate the best route between two spots, taking into account the conditions obtained from a traffic monitoring system. If their computation is interrupted, anytime algorithms inform a suboptimal response. This is made possible by limiting the search area. On the other hand, the routes produced by anytime algorithms are progressively improved according to the time available. During the search phase, the results may be recalculated if the traffic has changed. In this paper, we analyze two anytime algorithms: ARA*, and ITS-ARA*, a version of ARA* adapted to the ITS scenario. For the validation, experiments were performed using Raspberry Pi as a test platform and using real maps of two Brazilian cities. We evaluated the time and memory consumption. For different scenarios, the results show a better performance of ITS-ARA* in comparison with Dijkstra, A*, and ARA* algorithms.**

I. Introdução

A garantia da mobilidade urbana tem sido um dos grandes desafios para os gestores das metrópoles em todo o mundo, bastante prejudicadas pelos constantes congestionamentos que causam impactos econômicos, sociais e ambientais e que há muito fazem parte do cotidiano de seus habitantes. Falhas mecânicas ou elétricas nos veículos, obras nas vias, mudanças bruscas de faixa são fatores que colaboram para o desequilíbrio do sistema. O ser humano, motorista ou pedestre, com seu comportamento imprevisível, suscetível à fadiga, distração e a substâncias que causam o decréscimo de atenção e reflexos, é outra variável inerente ao sistema e que não pode ser ignorada [1], [2].

A ampliação da rede viária e as intervenções promovidas são ações largamente utilizadas e que visam aumentar a eficiência do fluxo dos veículos. No entanto, tais medidas possuem prazo delimitado, seguramente até o próximo crescimento inevitável da demanda [3]. O monitoramento das vias por sensores e câmeras, o uso de placas eletrônicas sobre as condições de tráfego, controle semafórico para favorecer o fluxo nos horário de grande densidade de veículos, são algumas aplicações de Sistemas de Transportes Inteligentes (ITS - *Intelligent Transportation Systems*), que visam auxiliar as Companhias de Engenharia de Tráfego (CET) na administração do trânsito [4]. Contudo, todas essas iniciativas possuem o objetivo de minimizar os atrasos percebidos pelos condutores de veículos, não contemplando o controle e o gerenciamento da rede viária.

A busca por rotas alternativas é uma solução desejada por muitos condutores para escapar dos congestionamentos. Entretanto, não existem garantias de sucesso, devido à ausência de informações atualizadas das condições físicas das vias e densidade de veículos. Mesmo com a tecnologia empregada nos sistemas de trânsito, ainda não existem técnicas de autogestão capazes de equilibrar o sistema e proporcionar uma gestão de tráfego proativa com garantia da mobilidade na rede viária [5].

Este trabalho tem como inspiração as pesquisas realizadas em redes de computadores, tendo por foco o congestionamento e otimização de rotas. Grafos dinâmicos são usados na modelagem da rede viária da seguinte forma: as vias correspondem às arestas e suas interseções, cruzamentos, indicam seus respectivos vértices. A ponderação é associada ao tempo para se percorrer o comprimento da via (condições de tráfego no trecho) [6]. Com isso, o problema passa a ser a escolha da melhor rota (caminho mínimo), utilizando a técnica de planejamento de rotas, largamente estudada em problemas relacionados à teoria dos grafos.

O planejamento de rotas consiste na descoberta do caminho mínimo com base nos estados das arestas com custo associado. O caminho é ótimo, quando a soma de seus custos de transição é mínima, a partir da origem até seu destino, em todas as possíveis combinações [7]. Este trabalho estuda o uso de uma versão modificada do algoritmo de busca *Anytime Repairing A** (ARA*) proposto por Likhachev [8], para o planejamento de rotas em redes viárias reais, que faz uso de um sistema de monitoramento de trânsito.

Para o estudo foram implementados os algoritmos *anytime* ARA* e ITS-ARA* (uma versão de ARA* para ITS), além dos clássicos Dijkstra e A*. Os cenários utilizados foram os mapas das cidades de Manaus e Rio de Janeiro. O Raspberry Pi foi a plataforma computacional escolhida para realização dos testes, devido ao tamanho diminuto e possilidade de comunicação por IEEE 802.11, características favoráveis à mobilidade. Foram medidos tempo de execução, memória consumida, quantidade de vértices componentes da solução encontrada e a influência da heurística nos resultados.

O artigo está organizado da seguinte forma. Na Seção II discute os trabalhos relacionados. Na Seção III são apresentados os algoritmos ARA* e ITS-ARA*. Na Seção IV são descritos

os experimentos, resultados e suas análises. Na Seção V o trabalho é concluído e trabalhos futuros são discutidos.

II. Trabalhos Relacionados

Um dos principais problemas em teoria dos grafos é o do caminho mínimo, onde deseja-se encontrar uma rota mais curta, caso exista, entre dois vértices do grafo. Para tal, um grafo é dado por $G = (V, A)$, onde V representa um conjunto não vazio de vértices e A um conjunto de pares não ordenados denominado de arestas. Tradicionalmente este problema é estudado em estruturas estáticas, no entanto, muitos problemas que envolvem grafos apresentam natureza dinâmica, como as redes viárias.

Em Faez e Khanjary [4] o algoritmo de Dijkstra é utilizado para encontrar o menor caminho baseado nas condições de trânsito coletadas por Redes de Sensores Sem Fio (RSSF) e sistemas ITS (sensores ópticos, indutivos, magnéticos e câmeras de vídeo). As rotas otimizadas são enviadas aos veículos via mensagens, no modo básico. No modo avançado, são previstos veículos equipados com transceptores para proporcionar comunicação direta e interativa.

O sistema *MobEyes* proposto por Lee *et al.* [9] utiliza-se de um algoritmo bio-inspirado para determinação do melhor caminho, usando informações de sensores que captam, classificam e geram periodicamente resumos das informações extraídas do contexto do tráfego de veículos. Estes resumos são disseminados na rede veicular utilizando agentes móveis oportunistas.

Em Li *et al.* [10] é proposta a utilização do algoritmo de busca A* com uma abordagem adaptativa de tempo real baseada nas condições do tráfego durante o período de um dia: a escolha é feita com base no menor custo para diferentes horários do dia. A proposta utiliza regras *fuzzy* para adaptação às mudanças no tráfego baseadas no tráfego ou ajustes feitos pelo motorista. Em Moura *et al.* [11] foi feito um comparativo do algoritmo A* com os algoritmos ARA* e *Anytime Dynamic A** (AD*) propostos por Likhachev [12]. Como cenário de testes foi utilizada uma rede rodoviária. Para o comparativo entre A* e ARA*, o coeficiente heurístico ϵ foi variado e seus resultados foram expressos em relação à aceleração do ARA*. Já para A* e AD*, foram feitas variações nos pesos das arestas da região de interesse para medir o desempenho em um ambiente dinâmico.

Neste trabalho são investigados quatro algoritmos: os clássicos Dijkstra e A*, o ARA* que calcula rapidamente um caminho subótimo, e o ITS-ARA*, uma versão proposta de ARA* adaptado para ITS. Para a realização dos testes foi utilizado o Rapberry Pi como plataforma computacional e os cenários utilizados foram os mapas viários das cidades de Manaus e Rio de Janeiro. Foram medidos tempo de execução, a memória consumida, quantidade de vértices componentes do caminho encontrado, além do estudo da influência do uso da função heurística nos resultados.

III. Algoritmos Anytime

Os algoritmos *anytime* de planejamento de rotas são oriundos da área de Inteligência Artificial (IA) e têm por característica apresentar resultados subótimos rapidamente e cuja melhora é gradual, caso o tempo seja suficiente o caminho é ótimo [13]. Analogamente ao caminho ótimo, diz-se que um algoritmo de planejamento é completo se encontrar um caminho, caso exista, em tempo finito. Caso não exista, o algoritmo sempre informa a sua não existência em tempo finito. Consequentemente, um algoritmo de planejamento é ótimo se ele encontra sempre um caminho ótimo [8], [12]. A seguir, serão apresentadas as descrições e os pseudo-códigos dos algoritmos estudados neste trabalho.

A. Anytime Repairing A* – ARA*

O algoritmo ARA* (Algoritmo 1) foi proposto por Maxim Likhachev [12] como uma alternativa ao algoritmo A* na busca baseada em grafo, mais precisamente para o planejamento de trajetórias de robôs. A diferença entre ARA* e A* está no mecanismo de heurística e no refinamento de pesquisa. ARA* tende a encontrar resultados subótimos mais rápido em detrimento à otimalidade do seu antecessor. No entanto, caso o tempo seja suficiente, ARA* encontra resultados ótimos, graças à utilização da técnica *heurística inflada*, que reduz o número de vértices a serem verificados e consequentemente o tempo de execução. Isto é possível devido o aumento no custo das distâncias $h(x)$, onde os primeiros vértices para serem verificados serão os vértices com menor custo, ignorando os vértices com custo semelhante que poderiam fazer parte do resultado.

A função de ARA* é dada por $f(x) = g(x) + \epsilon * h(x)$, onde $g(x)$ é a distância do vértice x até a origem, $h(x)$ é uma estimativa (heurística) da distância até o vértice final, ϵ é o coeficiente heurístico. Quanto maior o valor de ϵ, menos vértices serão visitados e os resultados são calculados rapidamente. A garantia da otimalidade é alcaçada conforme se decrementa o valor de ϵ até 1, pois a função passa a ser a mesma de A*. No processo de refino de resultados, decremento de ϵ, os valores $f(x)$, $g(x)$ e $h(x)$ são atualizados para cada vértice previamente encontrado. Para os experimentos foi utilizada a distância Euclidiana como função heurística, por ser intuitiva a ideia da distância em linha reta entre dois pontos: seja um plano p_1 em (x_1, y_1) e p_2 em (x_2, y_2), tem-se $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

O algoritmo trabalha com três listas: *OPEN*, *CLOSED* e *INCONS* (linha 27). A lista *OPEN* contém os vértices que foram submetidos à função heurística $h(x)$, mas que ainda não foram examinados e expandidos, trata-se de uma lista de prioridades de maior valor da função heurística. A lista *CLOSED* contém os vértices já examinados e expandidos uma vez na pesquisa. A lista *INCONS* (inconsistente) é responsável por evitar que vértices que tenham sua função $f(x)$ reduzida e que não estejam na lista *OPEN* sejam visitados mais de uma vez, é uma lista temporária até o momento do refinamento é feito e seu conteúdo é adicionado à lista *OPEN*, fazendo com que os vértices já visitados tenham suas funções $f(x)$ reduzidas e sejam novamente avaliados. Isso significa que não é necessário recalcular o caminho mais uma vez, mas somente atualizar os pesos, de modo que uma *"nova rota"* seja

Algoritmo 1 ARA*

```
1: função FVALUES(X)
2:     retorna g(x) + ε * h(x)
3: fim função
4:
5: procedure IMPROVEPATH())
6:     enquanto (key(x_destino) > min_{x∈OPEN}(key(x)) faça
7:         remove x com o menor fvalues(x)from OPEN;
8:         CLOSED = CLOSED ∪ x;
9:         para vizinhos(x) ← 1 até fim faça
10:            se vizinhos(x) nunca visitado por ARA* antes então
11:                g(vizinhos(x)) = ∞
12:            fim se
13:            se g(vizinhos(x)) > g(x) + c (x, vizinhos(x)) então
14:                g(vizinho(x)) = g(x) + h(x, vizinho(x))
15:                se (vizinho(x) ∉ CLOSED) então
16:                    insert/update vizinho(x) in OPEN com fvalues(vizinho(x));
17:                senão
18:                    insert vizinho(x) em INCONS;
19:                fim se
20:            fim se
21:        fim para
22:    fim enquanto
23: fim procedure
24:
25: procedure MAIN())
26:    g(x_destino) = v(x_destino) = ∞; v(x_destino) = ∞;
27:    g(x_origem) = 0; OPEN = CLOSED = INCONS = ∅;
28:    insere x_origem em OPEN com key(x_origem);
29:    improvePath();
30:    informa atual solução ε − suboptimal;
31:    enquanto ε > 1 faça
32:        decrementa ε;
33:        Mova os vértice de INCONS para OPEN
34:        Atualiza as prioridades x ∈ OPEN de acordo com fValues(x);
35:        CLOSED = ∅;
36:        improvePath();
37:        informa atual solução ε − suboptimal;
38:    fim enquanto
39: fim procedure
```

concebida. ARA* é composto por três procedimentos: *valorF* responsável pelo cálculo $f(x) = g(x) + \epsilon * h(x)$; *improvePath* que executa o refinamento de pesquisas; e *Main* que gerencia o algoritmo, desde a criação das três listas (*OPEN*, *CLOSED* e *INCONS*) ao decréscimo do coeficiente heurístico ϵ. Em essência, ARA* trabalha como se executasse A* várias vezes, seu controle está no decremento do coeficiente heurístico ϵ até 1, condição que garante a otimalidade da solução [8]. O ARA* difere do A*, por não garantir a visitação única a cada vértice durante a pesquisa, isto se deve à super-estimação da função heurística causada pela inflação de ϵ.

O procedimento *improvePath* (linhas 5–23) é responsável pela reutilização dos resultados encontrados e que fazem parte da lista *INCONS*. Esta lista é composta por vértices que foram visitados (pertencem à lista *CLOSED*), mas que tiveram sua função heurística reduzida, implicando em uma nova avaliação para verificar se será parte da solução. Esta operação é realizada no procedimento *Main* quando os vértices da lista *INCONS* são movidos para a lista *OPEN*. A função *valorF* é responsável pelo cálculo da heurística função $f(x)$ para cada vértice.

O procedimento *Main* (linhas 25–39) define os valores de ϵ e inicializa o cálculo da solução subótima. O refinamento da solução é realizado por um laço, onde as operações de atualização heurística nos vértices são realizados por cada diminuição de heurística coeficiente ϵ até 1 (solução ótima).

B. ITS-ARA*

O algoritmo ITS-ARA* (Algoritmo 2) é uma versão da ARA* para aplicações ITS, com rápida convergência do coeficiente heurístico na busca do caminho ótimo. ITS-ARA* reduz o número de chamadas de *improvePath* e *Atualiza as prioridades* (linhas 29–30 e 36–37 do Algoritmo 1) dentro do *Enquanto* (linhas 6–7), com intuito de promover um caso mais geral e acelerar a convergência ϵ para 1 para assegurar que a solução encontrada seja ótima (linhas 5–14). Em outras palavras, ITS-ARA* encontra rapidamente o caminho subótimo, verifica as possíveis alterações nos pesos e caso ocorram, recalcula o caminho ótimo. A seguir serão mostrados detalhes sobre os experimentos como cenários utilizados, metodologia empregada e resultados obtidos.

Algoritmo 2 ITS-ARA*

```
1: procedure MAIN())
2:    g(x_destino = v(x_destino) = ∞; v(x_destino) = ∞;
3:    g(x_destino = 0; OPEN = CLOSED = INCONS = ∅;
4:    insere x_destino em OPEN com fvalues(x_destino);
5:    enquanto ε >= 1 faça
6:        improvePath();
7:        informa atual solução ε − suboptimal;
8:        se (ε > 1) então
9:            Mova os vértice de INCONS para OPEN
10:           Atualiza as prioridades x ∈ OPEN de acordo com fvalues(x);
11:           CLOSED = ∅;
12:       fim se
13:       decrementa ε;
14:   fim enquanto
15: fim procedure
```

IV. Avaliação dos Algoritmos

Para o experimento foram utilizados mapas reais das cidades brasileiras de Manaus e Rio de Janeiro. Os mapas foram baixados do *OpenStreetMap*, onde os arquivos são do tipo OSM XML que fornecem basicamente dados primitivos como os nós (pontos no espaço), vias (formas lineares com limitações) e relações (explicam como os dados primitivos se relacionam). O cenário de Manaus abrange os bairros da zona centro-sul com 4.723 vértices, 5.645 arestas e diâmetro (maior distância entre dois vértices) de 139. Para o cenário do Rio de Janeiro foi utilizada a rede viária da Ilha do Governador, composta de 7.146 vértices, 7.968 arestas e diâmetro de 254. A avaliação de desempenho foi feita usando o ambiente de desenvolvimento *Eclipse* com Java e a biblioteca de grafos dinâmicos *GraphStream* [14], de onde foram gerados executáveis *.jar* para geração e coleta dos dados no Raspberry Pi. Para os cenários foram feitas ponderações aleatórias (variando até 80) em suas vias (arestas).

Para avaliar a influência das heurísticas no cálculo das rotas, foram implementadas versões dos algoritmos com e sem utilização da função heurística. Os experimentos consistiram de três sorteios aleatórios de dois pontos (origem e destino) para cada cenário. Em seguida a cada descoberta de rota, uma nova ponderação aleatória foi aplicada na região em estudo. No total foram realizadas 50 repetições para cada caminho estudado. Foram extraídas as médias dos resultados com intervalo de confiança de 95% e os experimentos foram

executados em um Raspberry Pi de modelo B com processador ARM de 700 MHz, 512 MB de RAM e cartão de memória de 4 GB, sistema operacional Raspbian e Java Embarcado 1.6.0_27.

A Figura 1 mostra a influência da heurística nos caminhos encontrados. As rotas dos algoritmos são designadas por cores, o azul designa Dijkstra, A* é representado por vermelho e a cor verde identifica a rota produzida por ITS-ARA*. As rotas de ARA* não foram identificadas devido serem as mesmas de ITS-ARA*. Com aplicação da heurística, os caminhos alternativos são mais evidentes conforme podem ser vistos nas Figuras 1(b) e 1(d). No entanto, nas Figuras 1(a) e 1(c) evidenciam a ausência da heurística, fazendo com que as rotas geradas fiquem mais próximas à gerada por Dijkstra. A Tabela I confirma a observação feita por meio da quantidade de vértices constituintes da rota. Quando a função heurística foi utilizada, os algoritmos A* e ITS-ARA* apresentaram rotas alternativas (Figuras 1(b) and 1(d)). Os pinos azul e verde são usados para marcar a origem e o destino, respectivamente. O resultado das análises é feito a seguir, mostrando tempo de execução, memória consumida e número de vértices para cada cenário.

A. Resultados sem Heurística

Nesta seção a função heurística $h(x)$ não foi considerada para o cálculo do menor caminho nos algoritmos de busca informados como A*, ARA* e ITS-ARA*. Dessa forma, temos a função expressa por $f(x) = g(x)$. Os resultados obtidos quanto ao tempo para ambos os cenários mostraram que A* foi superior aos demais. Todavia, quanto à memória Dijkstra foi o mais econômico. Isso se deve ao fato que os algoritmos de busca informada precisarem manter os resultados encontrados na memória durante as operações de busca. As listas (*OPEN*, *CLOSED* e *INCONS*, encontradas em ARA* e ITS-ARA*) são as estruturas utilizadas para tais operações. A Figura 2 exibe os resultados dos tempos gastos e memória consumida nos respectivos cenários. Para o cenário de Manaus (Figura 2(a)), em média todos os algoritmos tiveram melhor desempenho em relação ao tempo que no cenário do Rio de Janeiro (Figura 2(c)). O gasto de memória também é maior no Rio de Janeiro (Figura 2(d)) do que em Manaus (Figura 2(b)).

Manaus			
DJK	A*	ARA*	ITS-ARA*
38	38,50	39,02	39,46
82	89,04	88,38	88,38
58	60,46	62,62	60,92
Rio de Janeiro			
DJK	A*	ARA*	ITS-ARA*
129	133,40	135,08	130,48
81	82,20	82,10	84,26
73	73,92	75,64	75,42

TABELA I
RESULTADOS SEM HEURÍSTICA – QUANTIDADE MÉDIA DE VÉRTICES.

A Tabela I mostra os resultados dos experimentos para os cenários de Manaus e Rio de Janeiro, respectivamente. Pode-se notar que a variação aleatória dos pesos nos caminhos encontrados, foram refletidos nas médias decimais dos resultados dos algoritmos A*, ARA* e ITS-ARA* para os dois cenários.

B. Resultados com Heurística

Em ambos os cenários, é bastante significativo o aumento de desempenho em relação ao tempo, quando a função heurística é adicionada aos algoritmos de busca informada (Figura 3). Quando se compara os cenários de Manaus (Figura 3(a)) e Rio de Janeiro (Figura 3(c)) aos seus respectivos resultados sem heurística (Figuras 2(a) e 2(c)) pode-se notar o ganho em relação ao tempo de execução que ambos tiveram com o uso da heurística. No entanto, em relação à memória, os cenários de Manaus e Rio de Janeiro (Figuras 3(b) e 3(d)) tiveram poucas variações em comparação aos resultados sem heurística (Figuras 2(b) e 2(d)).

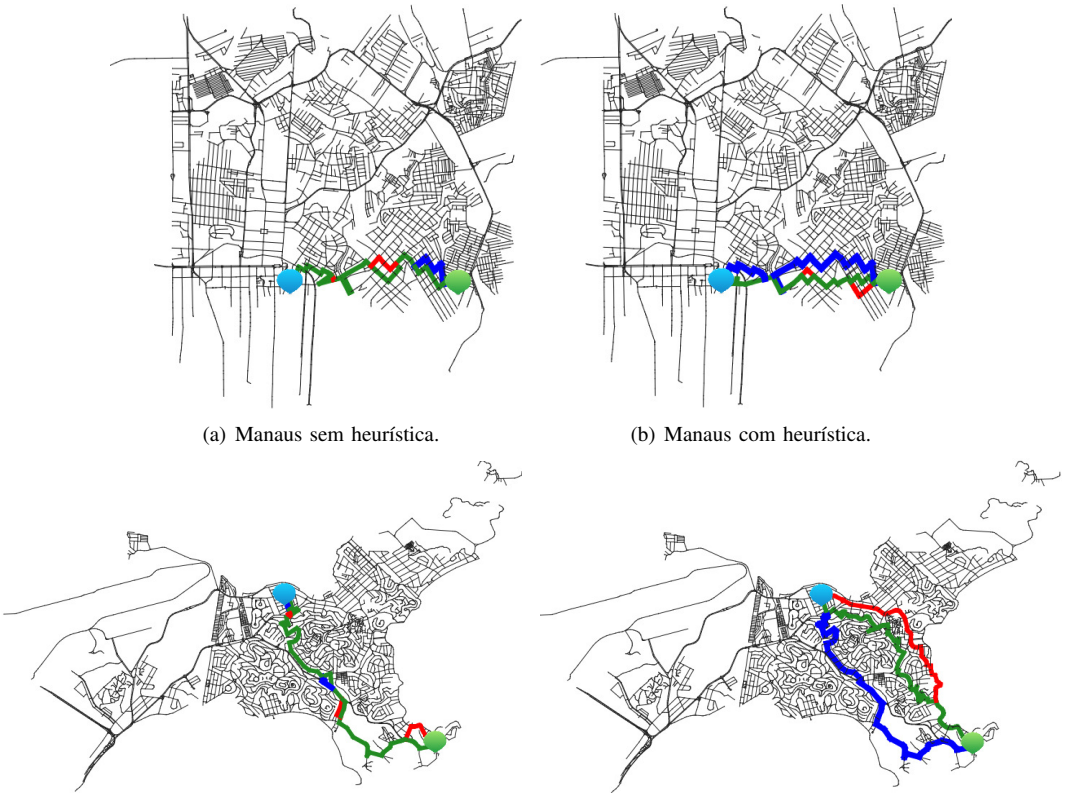
Manaus		
A*	ARA*	ITS-ARA*
42	54	54
92	129	129
64	60	60
Rio de Janeiro		
A*	ARA*	ITS-ARA*
162	190	190
87	87	87
76	75	75

TABELA II
RESULTADOS COM HEURÍSTICA – QUANTIDADE MÉDIA DE VÉRTICES.

A Tabela II ilustra os resultados em relação aos caminhos encontrados de Manaus e Rio de Janeiro respectivamente. Pode-se notar que ARA* e ITS-ARA* tiveram os mesmos valores inteiros quanto à quantidade de vértices, em ambos os cenários, tendo valores maiores quando comparados aos demais.

V. CONCLUSÕES E TRABALHOS FUTUROS

Este artigo apresentou uma análise da desempenho dos algoritmos de busca *Anytime* no planejamento de rotas nas redes viárias de Manaus e Rio de Janeiro utilizando a plataforma Raspberry Pi. Foi analisada a influência da função heurística nos cenários propostos. ITS-ARA*, a modificação proposta de ARA*, apresentou resultados relevantes em ambos os cenários quanto ao tempo de execução, principalmente quando se faz uso da função heurística. Para o cenário de Manaus, em média o ITS-ARA* foi mais rápido 7,71 vezes do que o Dijkstra (tempo DJK/tempo ITS-ARA*), usando 12,96% do seu tempo. Da mesma forma, em relação ao A*, o ITS-ARA* foi 3,16 vezes mais rápido consumindo 68,41% do tempo. E quando comparado com ARA*, o ITS-ARA* foi 1,45 vezes mais rápido e consumindo 31,37% de tempo. Para o Rio de Janeiro, em média: ITS-ARA* foi 14,70 vezes mais rápido que Dijkstra, usando 6,98% de seu tempo. Para o A*, ITS-ARA* foi 7,44 vezes mais rápido, consumindo 13,46% de seu tempo. Em comparação com o ARA*, ITS-ARA* foi 1,49 mais rápido, usando 67,05% de tempo.



(a) Manaus sem heurística. (b) Manaus com heurística.

(c) Rio de Janeiro sem heurística. (d) Rio de Janeiro com heurística.

Fig. 1. Influência da heurística. no planejamento de rotas.

Apesar de ITS-ARA* ser uma modificação de ARA*, ele mantém as características do original, como agregar a rapidez do A* e a eficiência da reutilização de resultados encontrados, não necessitando de novas buscas quando as ponderações são modificadas. Esta última característica é umd iferencial para aplicações em sistemas dinâmicos como as redes viárias. Os experimentos foram bastante satisfatórios tanto em relação ao desempenho do algoritmo ITS-ARA* quanto a viabilidade do Raspberry Pi para o desenvolvimento de um protótipo para uso veicular que utilize IEEE 802.11 no planejamento de rotas, sendo este um dos trabalhos futuros, bem como o desenvolvimento de um aplicativo para *smartphones*.

AGRADECIMENTOS

Esse trabalho foi realizado com recursos da CAPES, CNPq, Fapeam, Faperj e Tribunal de Justiça do Amazonas.

REFERENCES

[1] L. Iftekhar and R. Olfati-Saber, "Autonomous Driving for Vehicular Networks with Nonlinear Dynamics," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012, pp. 723–729.

[2] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, *Dynamic Traffic Assignment: A Primer*, 1st ed., ser. Transportation Research Circular E-C153. Transportation Research Board, (2011).

[3] F. Angius, M. Reineri, C. Chiasserini, M. Gerla, and G. Pau, "Towards a Realistic Optimization of Urban Traffic Flows," in *Intelligent Transportation Systems (ITSC)*, 2012, pp. 1661–1668.

[4] K. Faez and M. Khanjary, "Utospf with Waiting Times for Green Light Consideration," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 2009, pp. 4170–4174.

[5] W. Kim and M. Gerla, "Navopt: Navigator Assisted Vehicular Route Optimizer," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2011, pp. 450–455.

[6] J. G. Ribeiro Júnior, C. M., M. E. M., and L. H. M. K. Costa, "A Decentralized Traffic Monitoring System Based on Vehicle-to-Infrastructure Communications," in *Wireless Days (WD), 2013 IFIP*, 2013, pp. 1–6.

[7] D. Ferguson, M. Likhachev, and A. Stentz, "A Guide to Heuristic based Path Planning," in *in: Proceedings of the Workshop on Planning under Uncertainty for Autonomous Systems at The International Conference on Automated Planning and Scheduling*, 2005.

[8] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," in *In Advances in Neural Information Processing Systems 16: Proceedings of The 2003 Conference (NIPS-03*. MIT Press, 2004.

[9] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, P. Liu, and K. Lee, "Bio-Inspired Multi-Agent Collaboration for Urban Monitoring Applications," *Bio-Inspired Computing and Communication*, vol. 5151, no. 3, pp. 204–216, 2008.

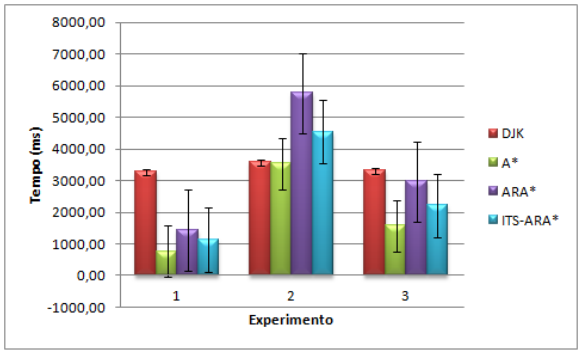
[10] C. Li, S. Anavatti, and T. Ray, "Adaptive Route Guidance System with Real-Time Traffic Information," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 367–372.

[11] L. Moura, M. Ritt, and L. S. Buriol, "Estudo Experimental de Algoritmos em Tempo Real de Caminho Mínimo Ponto a Ponto em Grafos Dinâmicos," in *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*, ser. SBPO, 2010.

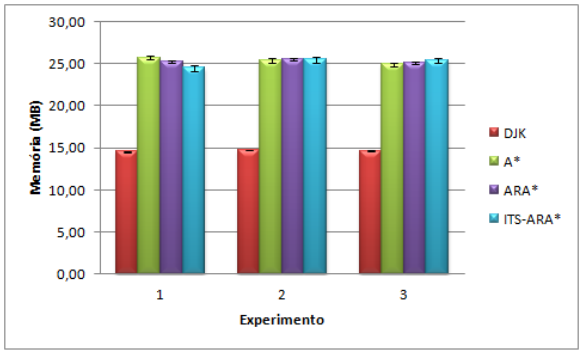
[12] M. Likhachev, "Search-based Planning for Large Dynamic Environments," Ph.D. dissertation, School of Computer Science – Carnegie Mellon University, 2005.

[13] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems," *AI Magazine*, vol. 17, no. 3, pp. 73–83, 1996.

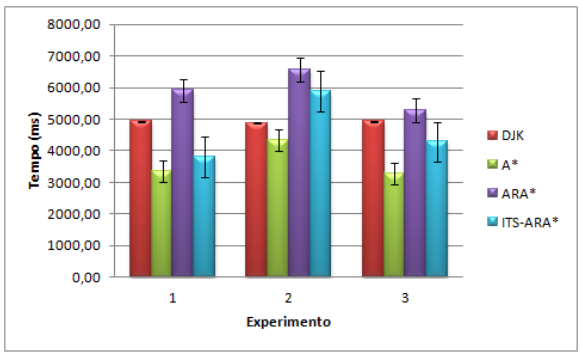
[14] "GraphStream Project," 2013, http://graphstream-project.org/.



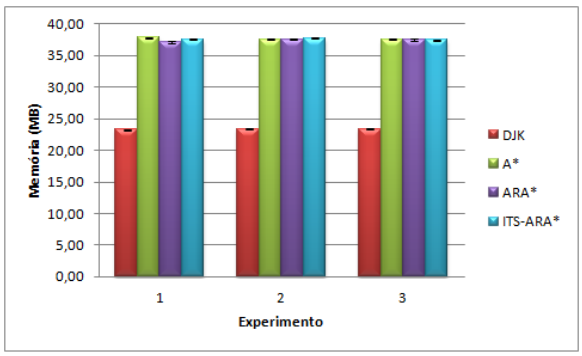
(a) Manaus – Tempo de Execução (ms).



(b) Manaus – Memória Consumida (MB).

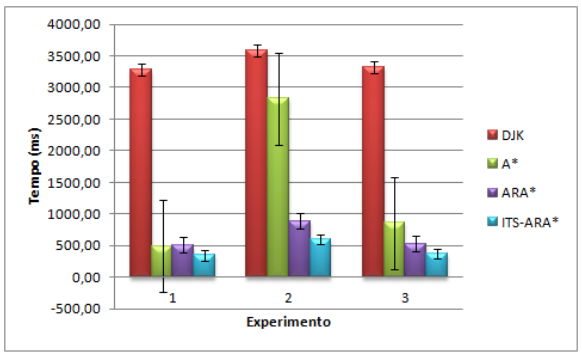


(c) Rio de Janeiro – Tempo de Execução (ms).

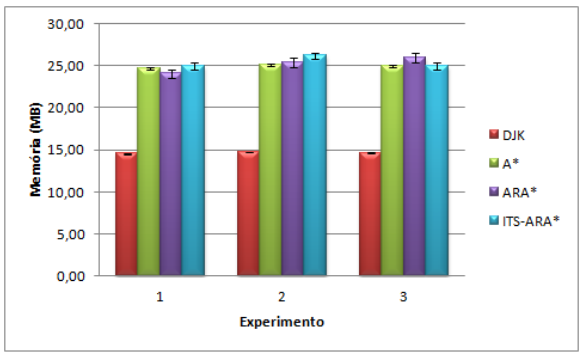


(d) Rio de Janeiro – Memória Consumida (MB).

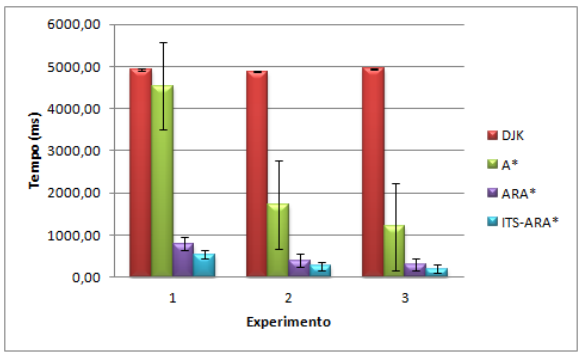
Fig. 2. Tempo de Execução e Memória Consumida sem heurística.



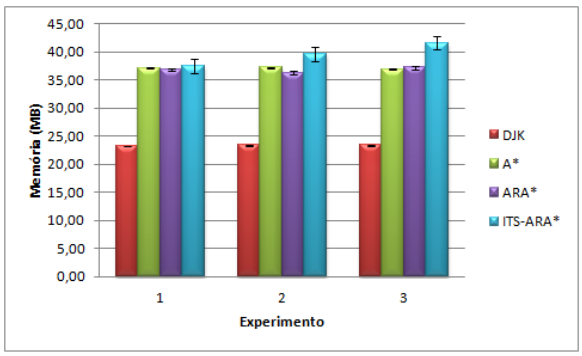
(a) Manaus – Tempo de Execução (ms).



(b) Manaus – Memória Consumida (MB).



(c) Rio de Janeiro – Tempo de Execução (ms).



(d) Rio de Janeiro – Memória Consumida(MB).

Fig. 3. Tempo de Execução e Memória Consumida com Heurística.