XoTransito: Utilização de Algoritmos Adaptativos no Planejamento Automático de Rotas no Trânsito

F. S. Komori, R. O. Morelli, T. B. Torres e T. Matos

***Resumo*—Este artigo tem como objetivo apresentar o Xo-Transito, um sistema automático inteligente para planejamento de rotas viárias em áreas urbanas utilizando algoritmos adaptativos.**

Esse sistema foi proposto pelos autores como projeto de formatura do curso de Engenharia Elétrica com Ênfase em Computação. O projeto foi desenvolvido no Laboratório de Linguagens e Técnicas Adaptativas sob a orientação do Prof. Dr. João José Neto. Dados os pontos de origem e de destino, o sistema planeja rotas no mapa da cidade de São Paulo levando em consideração o trânsito no momento e outros fatores relevantes para o fluxo de veículos.

***Index Terms*—algoritmos adaptativos, estratificação, planejamento automático de rotas, tecnologia adaptativa.**

I. INTRODUÇÃO

O sistema XoTransito foi proposto e desenvolvido pelos autores como projeto de formatura do curso de Engenharia da Computação no ano de 2008. A proposta do trabalho foi criar um sistema que planejasse uma rota entre duas localizações da cidade de São Paulo levando em conta variáveis importantes, como o trânsito e a velocidade permitida nas vias. Esses eram considerados durante a execução do algoritmo de roteamento do sistema.

O algoritmo central utilizado no trabalho para a geração de rotas foi uma variação do clássico algoritmo A* [1]. Foram propostas alterações no algoritmo de roteamento original para torná-lo adaptativo e aproveitar as vantagens que a tecnologia adaptativa oferece para aperfeiçoar a geração de rotas de trânsito, permitindo tratar situações diversas, como congestionamentos e velocidade de vias.

Foi implementada também uma estratégia de estratificação do mapa utilizado no sistema, com a finalidade de criar um algoritmo hierárquico de roteamento, o que permite resolver o problema de forma estruturada, partindo do contexto geral e progredindo para o particular. Na estratificação, as vias da cidade são classificadas em três níveis hierárquicos. Essa técnica foi utilizada visando também economia de recursos computacionais, pois, ao classificar as vias do mapa em diversas camadas, apenas as vias de determinada camada (ou nível) são utilizadas pelo algoritmo a cada momento.

II. ALGORITMOS DE ROTEAMENTO CLÁSSICOS

No projeto, foram analisados diversos algoritmos de roteamento clássicos para serem utilizados como estrutura central do algoritmo adaptativo. Alguns algoritmos pesquisados foram: busca em largura (*breadth-first search*), busca em profundidade (*depth-first search*), busca bidirecional, busca de custo uniforme, *greedy best-first search* e A* (que também é um algoritmo do tipo *best-first search*). Tais algoritmos estão descritos em [1] e [2].

Dentre esses algoritmos clássicos, o A* recebe destaque, uma vez que é completo (sempre encontra uma solução caso exista) e ótimo (encontra a solução com o menor custo possível), desde que a heurística utilizada seja admissível (nunca superestima o custo real) e monotônica (sempre decresce com o aumento do custo). Além disso, é otimamente eficiente, ou seja, nenhum outro algoritmo ótimo garante visitar um menor número de nós do que A*. Entretanto, a aplicação desse algoritmo em sua forma original neste projeto não é viável, pois o desempenho é prejudicado para rotas entre pontos muito distantes (nos testes realizados, caminhos da ordem de 100 quarteirões foram calculados após visitarem-se mais de 8000 nós). Além disso, não é um requisito essencial que a busca encontre o caminho ótimo, já que o fator trânsito, variável e algumas vezes imprevisível, é também considerado.

O algoritmo A* se norteia basicamente por uma função $f(n) = g(n) + h(n)$, na qual $g(n)$ representa o custo do caminho da origem até o nó n (nó que está sendo avaliado) e $h(n)$ é uma função heurística que estima o custo do nó n até o destino. Dessa forma, o próximo nó a ser visitado é aquele contido na lista aberta (lista que contém os nós que ainda não foram visitados) que possua o menor valor de f. De modo simples, o A* funciona como uma busca de custo uniforme que utiliza uma heurística para determinar quais nós têm maior probabilidade de conduzir ao destino por um caminho de baixo custo, de forma a expandir menos nós durante a busca.

III. ALGORITMO DE ROTEAMENTO ADAPTATIVO

Foram implementadas três modificações no algoritmo A* e no grafo representativo do mapa buscando construir o algoritmo adaptativo. As modificações foram:

estratificação do mapa, consideração da situação do trânsito em tempo real e adaptatividade dos componentes do algoritmo A* (variação dos valores da função h).

Figura 5. Autômato de mudança de nível de ruas.

A. *Estratificação*

Considerando-se que o desempenho do algoritmo A* é baixo (da ordem de dezenas de segundos, utilizando-se os recursos computacionais atuais) para rotas longas (com mais de 50 quarteirões, aproximadamente, entre a origem e o destino), foi aplicada a estratificação do mapa de ruas de São Paulo.

Isso acarretou a criação de um mapa hierárquico, ou seja, um mapa separado em diversos níveis de detalhamento. Quanto maior o nível da observação, menor é o detalhamento do mapa (mapa com menor quantidade de ruas visíveis pelo algoritmo de roteamento do projeto).

No projeto, os níveis utilizados foram os seguintes: no nível mais alto, se encontram as vias rápidas (como as Marginais); no nível intermediário, se encontram as vias de velocidade intermediária (como a Avenida Nove de Julho); no nível mais baixo, se encontram as ruas de baixa velocidade. Desse modo, cada nível representa um detalhamento diferente do mapa, estruturando-o em camadas. A partir disso, o algoritmo de roteamento altera o nível do mapa dinamicamente, dependendo do progresso da análise de nós. Considerando essa abordagem, o algoritmo avalia todas as ruas somente no início e no fim do percurso. No meio do caminho, as ruas de menor velocidade são inicialmente descartadas com o objetivo de otimizar o desempenho. Ao final da execução desse passo, o caminho encontrado é avaliado e, caso possua muitos trechos congestionados, uma nova instância do A* é executada. Essa segunda execução somente é efetuada nos trechos mais críticos em termos de trânsito. A utilização desse algoritmo permite uma maior agilidade no processamento do algoritmo, uma vez que não verifica todos os nós possíveis no meio do caminho.

Considerando que o mapa é visto como um conjunto de regras (para cada par de nós, define se existe ou não uma aresta que os une) e que essas regras são alteradas durante a execução do algoritmo, pode-se afirmar que a estratificação permite a utilização da adaptatividade no algoritmo.

1) Exemplo de Estratificação: Considera-se o caminho entre o Tatuapé e o Butantã, uma rota longa que passa por diversas vias principais, como a Marginal Tietê, e avenidas, como a Salim Farah Maluf. Essa rota, caso fosse gerada com o algoritmo A* adaptativo com estratificação, consideraria no início todas as possibilidades (ruas, avenidas e vias principais), visando chegar a uma via de nível superior. A partir de então, consideraria apenas as vias principais e tentaria chegar ao destino utilizando somente o nível em que está ou um nível mais alto de estratificação. Ou seja, quando o algoritmo encontra uma via de nível superior ao que está atualmente, ele muda seu conjunto de regras para apenas considerar as ruas do nível superior que encontrou, diminuindo, assim, o número de nós e arestas do grafo a percorrer.

Resumindo, ele se utiliza do autômato da figura 1 para trocar entre níveis.

B. *Trânsito em Tempo Real*

Para se aproximar da realidade, o algoritmo pode considerar uma base de dados estatística (que armazena um conjunto de situações de trânsito para as ruas monitoradas em diferentes horários do dia), com o objetivo de utilizar, no grafo, um perfil de trânsito mais coerente com o momento real de percurso. Nesse sentido, o sistema considera que o motorista demora certo tempo para percorrer sua rota (que pode ser bem considerável em certos horários).

Uma abordagem como essa força o algoritmo a ser executado em pequenos trechos do percurso desejado, sendo que, a cada trecho, uma atualização no grafo de ruas é executada, com o objetivo de representar os pesos referentes ao trânsito relativos ao momento estimado em que o motorista estará naquele ponto. Adotando esse procedimento, o funcionamento do algoritmo se aproxima de uma situação ideal, pois passa a atender a grandes variações da situação do trânsito nas vias da cidade de São Paulo.

1) Exemplo de Trânsito em Tempo Real: Considera-se uma rota longa, de 25 km de extensão, ligando dois pontos distantes da cidade. É razoável considerar que, em São Paulo, essa rota demore de 1 a 2 horas para ser percorrida, e que, durante esse tempo, o trânsito varie diversas vezes no decorrer do trajeto. Caso o caminho seja traçado com o algoritmo A* adaptativo por temporização, a extensão da rota seria detectada (por exemplo, mais de 20 km), o que permite concluir que poderia haver modificação na configuração original do trânsito. Desse modo, o algoritmo calcularia a rota com o trânsito atual até certo ponto do trajeto (por exemplo, 15 km a partir do ponto inicial). Ao chegar nessa situação, o algoritmo poderia carregar dados históricos de trânsito para a próxima hora, atualizando o peso das arestas e traçando a rota entre os pontos novamente.

C. *Adaptatividade dos Componentes do A**

O algoritmo A* funciona com um grafo ponderado e

uma função heurística. Pode-se adicionar adaptavidade para esses dois elementos.

O grafo ponderado pode ser submetido a operações de inclusões e exclusões de arestas. Tais ações são realizadas, respectivamente, em casos de vias muito congestionadas (vias em que o fator de congestionamento é maior que 7, numa escala de 0 a 10) e no momento em que essas passam a ter um trânsito melhor (o fator de congestionamento é menor que 7). Outra abordagem adaptativa consiste na alteração dinâmica dos pesos considerados para a rota, modificando seu grau de contribuição para o custo total. Assim, fatores como trânsito e velocidade da via podem ter sua importância modificada no cálculo do custo total.

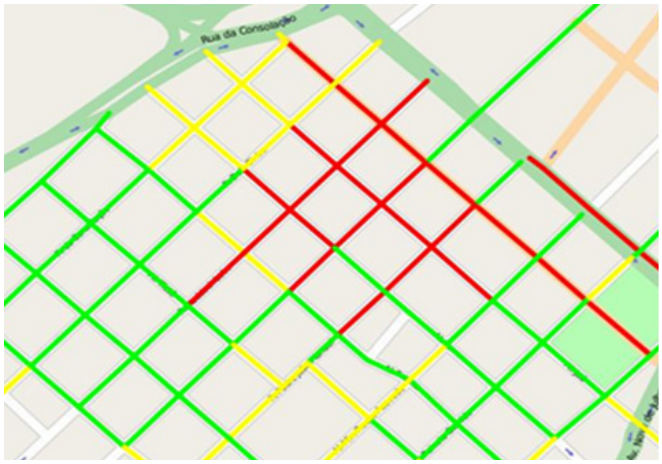

Figura 6. Simulação de trânsito local.

Figura 7. Rota traçada sobre mapa da figura 2.

Para adicionar adaptativade à heurística, é possível modificar os parâmetros da função que avalia o possível custo do nó atual até o destino. A cada invocação da função heurística, o cálculo varia de acordo com o contexto em que está sendo executado.

IV. INTERFACE COM O USUÁRIO

O sistema utiliza o *framework* e os mapas do OpenStreetMap. O OpenStreetMap é um projeto de código aberto que provê mapas de ruas e interface gráfica para visualização desses. Na base de dados geográficos, as vias (ruas, avenidas, pontes, ferrovias, etc.) são representadas através de conjuntos de nós, cada um especificado com latitude e longitude. Na interface do sistema com o usuário, que é baseada na Web, o mapa de São Paulo é apresentado. A partir dele, o usuário clica em dois pontos distintos no mapa e é traçada uma rota entre esses pontos. Caso haja trânsito, o algoritmo desvia do mesmo, passando preferencialmente pelas vias principais e avenidas.

Para representar o trânsito, foram escolhidas as cores vermelho (trânsito ruim), amarelo (trânsito mediano) e verde (trânsito bom). Na figura 2, é apresentado um mapa de uma região de São Paulo, em que o trânsito das ruas é simulado. Na figura 3, encontra-se um exemplo de rota (em azul), que procura evitar o trânsito.

V. ALGORITMO EM PSEUDO-CÓDIGO

A seguir é descrito o código implementado para o Algoritmo Adaptativo proposto nesse projeto.

```
função A*_geral(vértice_origem, vértice_destino,
horário) {
  caminho = A*(vértice_origem, vértice_destino,
  verdadeiro, horário);
   se peso_médio_trânsito(caminho) alto:
     para cada subcaminho de
     subcaminhos_alto_trânsito(caminho):
       novo_subcaminho =
       A*(vértice_inicio(subcaminho),
       vértice_fim(subcaminho), falso,
       horário[vértice_inicio(subcaminho)]);
        incorporar(novo_subcaminho, caminho);
   retorna caminho;
}
```

Um exemplo de implementação seguindo as idéias expostas é mostrado a seguir. O algoritmo principal é o `A*_geral`, que invoca o A* e as funções auxiliares:

- `caminho`: retorna uma lista de vértices que compõem o caminho encontrado;
- `vizinhos`: retorna um conjunto de vértices vizinhos considerando-se o nível passado como parâmetro (se não encontrar nada, modifica o nível até que encontre algum vértice vizinho);
- `decisão`: função que avalia se o determinado vértice vai ser processado ou não (arestas com um trânsito muito congestionado são desconsideradas, como se tivessem sido removidas do grafo);
- `peso`: calcula o custo total da aresta, considerando-se diversos aspectos: comprimento, trânsito e velocidade;
- `duração`: calcula uma duração estimada para o percurso entre os 2 vértices passados como parâmetros com base em banco de dados estatísticos;
- `heurística`: retorna uma estimativa de custo entre o vértice processado e o destino, considerando a distância em linha reta entre os dois vértices e

dados estatísticos de trânsito na região.

VI. CONCLUSÃO

Este artigo apresentou um sistema para o planejamento de rotas de trânsito para a cidade de São Paulo. A tecnologia adaptativa foi de grande importância nesse trabalho, pois facilitou a consideração de fatores adicionais aos pesos geralmente usados em algoritmos de roteamento clássicos, como o A*. Entre esses fatores, estão o trânsito e a velocidade nas vias, os quais são introduzidos em tempo real no modelo do grafo.

Com a adição desses fatores na geração da rota, o sistema passou a apresentar caminhos adequados a um motorista que irá trafegar no trânsito real da cidade de São Paulo, em que há variação das condições do trânsito durante o percurso do motorista. Os resultados obtidos foram adequados, visto que os dados de trânsito, fornecidos em etapas para o algoritmo, simulam a variação contínua do trânsito.

REFERÊNCIAS

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions of Systems Science and Cybernetics*, vol. 4, no. 2, 1968.

[2] S. Russell and P. Norvig, *Artificial Intelligence*, 2nd ed. New Jersey: Prentice Hall, 2003.

[3] H. Pistori and J. J. Neto, "Adaptree," *Conferência Latino Americana de Informática (CLEI)*, 2002.

[4] J. J. Neto, "Adaptatividade e tecnologia adaptativa," *Revista IEEE América Latina*, vol. 5, no. 7, 2007.

[5] H. Pistori, "Tecnologia adaptativa em engenharia de computação," *Tese de Doutorado*, 2003.