UNIVERSIDADE SAGRADO CORAÇÃO

CARLOS EDUARDO SIMÕES DOS SANTOS

SISTEMA DE RE-ROTEAMENTO DE ÔNIBUS URBANOS UTILIZANDO O ALGORITMO DE COLÔNIA DE FORMIGAS

Projeto de Trabalho de Conclusão

BAURU

2011

CARLOS EDUARDO SIMÕES DOS SANTOS

carlosliberi@gmail.com

SISTEMA DE RE-ROTEAMENTO DE ÔNIBUS URBANOS UTILIZANDO O ALGORITMO DE COLÔNIA DE FORMIGAS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação sob orientação do Prof. M.Sc. Patrick Pedreira Silva.

Professor Orientador: Patrick Pedreira Silva

Bauru, dezembro de 2011

Santos, Carlos Eduardo Simões dos

S2373s

Sistema de re-roteamento de ônibus urbanos utilizando o algoritmo de colônia de formigas / Carlos Eduardo Simões dos Santos -- 2011.

50f. : il.

Orientador: Prof. Ms. Patrick Pedreira da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Sagrado Coração - Bauru - SP

1. Roteamento. 2. Ant colony system. 3. Problema de roteamento de veículos. 4. PRV. 5. Colônia de formigas. I. Silva, Patrick Pedreira da. II. Título.

CARLOS EDUARDO SIMÕES DOS SANTOS

SISTEMA DE RE-ROTEAMENTO DE ÔNIBUS URBANOS UTILIZANDO O ALGORITMO DE COLÔNIA DE FORMIGAS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais e Aplicadas da Universidade Sagrado Coração como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação sob orientação do Prof. Ms. Patrick Pedreira Silva.

_____

Prof. Ms. Patrick Pedreira Silva
Universidade Sagrado Coração

_____

Prof. Ms. Anderson Francisco Talon
Universidade Sagrado Coração

_____

Prof. Ms. Patricia Bellin Ribeiro
Universidade Sagrado Coração

Dedico este trabalho a minha família e amigos.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, que me deram suporte durante toda minha vida. A minha irmã, que sempre esteve lá para me animar e descontrair. Ao meu avô que me proporcionou estudo de qualidade e uma chance de poder ser alguém na vida. Aos amigos que fiz no meu primeiro emprego, e ao meu primeiro chefe que sempre me ensinou muito e nunca me negou tempo de estudo. Aos amigos que conheci durante o curso, que aprendi a gostar durante os anos que convivemos diariamente. Aos amigos que me acompanham a anos, que sempre estiveram lá pra ouvir meus lamentos e minhas vitórias. Quero agradecer a meu orientador, que me ajudou muito no desenvolvimento deste trabalho. Muito obrigado.

“Cada jogador precisa aceitar as cartas que a vida lhe dá, mas cabe a cada um escolher o que fazer com elas uma vez que as tem na mão.” (Voltaire).

LISTA DE ILUSTRAÇÕES

Figura 1 14

Figura 2 19

Figura 3 22

Figura 4 23

Figura 5 25

Figura 6 26

Figura 7 31

Figura 8 33

Figura 9 35

Figura 10 36

Figura 11 37

Figura 12 37

Figura 13 38

Figura 14 39

Figura 15 40

Figura 16 40

Figura 17 41

Figura 18 42

Figura 19 43

Figura 20 44

Figura 21 44

Figura 22 45

Figura 23 46

LISTA DE TABELAS

Tabela 1 17

Tabela 2 25

Tabela 3 25

Tabela 4 27

Tabela 5 39

Tabela 6 41

RESUMO

Todo o Brasil vive com um grave problema de transporte de pessoas e carga. Pouco se investe nesse setor e poucas soluções dinâmicas e diferenciadas são criadas. Com o crescer constante dos grandes centros metropolitanos, faz-se necessário a criação de alternativas. Esse efeito de inchaço das grandes cidades já é notado até mesmo em cidades de menor porte como, por exemplo, a cidade Bauru no estado de São Paulo. Uma alternativa possível é o re-roteamento dos trajetos feitos pelos veículos a fim de evitar pontos de gargalo. Para aplicar tal solução, é preciso encontrar ferramentas que permitam o realocamento das rotas. Este estudo verificou a viabilidade da aplicação do algoritmo *Ant Colony Optimization* (ACO) para uma versão simplificada deste problema, utilizando como ferramentas para o desenvolvimento de um protótipo a linguagem JAVA e a API do GoogleMaps. Os resultados obtidos corroboram a utilidade do ACO como uma alternativa viável de resolução de problemas de roteamento de veículos.

Palavras-chave: roteamento, Ant Colony System, Problema de Roteamento de Veículos, PRV, colônia de formigas.

SUMÁRIO

1. INTRODUÇÃO 13

1.1 OBJETIVOS 16

2. FUNCIONAMENTO DO *ANT COLONY OPTIMIZATION* 17

2.1. A META-HEURÍSTICA ACO – FORMULAÇÃO MATEMÁTICA 19

2.2. EXEMPLO DE APLICAÇÃO DO ANT COLONY OPTIMIZATION AO PROBLEMA DO CAIXEIRO VIAJANTE 21

3.2.1 ACO EM FUNCIONAMENTO NO PCV (MODELAGEM MATEMÁTICA) 23

4. TRABALHOS CORRELATOS 28

***5. METODOLOGIA* 30**

5.1 FERRAMENTAS UTILIZADAS NO PROJETO 32

5.1.1 API GOOGLEMAPS 32

5.1.2 JAVA E NETBEANS 33

6 RESULTADOS 35

6.1 ACO EM FUNCIONAMENTO NO PCV 35

6.3 APLICAÇÃO EM UM CENÁRIO REAL 43

7 CONCLUSÃO 47

8 BIBLIOGRAFIA 49

APÊNDICE A - ARTIGO CIENTÍFICO 50

1. INTRODUÇÃO

A busca pela resolução de problemas complexos de modo mais simples e eficiente tem levado pesquisadores a propor novos métodos, dentre os quais podemos citar as técnicas de otimização inspiradas na natureza, a exemplo da inteligência de enxames. Esta técnica baseia-se na idéia de cooperação direta ou indireta entre indivíduos que, juntos, conseguem melhor adaptação ao ambiente. A Otimização por Colônia de Formigas (ACO, do inglês *Ant Colony Optimization*), inspirada no comportamento de formigas artificiais (agentes) em busca de alimentos, é um exemplo desta técnica de inteligência de enxames. Esta forma de otimização proposta por Dorigo e Gambardella (1997), baseia-se na comunicação dos indivíduos realizada indiretamente por meio de uma trilha feita por formigas artificiais durante a exploração do espaço de busca. A trilha é definida pelo feromônio artificial que atrai as outras formigas, servindo como um meio de troca de informações sobre a experiência no espaço de busca. Esta técnica é considerada robusta e eficiente já que é um processo de aprendizado distribuído, onde cada agente simples (formigas artificiais) trabalha de modo coletivo, promovendo adaptações e modificações no ambiente, contribuindo para a solução final do problema.

O ACO possui um vasto campo de aplicações, podendo ser utilizada na resolução de vários problemas de complexidade exponencial dos quais se pode citar, por exemplo, o Problema do Caixeiro Viajante, o Problema de Geração de Horários e o Problema de Roteamento de Veículos (PRV). O PRV é, inclusive, atualmente um dos problemas que mais merecem destaque uma vez que, nas últimas décadas, houve um crescimento significativo do número de veículos trafegando pelas avenidas e ruas das cidades, provocando engarrafamentos, maior consumo de combustíveis e aumento de tempo para se percorrer um determinado trajeto. Hoje em dia, nota-se que a frota de veículos cresce de forma desenfreada em várias cidades brasileiras. Em fevereiro de 2008, a cidade de São Paulo atingiu o espantoso número de seis milhões de veículos (BARTHOLOMEU, 2008). A quantidade absurda de carros, motos, ônibus e outros trazem vários problemas. Todos esses veículos são movidos a combustível que, fósseis ou não, liberam gases nocivos ao meio ambiente, ocasionando um grande impacto ambiental, doenças para a população (problemas respiratórios, estresse, etc.), aumento do calor e outras conseqüências. Na grande São Paulo

muitas medidas já foram tomadas para tentar amenizar este grave problema, dentre as principais destacam-se: o rodízio de veículos, a criação de vias alternativas (Rodoanel[1], por exemplo) e até mesmo a proibição de tráfego de veículos de grande porte (G1, 2010). E esse problema, ainda que em menor proporção, atinge também cidades de porte médio, por exemplo, em Bauru, cidade do interior paulista com cerca de 335.000 habitantes segundo último censo do IBGE, a frota já chega a mais de 236.000 automóveis (OLIVEIRA, 2010).

Diante deste crescimento das frotas de veículos, alternativas devem ser criadas de modo a contribuir para que o trânsito das cidades médias não se torne tão problemático quanto o das grandes metrópoles. Com base nesta possibilidade, este trabalho propõe utilizar a tecnologia como uma alternativa de combate a este problema, considerando para este estudo a cidade de Bauru e sua frota de ônibus (Figura 1) destinada ao transporte público de passageiros.

Figura 1 Exemplo de ônibus que transita em Bauru

Por não ser um grande centro comercial e industrial como São Paulo, as soluções para o problema local não podem demandar grandes quantidades de capital. Segundo afirmação feita por Mondelli, presidente da Emdurb (*EMPRESA* MUNICIPAL DE DESENVOLVIMENTO URBANO E RURAL DE *BAURU*), em (Oshiro, 2011) a cidade de

[1] O projeto do Rodoanel de São Paulo tem como principal objetivo ordenar o tráfego rodoviário de entrada e saída da cidade - uma região onde vivem mais de 10 milhões de habitantes -, calculado em 1,1 milhão de veículos por dia, além de disciplinar a circulação de caminhões rodoviários de grande parte na cidade - http://www.rodoanel.sp.gov.br/

Bauru cresceu muito de 2003 até hoje e a abertura de uma grande empresa pode aumentar a demanda de uma rota específica. Mondelli ainda afirma que uma das soluções seria o re-roteamento de algumas linhas de ônibus. Em 2009 a cidade de Bauru contava com uma frota de 234 ônibus (Jornal de Limeira, 2009), que já não era suficiente para transportar parte da população bauruense que usa o transporte público como principal meio de locomoção. O atraso é um problema com o qual todos que precisam do transporte público bauruense sofrem. A falta de rotas alternativas na hora o rush e, em casos extraordinários (enchentes e acidentes), afeta diretamente a qualidade do transporte. Nenhum plano de contingência existe, ou seja, os ônibus ficam presos nos engarrafamentos sem nenhuma saída ou alternativa. A única solução é aguardar ou aumentar o número de ônibus em uma mesma rota para atender a demanda. Este tipo de problema é estressante não só para quem espera durante muito tempo no ponto de ônibus para ir trabalhar, estudar ou voltar para casa mais também é cansativo para os motoristas e demais funcionários. Além disso, existe um custo financeiro associado, considerando o custo do ônibus parado, o combustível gasto e da menor quantidade de pessoas transportadas. Com a otimização das rotas existentes e a criação de rotas alternativas os gastos possivelmente diminuiriam, além de trazer inúmeros benefícios diretos para a população usuária do transporte público.

Este fato, associado ao anseio por parte das empresas de minimizar os custos no transporte de passageiros, tem despertado o interesse de inúmeros pesquisadores em estudos nessa área, na busca de soluções ótimas para os problemas de roteamento de veículos (PRV). Diante disso, a proposta deste projeto é utilizar-se do Algoritmo de Otimização de Colônia de Formigas como base para o desenvolvimento de um software que simule a criação de rotas alternativas para um sistema de transporte público de passageiros, tendo como caso de estudo a cidade de Bauru.

1.1 OBJETIVOS

O objetivo geral deste trabalho é o estudo e aplicação da heurística de Otimização por Colônias de Formigas (ACO) em uma versão simplificada do problema de construção de rotas para os veículos que fazem o transporte público na cidade de Bauru, SP, verificando a possibilidade de se oferecer, por meio os resultados obtidos, uma nova alternativa à solução adotada atualmente pela empresa responsável pelo sistema de transporte.

Os objetivos específicos no desenvolvimento deste projeto são:

- Implementar o algoritmo para a resolução do problema de rotas em linguagem de programação JAVA;
- Utilizar o Google Maps para criar uma interface simples e funcional com a ferramenta desenvolvida;
- Realizar testes com o sistema desenvolvido, considerando uma versão simplificada do problema proposto e do problema do Caixeiro Viajante;
- Avaliar o desempenho da heurística proposta para a resolução do problema descrito;

2. FUNCIONAMENTO DO *ANT COLONY OPTIMIZATION*

Segundo Dorigo e Gambardella (1997) o ACO é um modelo de algoritmo baseado no comportamento natural de formigas. Neste modelo, as formigas trabalham como agentes para o bem da colônia em geral. Criar um algoritmo similar exige a compreensão do sistema como um todo. A seguir vemos a Tabela 1 que estabelece uma comparação entre os aspectos biológicos e artificiais:

Tabela 1 - Comparação entre os Agentes Biológicos e Artificiais.

Agentes biológicos	Agentes Artificiais
Formigas	Agentes
Trilhas	Caminho
Intersecção da trilha	Pontos de parada
Concentração de feromônio	Pesos probabilísticos
Deposição feromônio	Incremento no peso
Evaporação de Feromônio	Decremento no peso

As formigas são os agentes que "caminham" nas trilhas em busca de um melhor caminho. Esses agentes são postos em um ponto inicial x e tem como objetivo chegar até um ponto final y. As formigas escolhem o caminho de forma aleatória, dessa forma, elas estão sempre caminhando rumo ao objetivo, havendo uma solução (mesmo que não ótima). Durante o percurso, os agentes irão depositar feromônio no decorrer da trilha como forma de indicar para os próximos agentes qual caminho tomar. Os feromônio depositados são responsáveis por atribuir peso a essas trilhas, fazendo com que os caminhos com maior peso tenham maior probabilidade de serem escolhidos pelos próximos agentes. Durante o trajeto, para poderem atingir o objetivo final, os agentes passam por intersecções entre as trilhas. Essas intersecções, que são consideradas pontos de parada obrigatória, podem variar em termos de quantidade e distância entre si.

O que faz esse algoritmo se tornar único é a forma como o feromônio é depositado: de modo aleatório no percurso e em uma quantidade baseada em um peso probabilístico.

Quanto mais rápido for o caminho, mais rápido o agente retorna e preenche a trilha com feromônio, logo a próxima formiga tenderá a pegar o caminho com maior quantidade de feromônio. Os agentes, durante a execução do algoritmo, estarão sempre se movendo pelas rotas e sempre encontrando novas soluções. Por esse motivo o resultado, isto é, o caminho mais rápido, poderá ser alterado.

As regras seguidas pelas formigas podem ser simplificadas em dois passos: (GARBE, B. *apud* COSTA, A., 2006):

- Formigas depositam feromônio a uma taxa aproximadamente constante enquanto viajam.
- Formigas são capazes de detectar diferenças na concentração de feromônio nas suas proximidades, e tendem a se mover na direção em que concentração é maior.

Na Figura 2 observa-se o funcionamento do algoritmo de uma forma bem simples: em (a) pode-se ver que existem duas rotas (A e B) para que as formigas cheguem ao objetivo (representado por uma fruta). Em (b) a formiga que usa a rota B chega primeiro no objetivo, portanto voltará primeiro ao ponto inicial. Em (c) vê-se que a próxima formiga tende a escolher a rota com mais feromônio (pontos azuis). Em (d) observa-se que mais formigas usam a rota B por ela ser mais rápida, dessa forma, mais feromônio é despejado e, portanto, mais formigas percorrerão esse caminho.

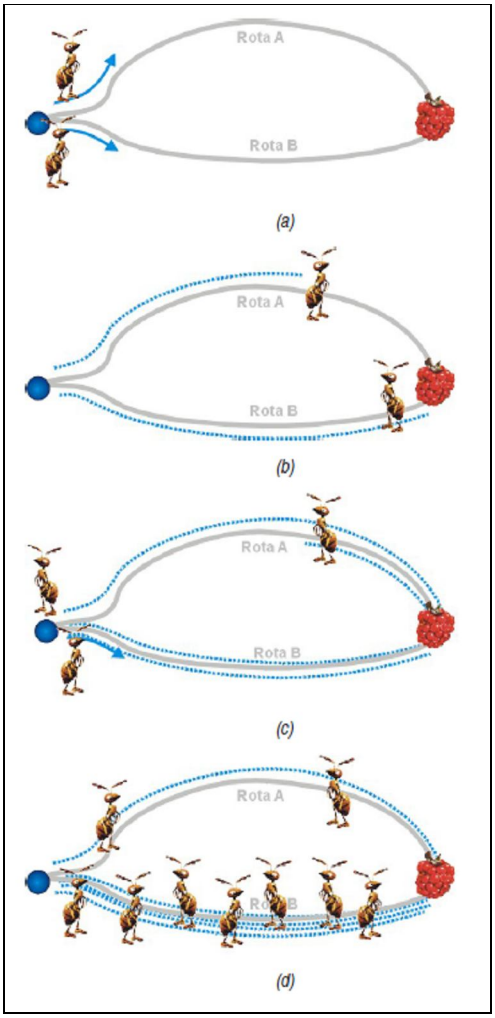

Figura 2 Comparação entre o Biológico e o Artificial(GARBE, 2006).

2.1. A Meta-Heurística ACO – Formulação Matemática

O ACO em sua primeira versão (DORIGO, 2004), inicia-se com cada agente (formiga) construindo uma solução própria para o problema baseada em soluções de agentes que já passaram por esse mesmo problema. Cada formiga *k* se move em um caminho onde os movimentos são selecionados segundo uma distribuição de probabilidade dada por (BABA, Cristina Mayumi ET AL, 2004):

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{il}]\cdot[\eta_{il}]^{\beta}}{\sum_{l \in J_i^k}[\tau_{il}]\cdot[\eta_{il}]^{\beta}} & \text{se } j \in J_i^k \\ 0 \end{cases}$$

(1)

Onde:

$_{ij}p_{k}$ = probabilidade da formiga *k*, que se encontra em *i*, escolher o nó *j* como próximo nó a ser visitado;

τ_{ij} = quantidade de feromônio existente no arco (*i*,*j*). Inicialmente, adota-se um mesmo valor τ_0 para todos os arcos da rede;

η_{ij} = função heurística que representa a atratividade do arco (*i*,*j*).

$_{i}J_{k}$ = conjunto de pontos ainda não visitados pela formiga *k*, que se encontra atualmente no ponto *i*;

β = valor heuristicamente escolhido, que pondera a importância da quantidade de feromônio existente no arco em relação à distância entre os nós *i* e *j*. (BABA, Cristina Mayumi ET AL, 2004).

A expressão acima faz com que a agente escolha o caminho com maior quantidade feromônio, ou seja, o trajeto que foi mais usado por outros agentes (que naquele momento é o caminho mais curto). Deve ser estipulado um ciclo de atualização de feromônio, uma vez que se os feromônio não tiverem seu peso diminuído, o custo computacional para que o agente escolha o menor caminho aumentará e o algoritmo perderá em desempenho. Assim, para cada clico de atualização (i,j), adiciona-se uma quantidade de feromônio proporcional ao tamanho da rota obtida (BABA, Cristina Mayumi et al, 2004):

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{melhor}$$

(2)

$$\Delta\tau_{ij}^{melhor} = \begin{cases} \dfrac{1}{L_{melhor}} \\ 0 \end{cases}$$

(3)

A Equação (2) visa a diminuição do peso do feromônio, onde ρ é o fator que determina a velocidade da diminuição do peso. Já a próxima Equação (3) é responsável por aumentar o peso do feromônio onde L é a distância total percorrida na rota construída pelo melhor agente da interação (BABA, 2004). A seção seguinte, mostra a aplicação deste algoritmo ao problema do Caixeiro Viajante como forma de ressaltar o uso potencial desta teoria na resolução de problemas de otimização, sobretudo aqueles que lidam com problemas de roteamento.

2.2. Exemplo de aplicação do *Ant Colony Optimization* ao Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um excelente exemplo do funcionamento do ACO, visto que é um problema de otimização NP-díficil, servindo como teste para aplicação de novos algoritmos, podendo auxiliar na prova da sua utilidade. Um problema de otimização NP-díficil é frequentemente usado para definir problemas de tomada de decisão (THOMAS, 2001). Alia-se a isso o fato de o PCV ser um problema de fácil percepção, o que faz com que a linha de desenvolvimento de um algoritmo não se altere em aspectos técnicos. O Problema do Caixeiro Viajante é descrito do seguinte modo: dado um conjunto de *n* cidades e uma matriz de distâncias, o objetivo é encontrar um caminho que tenha a menor distância a ser percorrida para que sejam visitadas todas as cidades passando exatamente uma única vez em cada cidade, retornando à cidade de origem.

No algoritmo ACO, as formigas são simples agentes que, no caso do PCV, constroem circuitos através do movimento entre cidades no grafo do problema. A solução construída pelas formigas é elaborada utilizando trilhas de feromônio (artificiais) e pela disponibilidade de informação heurística. Quando o algoritmo ACO é aplicado é associada uma força do feromônio que é modificada durante o algoritmo. Inicialmente, cada uma das *n* formigas é colocada numa cidade (ponto inicial), de forma aleatória, aplicando-se depois, de modo iterativo, uma regra de transição de estado para cada uma das cidades.

Uma formiga constrói um circuito da seguinte forma: em uma cidade inicial *i,* ela escolhe uma cidade *j* que ainda não tenha visitado. Esta escolha é feita probabilisticamente segundo a força do feromônio $\tau_{ij}(t)$ no arco entre as cidades *i* e *j*, e a informação heurística disponível localmente, que é função do comprimento do arco. As formigas, de um modo probabilístico, preferem as cidades que estejam mais próximas e ligadas por arcos com grande força de feromônio. Para construir uma solução viável, cada formiga possui uma forma de memória limitada, onde é guardado o corrente circuito parcial. A memória é utilizada para determinar, a cada passo da construção, o conjunto de cidades que têm ainda que forem visitadas de forma a garantir que seja elaborada uma solução viável. Adicionalmente, permite-se à formiga refazer o seu circuito, assim que esteja completo.

Depois de todas as formigas terem construído um circuito, os feromônio são atualizados. Isto é, tipicamente elaborado através da descida da força dos trilhos dos feromônio, através de um fator constante, e depois é dada liberdade para que as formigas depositem seus feromônio nos arcos que visitaram. A atualização dos trilhos é feita de tal forma que os arcos mais curtos, ou visitados por muitas formigas, recebem quantidades maiores de feromônio e, por isso, são escolhidas com maior probabilidade nas iterações posteriores (Figura 4). Neste sentido, a quantidade de feromônio representa a probabilidade de uma formiga escolher a próxima cidade *j* quando ainda estiver na cidade *i*.

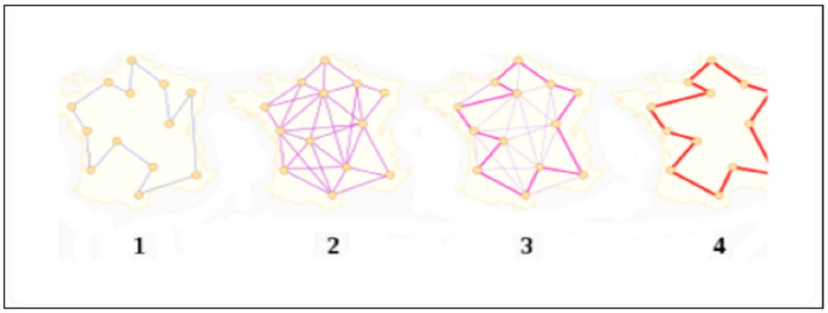

Figura 3 Identificação das rotas traçadas pelos agentes.

3.2.1 ACO em funcionamento no PCV (Modelagem Matemática)

O exemplo seguinte segue com um cenário mais genérico e simples para a resolução e demonstração da movimentação dos agentes, representação das rotas em uma árvore, aplicação da fórmula para movimentação e, finalmente, atualização dos valores de feromônio. Para efeitos de demonstração serão consideradas as rotas representadas pela árvore seguinte (figura 8):

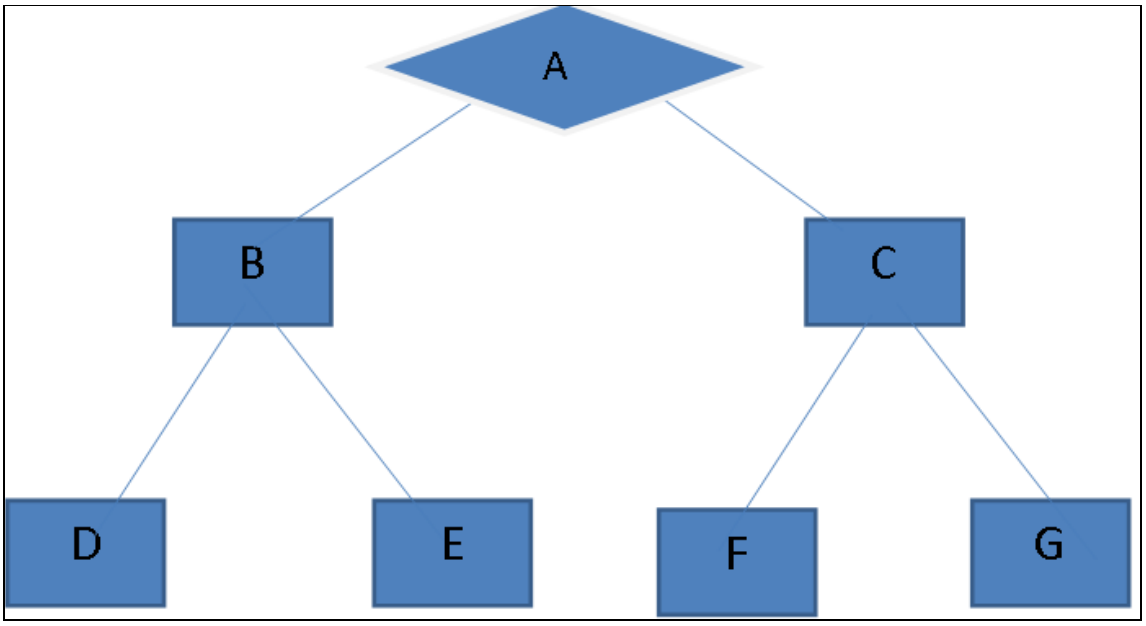

Figura 4 Árvore gerada a partir de um PCV

Todo o processo de escolha de rotas e atualização de pesos será baseado nas equações que seguem:

$$Prob_{ij}^{T} = \frac{[Fer_{ij}^{T}]^{\alpha} x\ [N_{PesoDoTrajeto}]^{\beta}}{\sum([Fer_{ij}^{T}]^{\alpha} + \ [N_{PesoDoTrajeto}]^{\beta}}$$

(4)

$$\Delta t_{ij}^{T} = \frac{p\_def}{L^{t}}$$

(5)

$$Fer_{ij}^{T} = \ Fer_{ij}^{T} * (1 - p) + \Delta t_{ij}^{T}$$

(6)

Onde:

- NpesoDoTrajeto foi definido após estudo e observação das rotas;
- Tij = Feromônio que inicialmente é igual a zero;
- T = Representa a formiga;
- ij = Caminho percorrido pela formiga na iteração;
- FerTij = Feromônio depositado pela formiga T no caminho ij;
- □= Peso definido pelo usuário para o feromônio aplicado nas trilhas percorridas;
- ß = Peso definido pelo usuário para o número de peças no lugar;
- p = Evaporação natural do feromônio que é igual a 0,5;
- p_def = Peso definido pelo usuário que será utilizado como parte do cálculo do feromônio que será aplicado na trilha;
- LT = Caminho percorrido pela formiga T pelos nós.

Os agentes têm A como ponto de partida. A partir deste ponto, a formiga tem duas opções: B ou C. A movimentação vai variar com o peso do trajeto e a quantidade de feromônio depositado no caminho. Em uma primeira interação, onde o primeiro agente vai caminhar pela primeira vez, os valores de feromônio são iguais em todos os caminhos (Tabela 2).

Tabela 2 Peso da Trajetória

A-B	3
A-C	4
B-D	5
B-E	2
C-F	6
C-G	4

Tabela 3 Valor do Feromônio

A-B	1
A-C	1
B-D	1
B-E	1
C-F	1
C-G	1

Definido os valores iniciais do feromônio e o peso das rotas (Tabela 3), aplica-se a equação 1 para que o agente tenha definida a probabilidade de qual será a melhor rota.

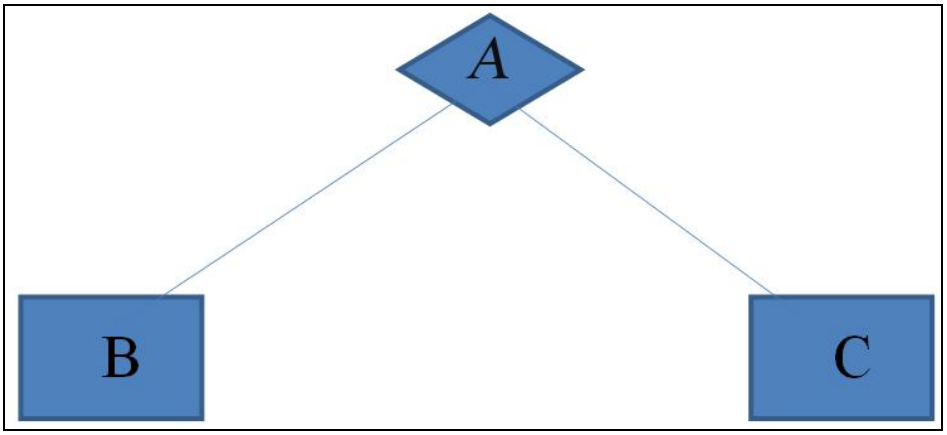

Figura 5 Primeira movimentação de um agente

Então primeiro calcula-se de A para B:

$$Prob_{ij}^{T} = \frac{[1\ \ ]^{0,33} x\ [3]^{1}}{([1\ \ ]^{0,10} x\ [3]^{1}) + (1\ \ ]^{0,10} x\ [4]^{1})} = 0{,}42$$

E em seguida, de A para C:

$$Prob_{ij}^{T} = \frac{[1\ \ ]^{0,33} x\ [4]^{1}}{([1\ \ ]^{0,10} x\ [3]^{1}) + (1\ \ ]^{0,10} x\ [4]^{1})} = 0{,}57$$

Observando os resultados, nota-se que existe probabilidade de aproximadamente 57% de que a movimentação do agente seja de A para C. Em seguida, o processo se repete, agora, supondo que o agente tenha escolhido mover-se para C, faz-se:

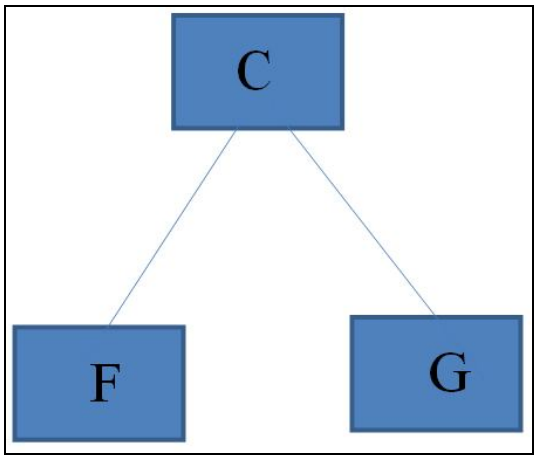

Figura 6 Segundo movimento da primeira interação

De C para F:

$$Prob_{ij}^{T} = \frac{[1\ \ ]^{0,33} x\ [6]^{1}}{([1\ \ ]^{0,10} x\ [6]^{1}) + (1\ \ ]^{0,10} x\ [4]^{1})} = 0{,}60$$

E de C para F:

$$Prob_{ij}^{T} = \frac{[1\ \ ]^{0,33} x\ [4]^{1}}{([1\ \ ]^{0,10} x\ [6]^{1}) + (1\ \ ]^{0,10} x\ [4]^{1})} = 0{,}40$$

Novamente, os agentes terão a tendência de se mover para o lado com a maior probabilidade, no caso para F com 60%. Ao chegar ao final da interação, deve se fazer a atualização dos valores de feromônio de acordo com as novas probabilidades, aplicando-se a equação (5).

$$\Delta t_{ij}^{T} = \frac{p_{def}}{L^{t}} = \frac{10}{2} = 5$$

$$Fer_{ij}^{T} = 1 * (1 - 0{,}5) + 5 = 5{,}5$$

Esse processo deve ser aplicado para todos os caminhos no qual o agente passou, fazendo o caminho de retorno. A nova tabela de feromônio ficaria assim:

Tabela 4 Valores de Feromônio atualizados

A-B	1
A-C	10,5
B-D	1
B-E	1
C-F	5,5
C-G	1

O processo se repetirá várias vezes até o critério de parada do algoritmo, devolvendo a melhor solução encontrada de acordo com as iterações realizadas.

A seção seguinte trata de alguns exemplos de aplicações do ACO na resolução de outros problemas de roteamento, demonstrando a viabilidade da escolha deste algoritmo para solucionar o problema proposto nesta pesquisa (RUBI, 2007).

4. TRABALHOS CORRELATOS

Os problemas de roteamento são comuns a várias áreas diferentes. Outros trabalhos com problemáticas diferentes já foram escritos e desenvolvidos e, em vários deles, a solução é alcançada através do ACO.

Em um primeiro exemplo pode ser mencionado o trabalho de Garbe (2006) que possui uma problemática semelhante à deste trabalho. Neste trabalho é proposto um novo algoritmo com desempenho equivalente ao ACO, mas com complexidade computacional menor. O trabalho é focalizado em redes do tipo data grama com topologia irregular, descrevendo suas propriedades e características e realizando uma análise e comparação de seus desempenhos em um ambiente de simulação. Os objetivos deste trabalho foram apresentar algoritmos baseados no comportamento de formigas para resolver o problema de encaminhamento de dados para aliviar o congestionamento nas redes e verificar a melhora dos resultados no problema de roteamento adaptativo utilizado em diferentes sistemas de comunicação de dados. Para isso foram estudados dois algoritmos de roteamento, são eles: o AntNet e o Trail Blazer.

O AntNet é composto por dois conjuntos de agentes móveis homogêneos em retorno(agentes que possuem prioridade sobre os pacotes de dados para acelerar a propagação da informação acumulada). Agentes de cada um dos conjuntos possuem a mesma estrutura básica, mas diferem em como são situados no ambiente, ou seja, eles podem sentir diferentes entradas e podem produzir diferentes saídas. Trail Blazer é um algoritmo que tem como função diminuir o congestionamento das redes a partir de decisões locais baseadas em medidas de latência coletadas por pacotes exploradores. Através de extensivos testes usando métricas variadas como quantidade de pacotes entregues, atraso nos pacotes, número de saltos e número de agentes gerados e recebidos, Garbe (2006) concluiu que os algoritmos baseados em agentes conseguem manter uma maior taxa de entrega de pacotes e um melhor desempenho quanto ao atraso dos pacotes, mas esses algoritmos não são compatíveis com os atuais algoritmos de roteamento encontrados em funcionamento, ou seja, eles não são capazes de interagir lado a lado com o OSPF (*Open Shortest Path First*).

Em outro trabalho, Tavares e Coelho (2005) descrevem uma variação do algoritmo da colônia de formigas utilizando um robô com a limitação de um cordão umbilical. No artigo

são mostradas modificações no ACO, a fim de simplificar o algoritmo. As modificações incluem: a não utilização da premissa de que todos os caminhos estão disponíveis e o fato de todas as formigas caminharem apenas por caminhos permitidos. Simplificações como essas permitiram maior consistência do algoritmo no âmbito dos sistemas autônomos baseados em sistemas processados com baixo poder computacional. Após implementar o algoritmo em utilizando a linguagem de programação JAVA, conclui-se que o cordão umbilical torna vários caminhos proibitivos devido ao retorno que causa na trajetória, pois, apesar de a formiga conseguir atingir seu objetivo, o traço de feromônio é estendido para mais nós, causando um distúrbio no meio, porém, conseguiu-se determinar um caminho ótimo para o mesmo.

Em um terceiro exemplo de aplicação do modelo, Silveira (2010) aborda o uso do ACO no teste da ordem de religamento de chaves de um sistema elétrico, de modo que se faça o menor número de mudanças de chaves para o restabelecimento de energia, resultando em uma maior economia de recursos necessários para essa tarefa. Em seu trabalho, o autor concluiu que a abordagem utilizada conseguiu atender os objetivos propostos. Adicionalmente, o tempo gasto e a qualidade das soluções obtidas mostram que a aplicação do ACO em problemas práticos da área da Engenharia Elétrica, que visam o restabelecimento de SEPs (Sistemas Elétricos de Potência), trará benefícios importantes, visto que se obteve a redução do tempo gasto para realizar a tarefa de restabelecimento de uma forma factível e eficiente em um problema real.

5. METODOLOGIA

Para que se possa aplicar um algoritmo capaz de criar resultados que sejam utilizados como base para alterações em um ambiente real necessita-se, primeiramente, coletar dados que fazem parte do ambiente estudado, no caso desta pesquisa este ambiente envolve as rotas de ônibus urbanos da cidade de Bauru. A quantidade de variáveis contidas nesse ambiente em particular é grande, dessa forma, torna-se necessário focar em algumas delas, consideradas mais importantes. Como etapa inicial, procedeu-se com a coleta dos dados, para isso fez-se uso de uma base informações contida no site da EMDURB[2]. Os dados coletados referem-se às rotas dos ônibus urbanos (tempo e distância percorrida). Entretanto, apenas os dados contidos no site (sem o processamento adequado) não eram suficientes para que se pudessem atingir os objetivos desta pesquisa. Dessa maneira, novos dados foram gerados com base nas informações coletadas no site da EMDURB (tempo de saída e de chegada dos ônibus) e informações de distância (obtidas através de medições usando o GoogleMaps[3]). Para a obtenção de um dado que pudesse servir como peso geral de um trajeto inteiro, o cálculo foi feito através de uma razão entre o tempo X que um único ônibus leva entre seu ponto inicial e final e a distância percorrida por esse mesmo ônibus entre esses dois pontos.

É necessário observar também a ocorrência de fatores não-determinísticos, como a alta incidência de enchentes em um determinado local, presença de sinais de trânsito, quantidade de pistas, qualidade do asfalto entre outros fatores que podem alterar o desempenho do veículo. Para que pudessem ser atribuídos pesos diferentes em trechos onde eventos não controláveis agem sobre o ambiente, é necessário um estudo especifico para cada rota em que o algoritmo for aplicado. Para esta pesquisa, estes fatores não foram considerados, uma vez que em um primeiro momento o objetivo está centrado em verificar a viabilidade de aplicação do algoritmo ACO, considerando inicialmente uma modelagem mais simples para o problema proposto.

Como Bauru conta com um grande número de ruas, pontos de parada de ônibus, torna-se inviável a aplicação do algoritmo em todas as rotas de ônibus presentes

[2] http://www.emdurb.com.br

[3] http://maps.google.com.br/

atualmente. Portanto, para esse projeto foram escolhidas apenas algumas rotas, envolvendo algumas ruas e avenidas principais da cidade.

A aplicação foi implementada usando as facilidades fornecidas pela ferramenta de desenvolvimento do NetBeans (descrita no capítulo 6). Inicialmente foram realizados testes utilizando um modelo básico do ACO para verificar a adequação da linguagem e do algoritmo, para isso, o clássico problema do Caixeiro Viajante foi utilizado. Uma vez constatada a viabilidade de utilização da linguagem e do algoritmo, um protótipo focado em dados da cidade de Bauru foi desenvolvido. Cabe destacar que testes com a ferramenta estão previstos como trabalhos futuros uma vez que, no estágio atual do protótipo, dados como, por exemplo, sentido das vias, pontos de enchentes e congestionamentos não estão disponíveis, impossibilitando uma análise mais profunda de utilidade da ferramenta no processo de criação de rotas reais.

A aplicação tem vários métodos vitais para o seu funcionamento. Como visto anteriormente, o ACO tem partes que funcionam separadas, mas no final agem como um todo. Portanto, teremos métodos responsáveis pela movimentação dos agentes, pelo depósito e evaporação dos feromônio, para a atribuição de peso as rotas e, finalmente, para o envio de informações para o GoogleMaps para que os resultados possam ser visualizados.

A seguir vemos um exemplo de um trecho de código, desenvolvido em Java, integrado à API GoogleMaps responsável por enviar informações para a criação de uma rota simples (este código é similar àquele que foi desenvolvido neste projeto):

```
269    directionsPanel = document.getElementById("r");
270    directions = new GDirections(map, directionsPanel);
271    <div id="r" style="width: 300px; height: 500px; position: absolute;"></div>
272    directions.load(string);
```

Figura 7 Trecho de código usado no funcionamento da API GoogleMaps

No exemplo estão destacados alguns elementos discutidos a seguir. Para mostrar a rota, é necessário criar uma nova *div* com o id igual à "*route*", é neste local é exibido o passo a passo da rota traçada. Essa captura do local foi feito nesta linha *directionsPanel = document.getElementById("route");* Para traçar a rota, deve-se instanciar o objeto GDirections e informar para ele carregar a rota através do

método *directions.load(string);* No método *load* deve-se passar os dois endereços escritos por extenso, antes o endereço de partida devemos atribuir *from*: e antes do endereço de destinoto: , ficando a string completa *from*: São Paulo to: Rio de Janeiro , por exemplo. Caso algum dos dois endereços não seja encontrado, o código exibirá um *alert* informando que a rota não pode ser traçada (FERNANDES, 2009). A próxima seção detalha as ferramentas que foram utilizadas no processo de desenvolvimento do prótipo.

5.1 FERRAMENTAS UTILIZADAS NO PROJETO

Para o desenvolvimento da aplicação foram utilizadas diversas ferramentas a fim de cumprir os objetivos deste trabalho. Basicamente, as ferramentas foram usadas para: desenvolver o algoritmo e a aplicação bem como exibir esses resultados de forma adequada.

A linguagem escolhida para o desenvolvimento da aplicação foi o Java e a ferramenta de desenvolvimento foi NetBeans 7.0. Para a exibição dos resultados obtidos pelo algoritmo, a API do GoogleMaps foi usada.

5.1.1 API GoogleMaps

O Google é conhecido mundialmente por ser o buscador mais poderoso e usado da internet. Além de fornecer esse serviço, o Google ainda oferece uma série de ferramentas gratuitas, como o Gmail (seviço de email), GoogleDocs (Office via *browser*) e outras várias ferramentas.

Umas das ferramentas do Google mais famosas e usada é sem dúvida é o GoogleMaps. A ferramenta possibilita a visualização de ruas de cidades de todo o mundo. Ainda permite que o usuário crie rotas de um ponto inicial A até um ponto final B. Na criação da rota, ainda pode-se personalizar vários aspectos, como: veículo escolhido para percorrer o trajeto ou ainda evitar pedágios no caminho.

Mas o GoogleMaps vai além da criação de simples rotas, como um GPS. Justamente por estar na internet, onde existe uma grande comunidade de desenvolvedores e programadores, o Google liberou uma série de API(do inglês, *Application Programming Interface*) que permitem o uso diferenciado dessas ferramentas. A GoogleMaps API fornece esses serviços da web como uma interface para que serviços externos solicitem dados desta

API e usem esses dados em seus aplicativos do Google Maps. Esses serviços foram desenvolvidos para serem usados em conjunto com um mapa, conforme Figura 12.

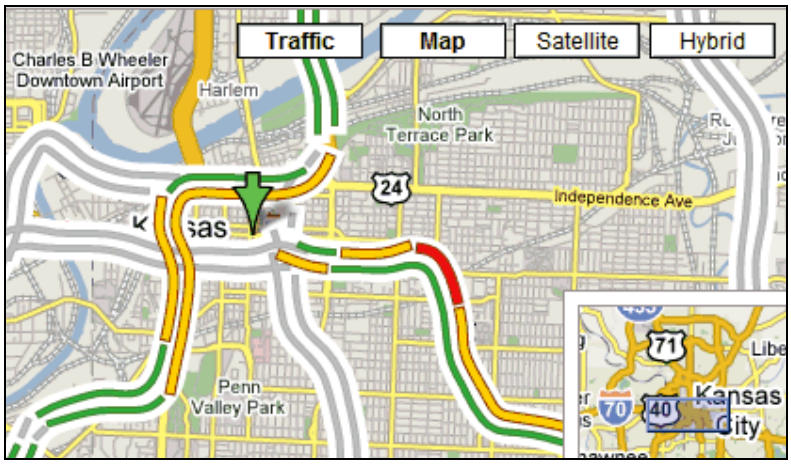

Figura 8 Exemplo de rota no GoogleMaps

Esses serviços da web usam solicitações HTTP para especificar URLs, passando parâmetros de URL como argumentos. Normalmente, esses serviços retornam dados nas solicitações HTTP no formato JSON ou XML para análise e/ou processamento por seu aplicativo(Documentação GoogleMaps, 2011).

5.1.2 Java e netbeans

A linguagem escolhida para o desenvolvimento do aplicativo foi o Java. O motivo por trás da escolhe vem pela familiaridade com a linguagem, fácil acesso a material didático e acadêmico. A linguagem Java é mundialmente conhecida por sua robustez e bom desempenho.

Além das características mencionadas anteriormente, uma das características mais marcantes do Java é a portabilidade. O mesmo código pode ser aproveitado para um ambiente *web*, *desktop* e *mobile*. Essa funcionalidade adiciona valor ao trabalho, uma vez que o sistema completo conta a opção de ser usado em várias plataformas diferentes, aumentando sua gama de aplicações.

Para aplicar o algoritmo em Java, foi usado a IDE NetBeans 7.0. O mesmo motivo citado anteriormente aplica-se a esta ferramenta. A escolha foi feita baseando-se na familiaridade com a aplicação.

A IDE NetBeans é desenvolvida pela Oracle e é uma ferramenta muito útil. Além das inúmeras facilidades nativas, ela ainda trabalha com uma enorme quantidade de linguagens diferentes, sendo Java uma delas. Por esses motivos, essa plataforma de desenvolvimento é a mais indicada para este trabalho.

Escolher ferramenta e linguagem conhecidas e familiares elimina a necessidade de tempo de adaptação e aprendizagem, ou seja, menos tempo de adaptação possibilita um maior tempo de desenvolvimento, testes e aperfeiçoamento.

É importante também mencionar que como será utilizada uma ferramenta que se utiliza da Internet para enviar e receber informações, o Java fornece maior facilidade para o desenvolvimento de uma aplicação Web.

6 RESULTADOS

Neste capitulo são apresentados os resultados obtidos pelo sistema desenvolvido em Java. Foram realizados dois tipos diferentes de teste, são eles: aplicação do algoritmo em uma rota simples (PCV) e aplicação em uma rota considerando dados reais do problema proposto no projeto.

6.1 ACO em funcionamento no PCV

A fim de facilitar a visualização da aplicação do ACO ao Problema do Caixeiro Viajante, tomou-se como exemplo a seguinte rota obtida a partir do GoogleMaps representada pela figura 4:

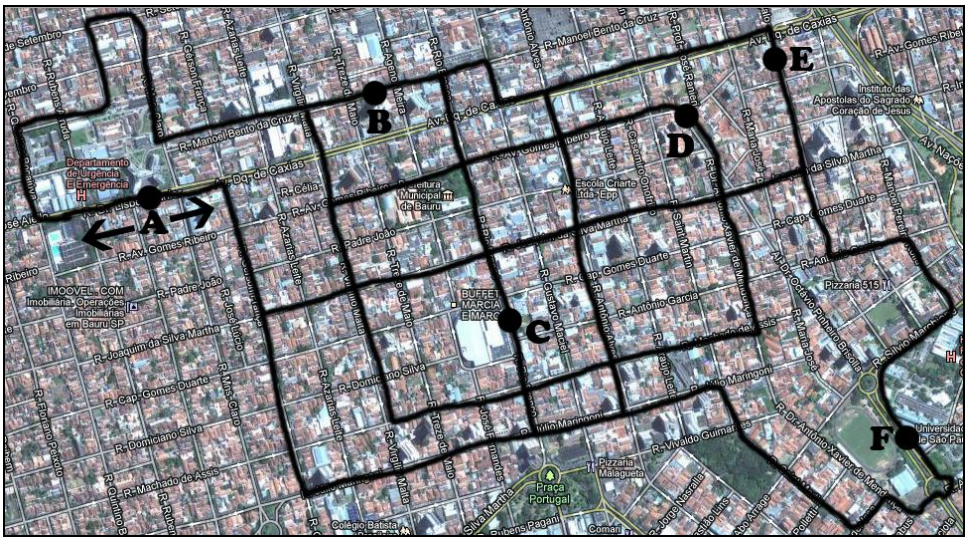

Figura 9 Cenário Inicial

Na figura 9 anterior, pode-se observar seis pontos que são distribuídos por toda a extensão da rota proposta, tendo A como ponto de partida. Neste trajeto, os agentes do ACO terão que se locomover por todo o percurso, passando por todos os pontos até voltar ao ponto inicial. Apenas para facilitar o entendimento, não será respeitado o sentido das ruas.

Na figura 10, observa-se o movimento dos agentes, fazendo a escolha (de forma aleatória) de qual a direção deverá seguir. Pode-se observar que à direita de A já existe uma

maior concentração de feromônio (representados por pontos vermelhos na imagem), isso indicará para os próximos agentes que entrarem na rota que o caminho à direita é, aparentemente, o melhor.

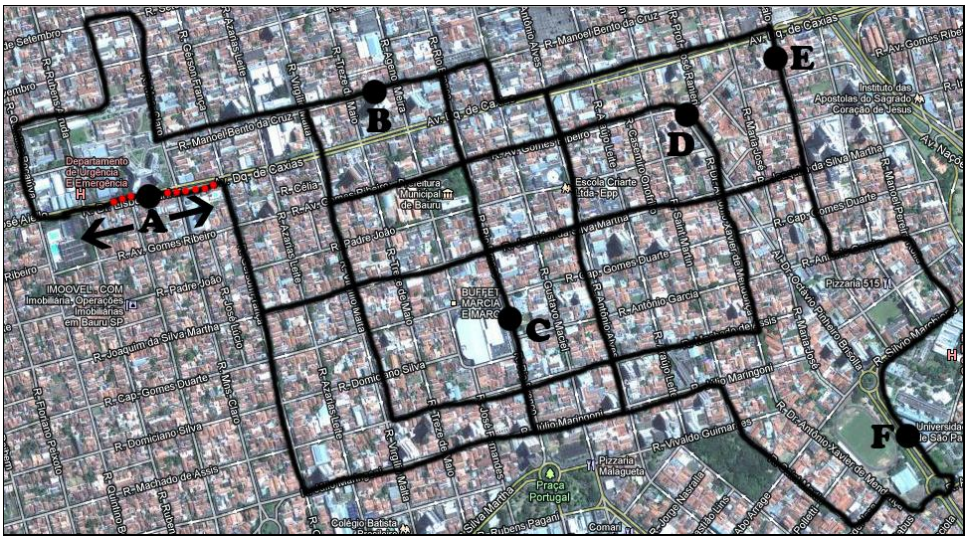

Figura 10 Movimentação dos Agentes

É importante notar que, ao mesmo tempo em que os agentes depositam os feromônio pelo caminho que passam o mesmo também "evapora" aos poucos para evitar que a concentração se torne tão grande que a probabilidade de um agente escolher outro caminho aleatoriamente se torne próxima de zero.

Pode-se notar então que ao mesmo tempo em que a evaporação do feromônio acontece, a deposição continua em todas as rotas em que os agentes seguem. Na figura 11 a seguir percebe-se que os agentes seguiram depositando feromônio até encontrarem outra bifurcação. Enquanto isso, o caminho à esquerda do início continua, mesmo que de forma reduzida, a ser escolhido por outros agentes que, por sua vez, também depositam feromônio.

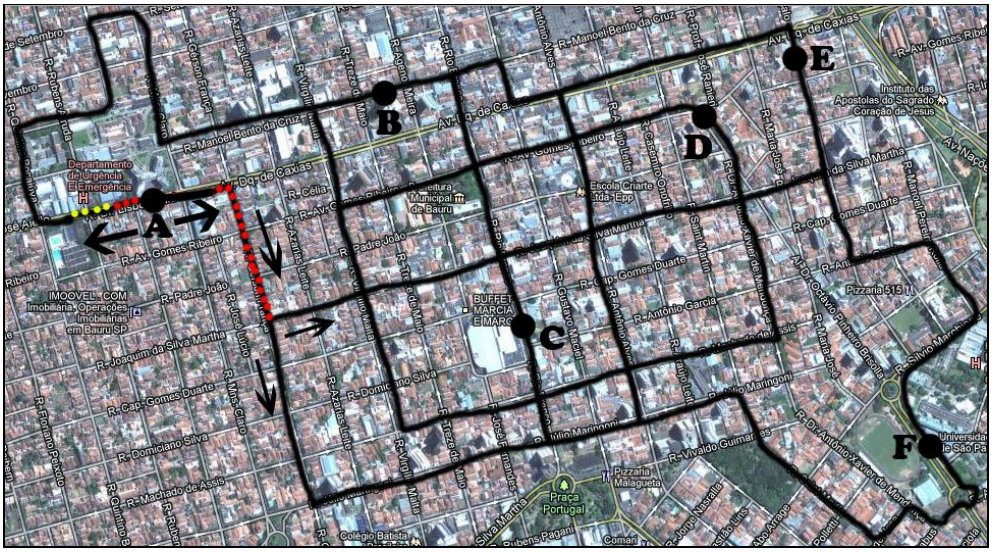

Figura 11 Movimentação e Evaporação

Esse processo de escolha aleatória, evaporação e liberação de feromônio e movimentação só terá fim quando algum fator de parada definido pelo algoritmo for alcançado. Mesmo que, como exibido na figura 12 seguinte, o algoritmo encontre um caminho que percorra todas as cidades, ele irá continuar funcionando, uma vez que uma rota melhor ainda possa existir.

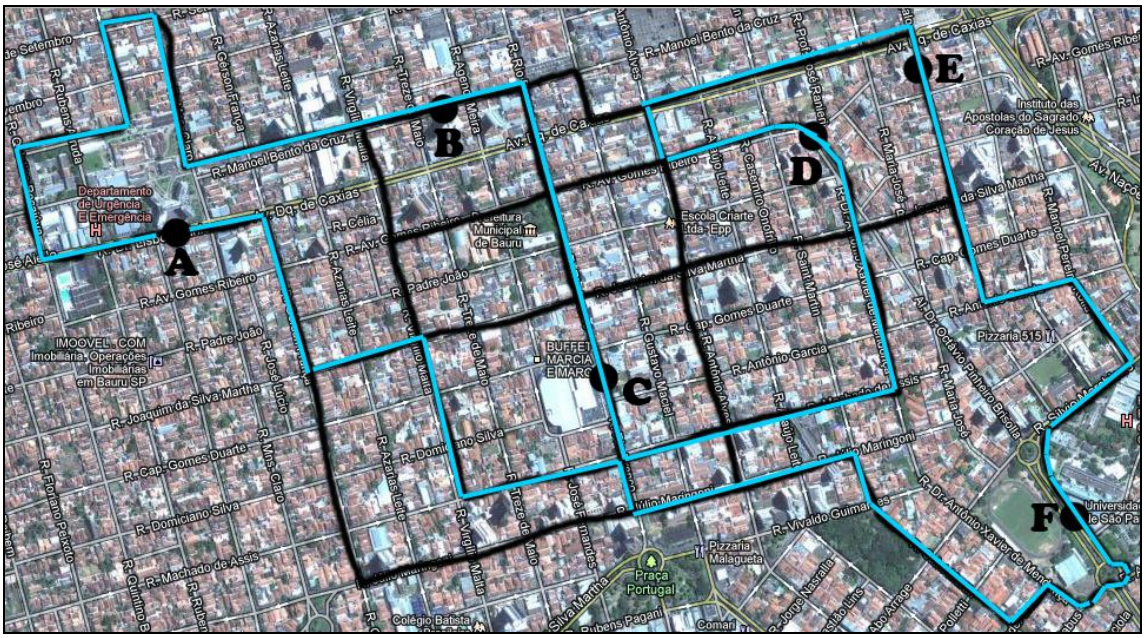

Figura 12 Circuito completo após uma interação

6.2 *ACO em funcionamento em uma Árvore Simples*

Inicialmente foi montada uma árvore simples, conforme a figura 13, em que foi aplicada o algoritmo em busca da resolução.

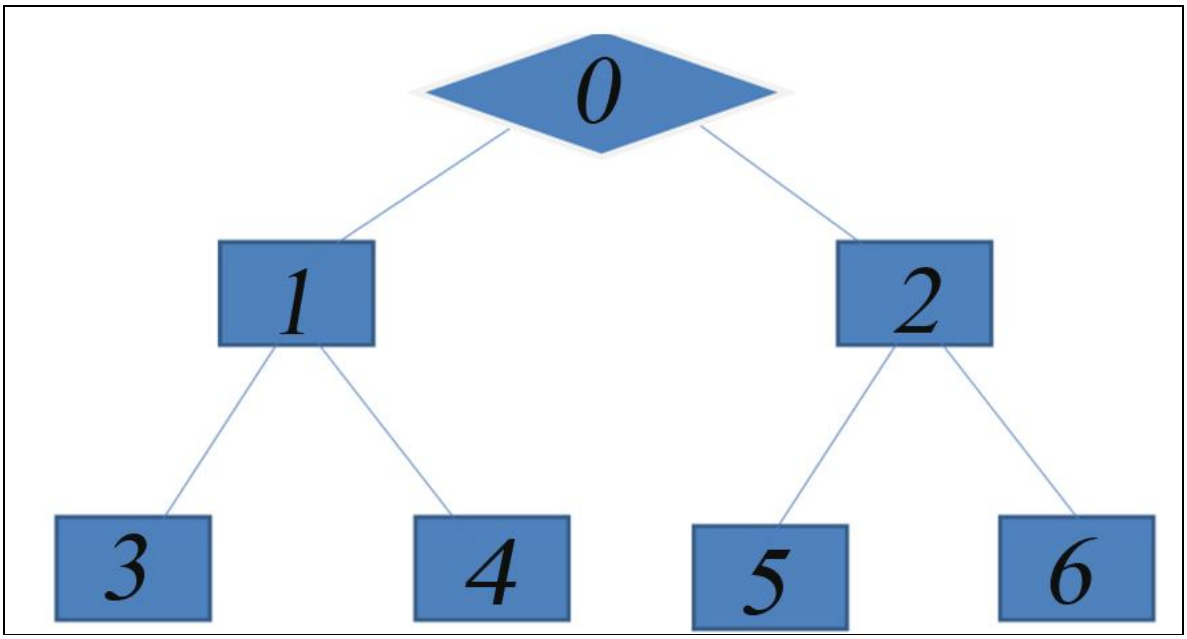

Figura 13 Árvore teste

Vale recordar que o peso atribuído no problema do caixeiro viajante é feito de forma aleatória, contudo, no contexto deste trabalho, por se tratar da resolução do problema de roteamento de ônibus, é feita uma razão entre tempo gasto e distância percorrida pelo veículo.

A primeira tela (figura 14) do sistema exibe uma página simples, com um menu *drop down* que possibilita a escolha de qual rota o usuário deseja calcular, um mapa gerado pela API (centralizado em Bauru) e, finalmente, um botão que dá inicio ao funcionamento do algoritmo.

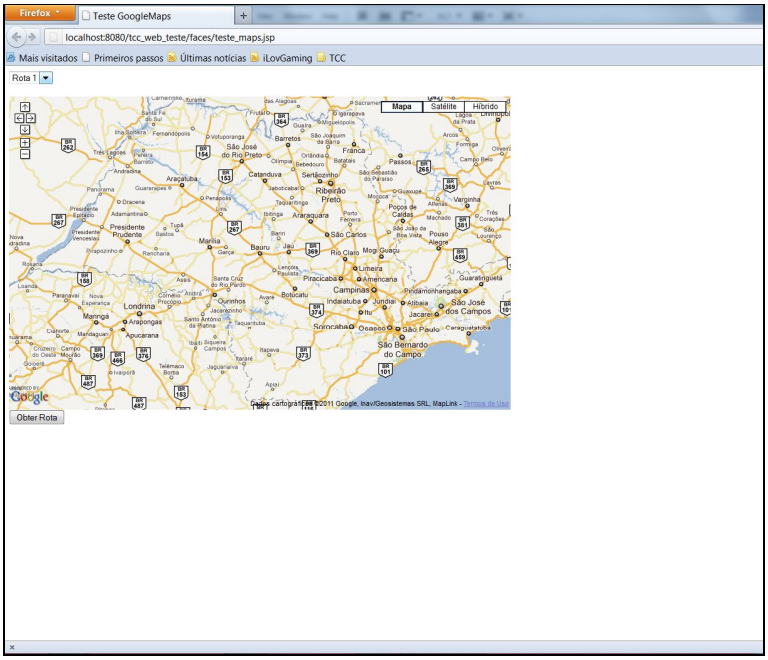

Figura 14 Tela inicial

Inicialmente a tabela de feromônio possui valores iguais, como segue a tabela seguir:

Tabela 5 Valor do feromônio em teste

Fer[0]	1.0
Fer[1]	1.0
Fer[2]	1.0
Fer[3]	1.0
Fer[4]	1.0
Fer[5]	1.0

Os valores da tabela anterior são atualizados a cada interação (uma interação é quando o agente percorre todo o caminho indo do ponto inicial ao final). Ao fim de cada interação, o trecho seguinte (figura 15) é aplicado:

```
75        cont = 0;
76        tri = constante / cont;
77        fer[(int) x] = (fer[(int) x] * (0.5) + tri) + fer[(int) x];
78        fer[(int) y] = (fer[(int) y] * (0.5) + tri) + fer[(int) y];
```

Figura 15 Atualização do Feromônio

Onde:

- cont: resultado de um contato atribuído para receber a quantidade de vértices percorridos pelo agente;
- constante: valor atribuído pelo usuário;
- fer[]: vetor que define a quantidade de feromônio;
- x: indice do vetor fer[];
- y: indice do vetor fer[];

Durante toda a execução do código, parte do trecho a seguir (figura 16) é executada. Ele tem como função calcular a probabilidade de movimentação. Lembrando que o movimento do agente, apesar de se basear em um peso probabilístico, se movimenta de forma aleatória.

```
90        j = nome[i];
91        l = nome[i + 2];
92        prob = (fer[1] * valores[j]) / ((valores[j]) + (valores[l]));
93        prob = prob * 0x64;
94        prob2 = (fer[1] * valores[2]) / ((valores[j]) + (valores[l]));
95        prob2 = prob2 * 0x64;
96        double numero = (int) (Math.random() * 100);
```

Figura 16 Cálculo da probabilidade na direção do movimento

Onde:

- prob, prob2: pesos probabilísticos;
- valores[]: vetor que contém os pesos de cada trecho

É então feita uma comparação entre os valores e o um caminho é escolhido. Esse trecho se repetirá até que o fim da árvore gerada pelo desmembramento da rota atinja o fim. Todo esse processo por ser repetido N vezes, a escolha de N baseia-se na necessidade de

precisão do algoritmo e custo computacional. É interessante mencionar que quanto maior a quantidade de interações, mais preciso será o resultado final.

Após uma interação completa, podemos observar a mudança na tabela de feromônio, um exemplo dessa mudança corresponde à tabela 6.

Tabela 6 Valores de feromônio atualizados

Fer[0]	1.0
Fer[1]	3.0
Fer[2]	1.0
Fer[3]	1.0
Fer[4]	4.5
Fer[5]	1.0

Na figura 17 pode-se ver o resultado do teste:

Figura 17 Resultado apresentado no *promt*

Onde, os agentes percorreram a árvore cem vezes, encontrando o caminho 0 -> 2 -> 5 como melhor. Observando a tabela abaixo, pode-se notar que o caminho escolhido não é ótimo, mas ainda seria melhor que 0->1->4, por exemplo.

Uma comparação pode ser feita entre o problema real e o teste com o caixeiro viajante: apesar da solução não ser ótima, ela pode ser atingida de forma rápida e com baixo custo computacional e ainda ser uma melhor opção que a gerada.

Por fim, o resultado é exibido conforme a imagem (figura 18) a seguir, onde a linha azul delimita o trecho escolhido pelo algoritmo e do lado direito é exibido o resultado detalhadamente.

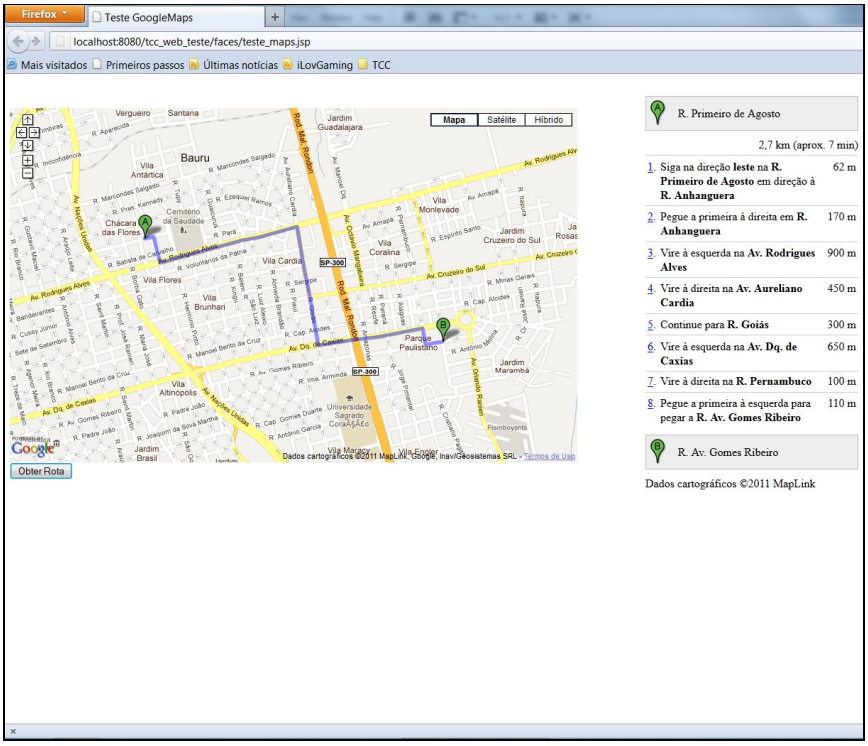

Figura 18 Resultado apresentado no GoogleMaps

É importante notar que o controle do sentido das ruas foi abstraido do código final, porém a API do GoogleMaps possui um controle própio para isso, afim de evitar que o trecho seja irreal.

6.3 Aplicação em um cenário real

A *interface* de saída (como vista na figura 19) foi construída com o intuito de exibir os resultados no mapa, dar ao usuário as opções de seleção de ponto inicial e final, bem como a atribuição dos pesos de cada nó da árvore (ruas). O GoogleMaps ainda oferece a opção de seleção de textura do mapa, como padrão foi escolhida a opção mais simples para que as rotas pudessem ser visualizadas de forma mais clara.

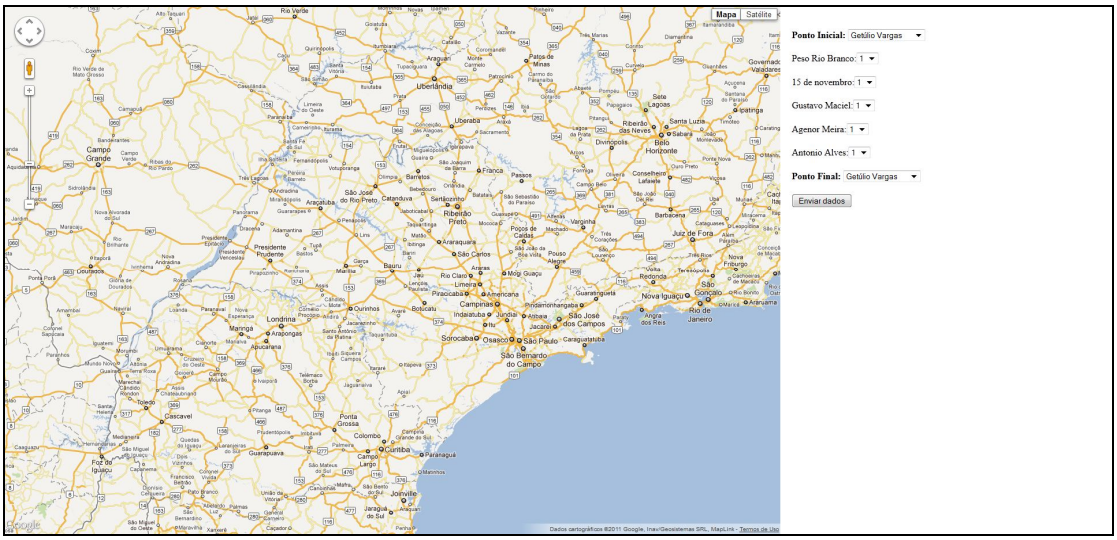

Figura 19 - Tela inicial

Como mencionado, o usuário pode selecionar tanto o ponto inicia quanto o final. No atual estágio de desenvolvimento, foram escolhidas quatro das principais avenidas da cidade de Bauru (Getúlio Vargas, Nações Unidas, Duque de Caxias e Castelo Branco). Conforme observado na imagem seguinte (figura 20), os pontos selecionados podem ser diferentes, contudo, para uma melhor simulação é interessante manter ambos os pontos (inicial e final) iguais, de modo que o ambiente lembre o funcionamento real de uma linha de ônibus.

É importante notar que, indepente dos pontos selecionados, não existe a possibilidade da criação de uma rota inconsistente, ou seja, uma rota que não exista (que invada contra mão, por exemplo). O proprio GoogleMaps cuida para que a rota respeite seus sentidos.

Figura 20 - Seleção de Rotas

Ainda é dado ao usuário a opção de seleção do peso de cada uma das ruas que estão entre os dois pontos (conforme figura 21), já pré-selecionados. A escala de peso varia entre um e cinco, sendo cinco o melhor e um o pior. A seleção dos pesos irá influenciar diretamente no resultado final e possibilita um maior grau de realismo e customização do protótipo.

Inserir os valores de forma livre é uma tentativa de solução para os problemas com os fatores não-determinísticos. Por exemplo, imagina-se em um ambiente real em um cenário normal que a rua Gustavo Maciel tenha um valor padrão de 3 (devido a variaveis fixas, como sinais de trânsito ou poucas preferenciais). Devido a um acidente, o cenário torna-se anormal e o valor pode cai para 1 e praticamente inviabliza a utilidade daquele trecho especifico.

Figura 21 - Atribuição de peso

Finalmente, quando o botão “Enviar dados” for usado, o algoritmo irá realizar os procedimentos mencionados no capítulo anterior gerando a seguinte tela de resultados (figura 22):

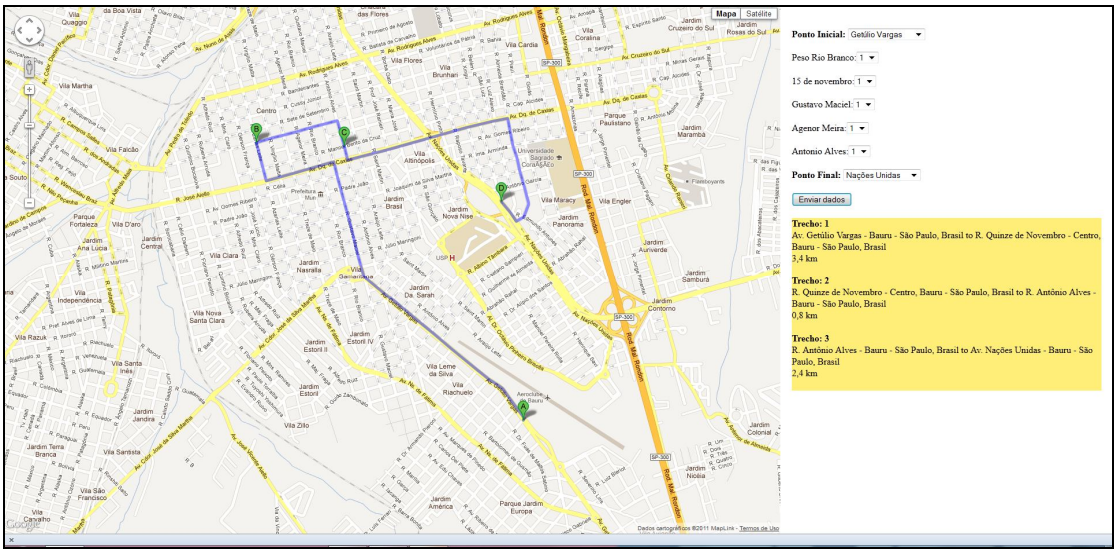

Figura 22 - Exibição do resultado

Nela, pode-se observar que foi traçada uma linha entre a Avenida Getúlio Vargas e a Avenida Nações unidas, passando pelos pontos B (Quinze de Novembro) e C (Antonio Alves).

Já no exemplo a seguir, pode-se observar um ambiente mais próximo do proposto (figura 23). Na imagem, o ponto de origem e o final são o mesmo. Isso significa que o agente precisa caminhar entre dois pontos (no exemplo: Rio Branco e Gustavo Maciel) e depois voltar ao ponto de origem. No estado atual do desenvolvimento do protótipo a volta ao ponto inicial não é calculada, a tarefa de roteamento para o ponto de origem é deixada a cargo do GoogleMaps.

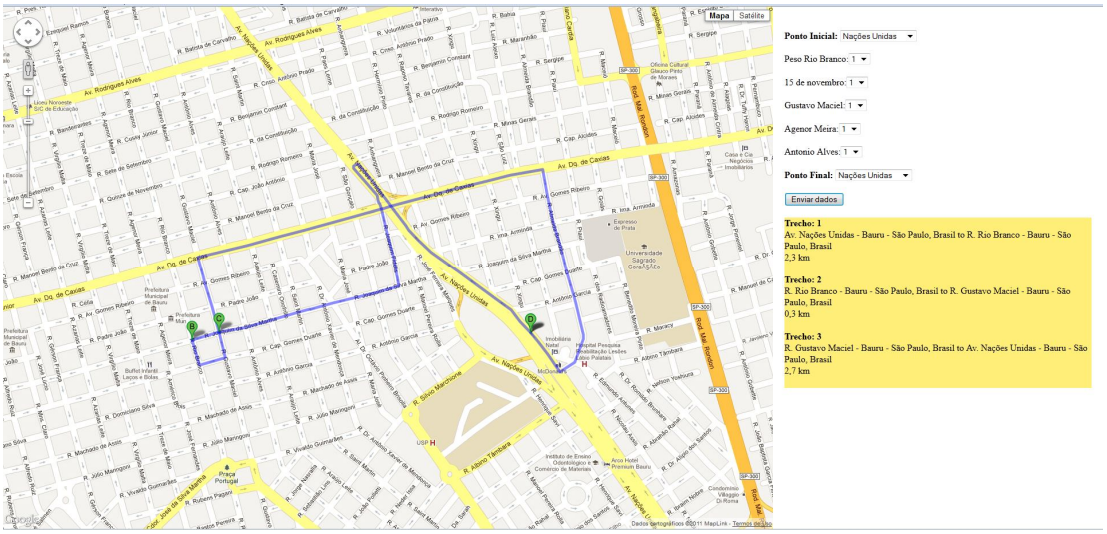

Figura 23 - Segundo exemplo de resultado

Em ambos os cenários, os resultados são obtidos de forma rápida. O bom desempenho do protótipo atribuisse ao fato do ACO (*Ant Colony Optimization*) não buscar o melhor resultado, dando como resposta um possível melhor caminho. As respostas rápidas também são creditadas ao GoogleMaps, que faz o calculo das rotas e o ligamento entre os pontos de forma ágil.

7 CONCLUSÃO

O ACO mostra-se uma alternativa robusta para problemas de roteamento, seja ele qual for. Do básico problema do caixeiro viajante, a exemplos mais complexos como roteamento de rede ou ainda o proposto neste trabalho. É uma opção de otimização extremamente versátil que pode trazer excelentes resultados, se bem aplicada. Entretanto, nem sempre se trata da melhor escolha para certos tipos de problema, por se tratar de um algoritmo que nem sempre traz soluções ótimas e depende do movimento baseado na probabilidade do feromônio depositado pelos agentes (formigas) que trilham pelo caminho proposto.

O problema com as rotas de ônibus na cidade de Bauru é claro. Ruas com pouca vazão de trânsito, um número limitado de ônibus e a grande demanda tornam o serviço lento e defasado. Criar novas rotas e mudar as atuais é um processo lento e custoso que demanda um estudo especifico de cada rota para que as alterações não interfiram no dia-a-dia dos usuários do serviço.

Também é importante mencionar as dificuldades encontradas durante a implementação do protótipo. A API GoogleMaps, mostrou-se eficaz, contudo ainda é uma ferramenta que não oferece muitos recuros extras. Uma das principais dificuldades foi o envio da informação para que a API encontrasse| o local desejado. Foram encontradas duas soluções: uma *String* que continha todas as informações do ponto ou a passagem dos valores de latitude e longitude. O primeiro meio permite uma maior probabilidade de erros, por se tratar de uma longa *String* que estará sujeita a erro. A segunda opção, mais precisa, possui problemas em sua obtenção. Não existe um meio automatizado para que o algoritmo busque esses valoreres.

A implementação do ACO para resolver o problema proposto, pelos resultados preliminares obtidos, parece ser uma alternativa viável, contudo o grande número de variáveis que influenciam o ambiente por onde os ônibus transitam tornam complexa a tarefa de encontrar todas as variáveis que realmente influenciam no resultado. Fez-se necessário então abstrair uma grande quantidade de dados, diminuindo até um mínimo possível onde os agentes tenham menos problemas na movimentação. Claramente, para que o protótipo desenvolvido possa ser aplicado em situações reais muitas outras variáveis, além de tempo e distância, deverão ser levadas em consideração. No estágio atual desta pesquisa apenas pode-

se concluir que o algoritmo e sua implementação em linguagem Java, juntamente com a API do GoogleMaps, parecem bastante adequadas para geração de uma aplicação completa. A exibição dos resultados alcançados no GoogleMaps também provou-se um recurso indispensável para a compreensão do resultado final, de forma clara e eficaz. Certamente, a automatização do processo de geração de rotas pode trazer benefícios uma vez que, atualmente, todo este processo é feito de modo manual.

.

8 BIBLIOGRAFIA

BABA, Cristina Mayumi et al. Otimização da colônia de formigas aplicada ao problema da programação e roteirização de veículos para o transporte de pessoas portadoras de deficiência. Encontro Nac. de Eng. de Produção, XXIV,2004, Florianópolis..ENEGEP.

BARTHOLOMEU, A. L. Frota de veículos em São Paulo cresce acima da média em março e amplia "nó" do trânsito da maior cidade do país. UOL, 2008 abril. Disponível em:http://noticias.uol.com.br/ultnot/especial/2008/transito/2008/04/24/ult5848u12.hjtm> Acesso em: 03 Abr 2011.

DOCUMENTAÇÃO GOOGLE MAPS<http://code.google.com/intl/pt-BR/apis/maps/documentation/mapplets/guide.html> Acesso: 17 Abr 2011.

DORIGO, Marco ET AL. Ant Colony Optimization and Swarm Intelligence. Berlin: Springer, 2004.

DORIGO, M. e GAMBARDELLA, L.M. (1997) Ant Colonies for the Travelling Salesman

Problem. In Biosystems, v. 43, n. 2, p. 73- 81. Julho.

DORIGO, Marco. STÜTZLE, Thomas. Ant Colony Optimization. Massachusetts: Bradford, 2004.

FERNANDES, Rodrigo. Entendendo um pouco a API Google Maps. Disponível em: <http://javafree.uol.com.br/artigo/874529/Entendendo-um-pouco-a-API-Google-Maps.html>. Acesso em: 6 Jun 2011.

GARBE, B.(2006) Análise de Algoritmos de Roteamento Baseados em Formigas. Escola Politécnica, Universidade de São Paulo.

OLIVEIRA, U. Frota de veículos aumenta e qualidade do ar só piora. Rede Bom Dia, Bauru,19 Set 2011. Disponível em: <http://www.redebomdia.com.br/Noticias/DiaaDia/25318/Frota+de+veiculos+aumenta+e+qualidade+do+ar+so+piora>.Acesso em: 03 Abr 2011.

Ônibus: tarifa de Limeira estará entre as mais caras. Jornal de Limeira, Limeira, 08 nov 2009 Disponível em: <http://www.jornaldelimeira.com.br/site/noticias_detalhes.php?ID_Noticia=30795>. Acesso em: 06 Jun 2011.

OSHIRO, V. Plano de mobilidade em Bauru vai começar. Jornal da Cidade, Bauru. Mar. 2011

Proibição para caminhões começa a valer em dez vias do Morumbi. G1, São Paulo, 27 Set 2010 Disponível em: <http://g1.globo.com/sao-paulo/noticia/2010/09/proibicao-para-caminhoes-comeca-valer-em-dez-vias-do-morumbi.html>. Acesso em: 06 Jun 2011

RUBI, R. R. (2007). Aplicação do Algoritmo de "Colônia de formigas" na resolução do 8-puzzle. Faculdade de Jaguariúna.

TAVARES, R. F. N. (2005). Planejamento de vistorias usando robôs móveis autônomos e otimização pelo algoritmo de colônia de formigas. Pontifícia Universidade Católica do Paraná.

THOMAS H. Cormen, CHARLES E. Leiserson, RONALD L. Rivest, CLIFFORD Stein. *Introduction to Algorithms*, Segunda Edição. MIT Press and McGraw-Hill, 2001

SILVEIRA, T. A utilização de meta-heurística de otimização denominada colônia de formigas para o restabelecimento de sistemas elétricos de potência. (2010). Universidade Federal de Alfenas.

STAGER, B. (2001). The Solving of Real World Problems using Evolutionary Algorithms. Undergraduate Thesis. University of Queensland. Outubro.

STÜTZLE, Thomas; DORIGO, Marco – *ACO Algorithms for the traveling salesman problem* [Em Linha]. New York: Swarm Intelligence Resources, 1999. [Acesso em: 09 Set 2011]. Disponível em WWW: <URL:http://staff.washington.edu/paymana/swarm/stutzle99-eaecs.pdf>

APÊNCIDE A – ARTIGO CIENTÍFICO

Sistema de Re-Roteamento de Ônibus Urbanos Utilizando o Algoritmo de Colônia de Formigas

Carlos Eduardo S. Dos Santos[1], Patrick P. Silva[1]

[1]Departamento de Ciências Exatas e Sociais Aplicadas – Universidade Sagrado Coração, São Paulo.

carlos1.santos@usc.edu.br, patrickpsilva@usc.edu.br

Abstract. *Brazil as a whole lives with a serious problem of transporting people and cargo. Little is invested in this sector and few differentiated and dynamic solutions are created. With the steady growth of large metropolitan centers, it is necessary to develop alternatives. This effect of swelling of the large cities is already seen even in smaller cities. A possible alternative is the re-routing of paths made by vehicles in order to avoid bottleneck points. To implement such a solution, we must find tools that allow the relocating of the routes. This study assessed the feasibility of applying the algorithm Ant Colony Optimization (ACO) for a simplified version of this problem, using as tools for developing a prototype of the Java language and the GoogleMaps API. The results support the utility of the ACO as a viable alternative for solving vehicle routing problems.*

Resumo. *Todo o Brasil vive com um grave problema de transporte de pessoas e carga. Pouco se investe nesse setor e poucas soluções dinâmicas e diferenciadas são criadas. Com o crescer constante dos grandes centros metropolitanos, faz-se necessário a criação de alternativas. Esse efeito de inchaço das grandes cidades já é notado até mesmo em cidades de menor porte. Uma alternativa possível é o re-roteamento dos trajetos feitos pelos veículos a fim de evitar pontos de gargalo. Para aplicar tal solução, é preciso encontrar ferramentas que permitam o realocamento das rotas em tempo real. Este estudo verificou a viabilidade da aplicação do algoritmo Ant Colony Optimization*

(ACO) para uma versão simplificada deste problema, utilizando como ferramentas para o desenvolvimento de um protótipo a linguagem JAVA e a API do GoogleMaps. Os resultados obtidos corroboram com a utilidade do ACO como uma alternativa viável de resolução de problemas de roteamento de veículos.

1. Introdução

A busca pela resolução de problemas complexos de modo mais simples e eficiente tem levado pesquisadores a propor novos métodos, dentre os quais podemos citar as técnicas de otimização inspiradas na natureza como a Otimização por Colônia de Formigas (ACO).

Esta forma de otimização proposta por Dorigo e Gambardella (1997), baseia-se na comunicação dos indivíduos realizada indiretamente por meio de uma trilha feita por formigas artificiais durante a exploração do espaço de busca. Esta técnica é considerada robusta e eficiente já que é um processo de aprendizado distribuído, onde cada agente simples (formigas artificiais) trabalha de modo coletivo, promovendo adaptações e modificações no ambiente, contribuindo para a solução final do problema.

O ACO possui um vasto campo de aplicações, podendo ser utilizada na resolução de vários problemas de complexidade exponencial dos quais se pode citar, por exemplo, o Problema de Roteamento de Veículos (PRV). O PRV é, inclusive, atualmente um dos problemas que mais merecem destaque uma vez que, nas últimas décadas, houve um crescimento significativo do número de veículos trafegando pelas avenidas e ruas das cidades, provocando engarrafamentos, maior consumo de combustíveis e aumento de tempo para se percorrer um determinado trajeto. Hoje em dia, nota-se que a frota de veículos cresce de forma desenfreada em várias cidades brasileiras de grande e médio porte. Diante deste crescimento das frotas de veículos, alternativas devem ser criadas de modo a contribuir para que o trânsito das cidades médias não se torne tão problemático quanto o das grandes metrópoles. Com base nesta possibilidade, este trabalho propõe utilizar a tecnologia como uma alternativa de combate a este problema, focando no transporte público de passageiros.

Diante disso, o objetivo geral deste trabalho é o estudo e aplicação da heurística de Otimização por Colônias de Formigas (ACO) em uma versão simplificada do problema de construção de rotas para os veículos que fazem o transporte público na cidade de Bauru, SP, verificando a possibilidade de se oferecer, por meio os resultados obtidos, uma nova alternativa à solução adotada atualmente pela empresa responsável pelo sistema de transporte.

2. Funcionamento do Ant Colony Optimization

Segundo Dorigo e Gambardella (1997) o ACO é um modelo de algoritmo baseado no comportamento natural de formigas. Neste modelo, as formigas trabalham como agentes para o bem da colônia em geral. Criar um algoritmo similar exige a compreensão do sistema como

um todo. A seguir vemos a Tabela 1 que estabelece uma comparação entre os aspectos biológicos e artificiais:

Tabela 7 - Comparação entre os Agentes Biológicos e Artificiais.

Agentes biológicos	Agentes Artificiais
Formigas	Agentes
Trilhas	Caminho
Intersecção da trilha	Pontos de parada
Concentração de feromônio	Pesos probabilísticos
Deposição feromônio	Incremento no peso
Evaporação de Feromônio	Decremento no peso

As formigas são os agentes que “caminham” nas trilhas em busca de um melhor caminho. Esses agentes são postos em um ponto inicial x e tem como objetivo chegar até um ponto final y. As formigas escolhem o caminho de forma aleatória, dessa forma, elas estão sempre caminhando rumo ao objetivo, havendo uma solução (mesmo que não ótima). Durante o percurso, os agentes irão depositar feromônios no decorrer da trilha como forma de indicar para os próximos agentes qual caminho tomar. Os feromônios depositados são responsáveis por atribuir peso a essas trilhas, fazendo com que os caminhos com maior peso tenham maior probabilidade de serem escolhidos pelos próximos agentes. Durante o trajeto, para poderem atingir o objetivo final, os agentes passam por intersecções entre as trilhas. Essas intersecções, que são consideradas pontos de parada obrigatória, podem variar em termos de quantidade e distância entre si.

O que faz esse algoritmo se tornar único é a forma como o feromônio é depositado: de

modo aleatório no percurso e em uma quantidade baseada em um peso probabilístico.

Quanto mais rápido for o caminho, mais rápido o agente retorna e preenche a trilha com feromônios, logo a próxima formiga tenderá a pegar o caminho com maior quantidade de feromônio. Os agentes, durante a execução do algoritmo, estarão sempre se movendo pelas rotas e sempre encontrando novas soluções. Por esse motivo o resultado, isto é, o caminho mais rápido, poderá ser alterado.

2.1. A Meta-Heurística ACO – Formulação Matemática

O ACO em sua primeira versão (DORIGO, 2004), inicia-se com cada agente (formiga) construindo uma solução própria para o problema baseada em soluções de agentes que já passaram por esse mesmo problema. Cada formiga k se move em um caminho onde os movimentos são selecionados segundo uma distruição de probabilidades dada por (BABA, Cristina Mayumi ET AL, 2004):

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]\cdot[\eta_{ij}]^{\beta}}{\sum_{j \in J_i^k}[\tau_{ij}]\cdot[\eta_{ij}]^{\beta}} & \text{se } j \in J_i^k \\ 0 \end{cases}$$

(1)

Onde:

$_{ij}p_{k}$ = probabilidade da formiga *k*, que se encontra em *i*, escolher o nó *j* como próximo nó a ser visitado;

τ_{ij} = quantidade de feromônio existente no arco (*i*,*j*). Inicialmente, adota-se um mesmo valor

τ_0 para todos os arcos da rede;

η_{ij} = função heurística que representa a atratividade do arco (*i*,*j*).

$_{i}J_{k}$ = conjunto de pontos ainda não visitados pela formiga *k*, que se encontra atualmente no ponto *i*;

β = valor heuristicamente escolhido, que pondera a importância da quantidade de feromônio existente no arco em relação à distância entre os nós *i* e *j*. (BABA, Cristina Mayumi ET AL, 2004).

A expressão acima faz com que o agente escolha o caminho com maior quantidade feromônio, ou seja, o trajeto que foi mais usado por outros agentes (que naquele momento é o caminho mais curto). Deve ser estipulado um ciclo de atualização de feromônio, uma vez que se os feromônios não tiverem seu peso diminuído, o custo computacional para que o agente escolha o menor caminho aumentará e o algoritmo perderá em desempenho. Assim, para cada clico de atualização (i,j), adiciona-se uma quantidade de feromônio proporcional ao tamanho da rota obtida (BABA, Cristina Mayumi et al, 2004):

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{melhor}$$

(2)

$$\Delta\tau_{ij}^{melhor} = \begin{cases} \frac{1}{L_{melhor}} \\ 0 \end{cases}$$

(3)

A Equação (2) visa a diminuição do peso do feromônio, onde ρ é o fator que determina a velocidade da diminuição do peso. Já a próxima Equação (3) é responsável por aumentar o peso do feromônio onde L é a distância total percorrida na rota construída pelo melhor agente da interação (BABA, 2004).

3. Metodologia

Para que se possa aplicar um algoritmo capaz de criar resultados que sejam utilizados como base para alterações em um ambiente real necessita-se, primeiramente, coletar dados que fazem parte do ambiente estudado, no caso desta pesquisa este ambiente envolve as rotas de ônibus urbanos da cidade de Bauru. A quantidade de variáveis contidas nesse ambiente em particular é grande, dessa forma, torna-se necessário focar em algumas delas, consideradas mais importantes.

Como etapa inicial, procedeu-se com a coleta dos dados, referentes às rotas dos ônibus urbanos (tempo e distância percorrida). Dessa maneira, novos dados foram gerados com base nas informações (tempo de saída e de chegada dos ônibus) coletadas no site da empresa responsável pelo transporte público e informações de distância obtidas através de medições usando o GoogleMaps. Para a obtenção de um dado que pudesse servir como peso geral de um trajeto inteiro (para o processo de atualização dos feromônios), o cálculo foi feito através de uma razão entre o tempo X que um único ônibus leva entre seu ponto inicial e final e a distância percorrida por esse mesmo ônibus entre esses dois pontos.

É necessário observar também a ocorrência de fatores não-determinísticos, como a alta incidência de enchentes em um determinado local, presença de sinais de trânsito, quantidade de pistas, qualidade do asfalto entre outros fatores que podem alterar o desempenho do veículo. Para que pudessem ser atribuídos pesos diferentes em trechos onde eventos não controláveis agem sobre o ambiente, é necessário um estudo especifico para cada rota em que o algoritmo for aplicado. Para esta pesquisa, estes fatores não foram considerados, uma vez que em um primeiro momento o objetivo está centrado em verificar a viabilidade de aplicação do algoritmo ACO, considerando incialmente uma modelagem mais simples para o problema proposto. Como Bauru conta com um grande número de ruas, pontos de parada de ônibus, torna-se inviável a aplicação do algoritmo em todas as rotas de ônibus presentes atualmente. Portanto, para esse projeto foram escolhidas apenas algumas rotas, envolvendo algumas ruas e avenidas principais da cidade.

O protótipo foi implementado usando as facilidades fornecidas pela ferramenta de desenvolvimento do NetBeans (utilizando a linguagem JAVA). Inicialmente foram realizados testes utilizando um modelo básico do ACO para verificar a adequação da linguagem e do algoritmo, para isso, o clássico problema do Caixeiro Viajante foi utilizado. Uma vez constatada a viabilidade de utilização da linguagem e do algoritmo, um protótipo focado em dados da cidade de Bauru foi desenvolvido. Cabe destacar que testes com a ferramenta estão previstos como trabalhos futuros uma vez que, no estágio atual do protótipo, dados como, por exemplo, sentido das vias, pontos de enchentes e congestionamentos não estão disponíveis, impossibilitando uma análise mais profunda de utilidade da ferramenta no processo de criação de rotas reais.

O protótipo tem vários métodos vitais para o seu funcionamento. Portanto, foram desenvolvidos métodos responsáveis pela movimentação dos agentes, pelo depósito e evaporação dos ferômonios, para a atribuição de peso as rotas e, finalmente, para o envio de informações para o GoogleMaps para que os resultados pudessem ser visualizados.

4. Resultados

Nesta seção é apresentado o protótipo desenvolvido através da sua aplicação em uma rota, considerando dados reais do problema proposto na pesquisa.

Inicialmente foi montada uma árvore simples, conforme a figura seguinte, em que foi aplicado o algoritmo em busca da resolução. Nesta árvore o “0” representa o ponto de início e os demais números ruas que levam ao ponto de destino. Estes números representam abstrações referentes às ruas da cidade de Bauru. No contexto deste trabalho, por se tratar da resolução do problema de roteamento de ônibus, é feita uma razão entre tempo gasto e distância percorrida pelo veículo para atribuição de pesos às rotas.

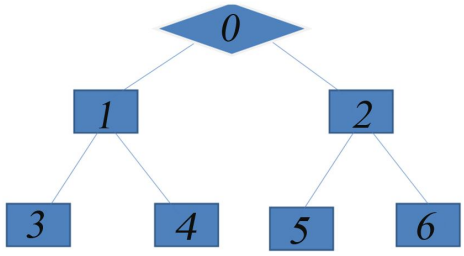

Figura 1 - Árvore teste para PCV

A primeira tela (figura 1) do protótipo exibe uma página, com um menu que possibilita a escolha de qual rota o usuário deseja calcular (que corresponderiam aos números da árvore anterior), um mapa gerado pela API (centralizado em Bauru) e finalmente um botão que dá inicio ao funcionamento do algoritmo.

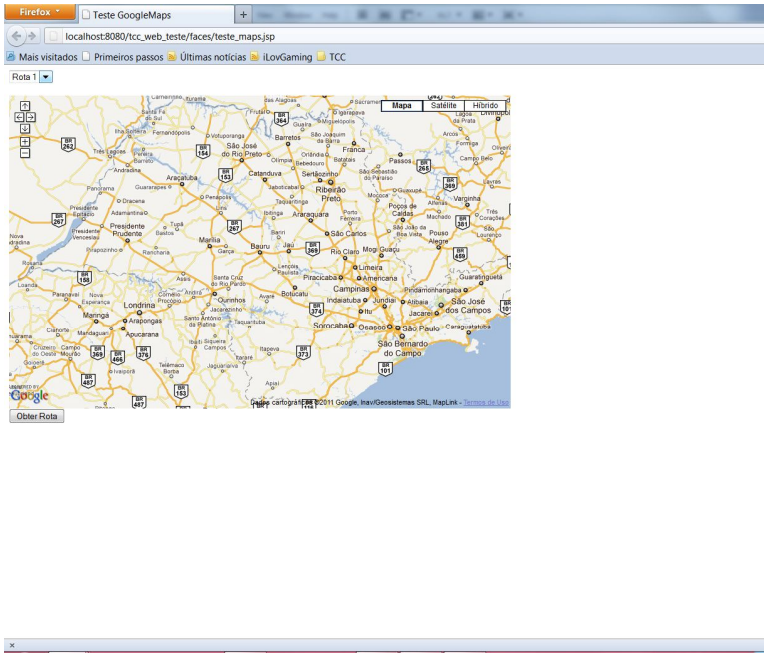

Figura 2 - Tela inicial

Para o cálculo da rota, inicialmente a tabela de ferômonios possui valores iguais, portanto: Fer[0]= Fer[1]= ... Fer[5] = 0.

Os valores anteriores são atualizados a cada interação, percorrendo todo o caminho entre o ponto inicial e final. Ao fim de cada interação, o trecho seguinte do algoritmo (figura 3) é executado:

```
75      cont = 0;
76      tri = constante / cont;
77      fer[(int) x] = (fer[(int) x] * (0.5) + tri) + fer[(int) x];
78      fer[(int) y] = (fer[(int) y] * (0.5) + tri) + fer[(int) y];
```

Figura 3- Atualização do Feromonio

Onde:

- *cont: resultado de um contato atribuído para receber a quantidade de vértices percorridos pelo agente;*
- *constante: valor atribuído pelo usuário;*
- *fer[]: vetor que define a quantidade de feromônio;*
- *x: indice do vetor fer[];*
- *y: indice do vetor fer[];*

Durante toda a execução do código, parte do trecho a seguir (figura 4) é executado. Ele tem como função calcular a probabilidade de movimentação do agente.

```
90    j = nome[i];
91    l = nome[i + 2];
92    prob = (fer[1] * valores[j]) / ((valores[j]) + (valores[l]));
93    prob = prob * 0x64;
94    prob2 = (fer[1] * valores[2]) / ((valores[j]) + (valores[l]));
95    prob2 = prob2 * 0x64;
96    double numero = (int) (Math.random() * 100);
```

Figura 4 - Cálculo da probabilidade na direção do movimento

Onde:

- *prob, prob2: pesos probabilisticos;*
- *valores[]: vetor que contém os pesos de cada trecho*

É então feita uma comparação entre os valores e um caminho é escolhido. Esse trecho se repetirá até que o fim da árvore gerada pelo desmembramento da rota seja atingido. Todo esse processo por ser repetido N vezes, baseando-se na necessidade de precisão do algoritmo e no custo computacional. É interessante mencionar que quanto maior a quantidade de interações mais preciso será o resultado final.

Após uma interação completa, podemos observar a mudança de valores dos feromônios. Um exemplo dessa mudança corresponde aos dados seguintes: Fer[0]=1.0, Fer[1]=3.0, Fer[2]=1.0, Fer[4]=4.5, Fer[5]=1.0.

Na figura XX pode-se ver o resultado do teste descrito fornecido pelo protótipo. Onde,

os agentes percorreram a árvore cem vezes, encontrando o caminho 0 -> 2 -> 5 como melhor.

```
run:
2.0
5.0
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Figura 5 - Resultado apresentado no *promt*

Devido à utilização de um algoritmo de otimização e ao número de iterações realizadas o caminho escolhido não é necessariamente o melhor caminho, mas de um modo geral, o algoritmo aponta para boas soluções.

Uma comparação pode ser feita entre o problema real e o teste realizado: apesar da solução não ser ótima, ela pode ser atingida de forma rápida e com baixo custo computacional e ainda ser uma melhor opção que a gerada de forma manual.

Por fim, o resultado é exibido conforme a imagem (figura 6) a seguir, onde a linha azul delimita o trecho escolhido pelo algoritmo e do lado direito é exibido o resultado detalhadamente.

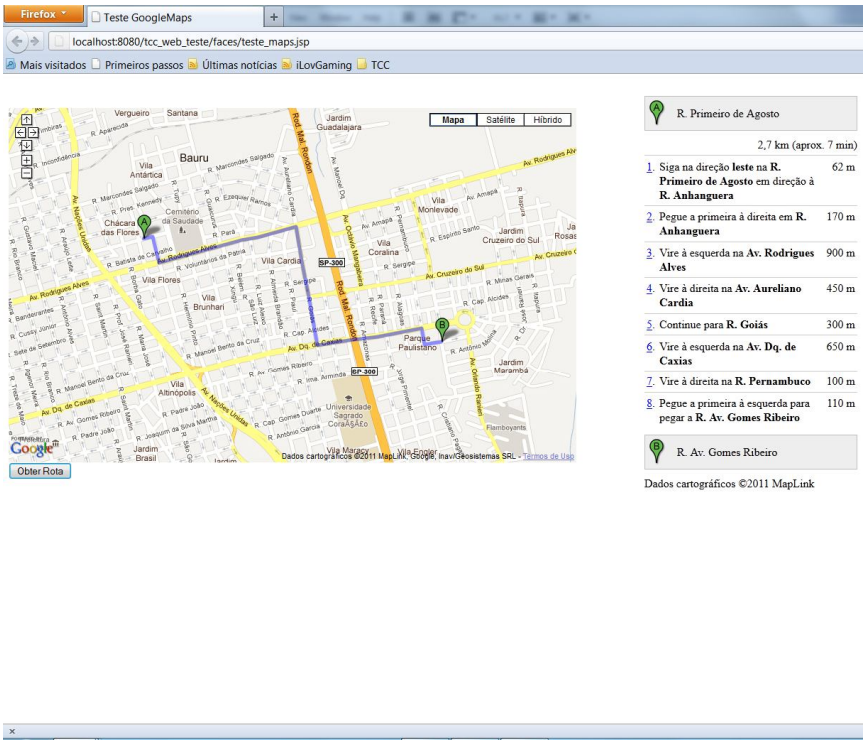

Figura 6 - Resultado apresentado no GoogleMaps

É importante notar que o controle do sentido das ruas foi abstraido do código final, porém a API do GoogleMaps possui um controle própio para isso, afim de evitar que o trecho

seja irreal.

5. Conclusão

O ACO mostra-se uma alternativa robusta para problemas de roteamento, seja ele qual for. Do básico problema do caixeiro viajante, a exemplos mais complexos como roteamento de rede ou ainda o proposto neste trabalho. É uma opção de otimização extremamente versátil que pode trazer excelentes resultados, se bem aplicada.

Entretanto, nem sempre se trata da melhor escolha para certos tipos de problema, por se tratar de um algoritmo que nem sempre traz soluções ótimas e depende do movimento baseado na probabilidade do ferômonio depositado pelos agentes (formigas) que trilham pelo caminho proposto.

O problema com as rotas de ônibus na cidade de Bauru é claro. Ruas com pouca vazão de trânsito, um número limitado de ônibus e a grande demanda tornam o serviço lento e defasado. Criar novas rotas e mudar as atuais é um processo lento e custoso que demanda um estudo especifico de cada rota, para que as alterações não interfiram no dia-a-dia dos usuários do serviço.

A implementação do ACO para resolver o problema proposto, pelo resultados preliminares obtidos, parece ser uma alternativa viável, contudo o grande número de variáveis que influenciam o ambiente por onde os ônibus transitam tornam complexa a tarefa de encontrar todas as variáveis que realmente influenciam no resultado. Fez-se necessário então abstrair uma grande quantidade de dados, diminuindo até um mínimo possível onde os agentes tenham menos problemas na movimentação.

Claramente, para que o protótipo desenvolvido possa ser aplicado em situações reais muitas outras variáveis, além de tempo e distância, deverão ser levadas em consideração. No estágio atual desta pesquisa apenas pode-se concluir que o algoritmo e sua implementação em linguagem Java, juntamente com a API do Google Maps, parecem bastante adequadas para geração de uma aplicação completa. A exibição dos resultados alcançados no GoogleMaps também provou-se um recurso indispensável para a compreensão do resultado final, de forma clara e eficaz. Certamente, a automatização do processo de geração de rotas pode trazer benefícios uma vez que, atualmente, todo este processo é feito de modo manual.

6. Referências

BABA, Cristina Mayumi et al. Otimização da colônia de formigas aplicada ao problema da programação e roteirização de veículos para o transporte de pessoas portadoras de deficiência. Encontro Nac. de Eng. de Produção, XXIV,2004, Florianópolis..ENEGEP.

BARTHOLOMEU, A. L. Frota de veículos em São Paulo cresce acima da média em março e amplia “nó” do trânsito da maior cidade do país. UOL, 2008 abril. Disponível em:http://noticias.uol.com.br/ultnot/especial/2008/transito/2008/04/24/ult5848u12.hjtm> Acesso em: 03 Abr 2011.

DORIGO, Marco ET AL. Ant Colony Optimization and Swarm Intelligence. Berlin: Springer, 2004.

DORIGO, M. e GAMBARDELLA, L.M. (1997) Ant Colonies for the Travelling Salesman

Problem. In Biosystems, v. 43, n. 2, p. 73- 81. Julho.

DORIGO, Marco. STÜTZLE, Thomas. Ant Colony Optimization. Massachusetts: Bradford, 2004.

FERNANDES, Rodrigo. Entendendo um pouco a API Google Maps. Disponível em: <http://javafree.uol.com.br/artigo/874529/Entendendo-um-pouco-a-API-Google-Maps.html>. Acesso em: 6 Jun 2011.

GARBE, B.(2006) Análise de Algoritmos de Roteamento Baseados em Formigas. Escola Politécnica, Universidade de São Paulo.

OLIVEIRA, U. Frota de veículos aumenta e qualidade do ar só piora. Rede Bom Dia, Bauru,19 Set 2011. Disponível em: <http://www.redebomdia.com.br/Noticias/DiaaDia/25318/Frota+de+veiculos+aumenta+e+qualidade+do+ar+so+piora>.Acesso em: 03 Abr 2011.

Ônibus: tarifa de Limeira estará entre as mais caras. Jornal de Limeira, Limeira, 08 nov 2009 Disponível em: <http://www.jornaldelimeira.com.br/site/noticias_detalhes.php?ID_Noticia=30795>. Acesso em: 06 Jun 2011.

OSHIRO, V. Plano de mobilidade em Bauru vai começar. Jornal da Cidade, Bauru. Mar. 2011

Proibição para caminhões começa a valer em dez vias do Morumbi. G1, São Paulo, 27 Set 2010 Disponível em: <http://g1.globo.com/sao-paulo/noticia/2010/09/proibicao-para-caminhoes-comeca-valer-em-dez-vias-do-morumbi.html>. Acesso em: 06 Jun 2011

TAVARES, R. F. N. (2005). Planejamento de vistorias usando robôs móveis autônomos e otimização pelo algoritmo de colônia de formigas. Pontifícia Universidade Católica do Paraná.

SILVEIRA, T. A utilização de meta-heurística de otimização denominada colônia de formigas para o restabelecimento de sistemas elétricos de potência. (2010). Universidade Federal de Alfenas.

STAGER, B. (2001). The Solving of Real World Problems using Evolutionary Algorithms. Undergraduate Thesis. University of Queensland. Outubro.

STÜTZLE, Thomas; DORIGO, Marco – *ACO Algorithms for the traveling salesman problem* [Em Linha]. New York: Swarm Intelligence Resources, 1999. [Acesso em: 09 Set 2011]. Disponível em WWW: <URL:http://staff.washington.edu/paymana/swarm/stutzle99-eaecs.pdf>