UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

Método GRASP e ACO em Otimização

Aluno: Pedro Henrique Rivera Santiago - RA: 120022
Orientadora: Prof. Dra. Márcia A. Gomes Ruggiero

Campinas
2015

Introdução

Travelling Salesman Problem (TSP), é um dos mais tradicionais e conhecidos problemas de programação matemática. Sua relevância se justifica pela sua grande aplicação prática, relação com outros modelos e dificuldade de solução exata, que motiva o estudo de heurísticas. Além disso, ele está presente em operações de máquinas e transporte entre pontos de manufatura, movimento de ferramentas de corte, roteamento de veículos, sequenciamento de DNA, distribuição de tarefas em plantas, trabalhos administrativos, entre outros. Visto isso, sua aplicabilidade torna-se mecanismo imprescindível em tempos de crise econômica, onde minimizar custos e preservar os lucros são objetivos almejados por todo o mundo.

Entretanto, a resolução destes problemas por métodos diretos torna-se, na maioria das vezes, inviável devido à dimensão e modelagem de aplicações reais.

Desta maneira, o projeto em questão apresenta uma abordagem heurística na resolução desses problemas, em particular, GRASP e ACO. Métodos que apesar de não serem diretos, com soluções exatas, apresentam resultados eficientes para aplicação prática. De modo que ao aplicarmos tais métodos, em específico, no problema de achar a menor rota entre um conjunto de n cidades, os resultados terão maior eficácia e perspectiva em termos de aplicação real.

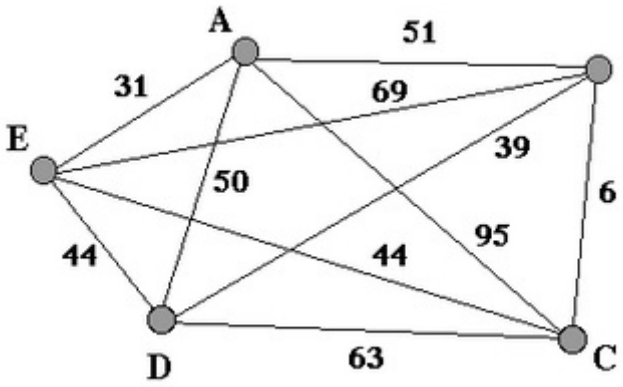
1 Problema do Caixeiro Viajante

O texto a seguir teve como base as referências [5], [6] *e* [9].

Travelling Salesman Problem (TSP) é um problema clássico da Otimização Combinatória e tem sido estudado por diversos pesquisadores de diferentes áreas. Isso se deve ao fato de possuir diversas aplicações práticas. O TSP pertence a uma classe de problemas não determinísticos de tempo não polinomial, classificado com *NP-difícil*. Por isso, sua resolução, utilizando métodos heurísticos, ganhou maior importância, principalmente quando aplicado a problemas de grande porte. Antigamente muitos vendedores, chamados de caixeiros viajantes, ganhavam a vida oferecendo seus produtos em diferentes cidades. Imagine que um caixeiro viajante escolhesse algumas cidades e precisasse encontrar o caminho mais curto para suas andanças. Em que ordem ele deveria percorrer as cidades escolhidas? Se fossem poucas, seria fácil descobrir. Mas e se o caixeiro estivesse em Brasília e quisesse percorrer as capitais de todos os 26 estados brasileiros? Em que ordem ele deveria visitar essas cidades? Como descobrir qual o caminho mais curto? Uma opção seria considerar todas as possíveis ordenações das 26 cidades. Isso levaria a um número total de possibilidades igual ao fatorial de 26 (26!), sendo que para cada uma dessas ordenações, são realizadas $n^2 - n$ operações básicas (soma, subtração, multiplicação e divisão), onde n é o número de cidades. Assim, se quiséssemos encontrar a melhor escolha, teríamos que calcular, aproximadamente, 26! distâncias totais, que é aproximadamente 4×10^{26}, uma para cada percurso, que necessita de 650 operações cada um para ser calculado. Um número tão expressivo, que se levássemos apenas um segundo para calcular a distância total de cada possível percurso, levaríamos cerca de um bilhão de vezes a idade do universo para encontrar a resposta. Isso acontece porque o número de percursos alternativos cresce assustadoramente quando aumentamos a quantidade de cidades. Esse é o chamado Problema do Caixeiro Viajante, uma questão matemática muito importante para a indústria.

1.1 Definição e Descrição

Seja um conjunto de pontos representando n cidades. O problema do Caixeiro Viajante consiste na determinação de uma rota que se inicia em uma cidade, passa por cada uma das outras do conjunto apenas uma vez, e retorna à origem da rota perfazendo uma distância total mínima. Esta rota é denominada ciclo Hamiltoniano de custo mínimo.
A representação deste problema pode ser feita através de um grafo completo $G = (V, A)$, onde V é o conjunto de vértices representando as cidades e A o conjunto de arcos ou arestas que conectam cada par de cidades $i, j \in V$. A cada aresta é atribuído um valor de custo c_{ij}, que é a distância da entre as cidades i e j.



1.2 Formulação Matemática

Em sua forma padrão, temos:

Função objetivo: $\sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} x_{ij})$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, ..., n \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, ..., n \tag{2}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., n \tag{3}$$

onde c_{ij} e x_{ij} são, respectivamente, os custos e as variáveis de decisão associados à designação do elemento i à posição j. A variável $x_{ij} = 1$ indica que a cidade j é visitada logo após a i, caso contrário $x_{ij} = 0$.
A função objetivo representa a minimização do somatório das distâncias entre as cidades da rota, a variável n representa o número total de cidades no problema. As restrições (1) e (2) garantem que cada cidade $i \in n$ será designada para exatamente uma cidade j. A restrição (3) define x como uma variável binária.
Entretanto, essa formulação não impede a ocorrência de sub-rotas, ou seja, mesmo respeitando as condições impostas, há a possibilidade da formação de sub-rotas formadas por subconjuntos de cidades, sem que haja conexão entre tais sub-rotas. Há algumas propostas para incluir na formulação original, restrições que impeçam a formação de sub-rotas.

Entre estas propostas, destacam-se a formulação MTZ, Miller-Tucker-Zemlin, que inclui a restrição na forma:

$$u_1 = 1,$$

$$\begin{aligned} 2 \leq u_i \leq n \qquad & \forall i \neq 1, \\ u_i - u_j + 1 \leq (n-1)(1 - x_{ij}) \qquad & \forall i \neq 1, \forall j \neq 1. \end{aligned}$$

onde $u_i (i = 1, ..., n)$ é uma variável extra.

E a proposta de Dantzig-Fulkerson-Johnson, que resulta da formulação DFJ, que está a seguir:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset V, S \neq \phi \tag{4}$$

Sendo S um subconjunto do conjunto $V = \{1, 2, ..., n\}$, e o símbolo $||$ denota a cardinalidade do conjunto.
A equação (4) é verificada para cada subconjunto S possível de V, ou seja, para $S = \{1\}$, $S = \{1, 2\}$, ..., $S = \{1, 2, ..., n\}$. O duplo somatório computa a quantidade de arcos presentes no grafo, enquanto o lado direito da desigualdade limita essa quantidade para o número de nós (S) menos um. Pois para que haja sub-rotas, formando um conjunto de cidades isoladas, é necessário que a quantidade de arestas seja igual a de nós. Assim essa restrição (4) garante que não serão formadas sub-rotas na solução.

2 Métodos Heurísticos

O texto a seguir teve como base as referências [1] *e* [7].

Em ciência da computação, em especial otimização, infelizmente alguns dos algoritmos desenvolvidos para resolver problemas gerados pela aplicação da matemática e física à realidade são ineficientes para casos de dimensão real. Assim, métodos exatos podem resultar em formulações com complexidades intratáveis e tempos de resolução excessivos e até proibitivos. Por esta razão, é criado um procedimento simplificador (*heurística*) que ou encontra soluções ótimas a maioria das vezes, mas não tem garantias de que sempre encontrará ou que tem processamento rápido, mas não tem provas de que será rápido em todas as situações.
As *heurísticas* são classificadas em:

- *Heusrísticas de construção*: são aquelas onde umas ou mais soluções são construídas elemento a elemento, seguindo algum critério heurístico de otimização, até que se tenha uma solução viável.
- *Heurística de busca*: partimos de uma solução inicial viável (em alguns casos podendo ser somente uma solução possível qualquer), tentando melhorar esta solução através de operações de troca, remoção ou inserção, até que não seja mais possível a melhoria ou algum outro critério de parada seja satisfeito.
- *Heurísticas híbridas*: resultantes da combinação de duas ou mais heurísticas com estratégias diferentes.
- *Metaheurísticas*: são heurísticas genéricas mais sofisticadas, onde uma heurística mais simples é gerenciada por um procedimento que visa explorar de forma eficaz a instância do problema e o seu espaço de soluções.

3 GRASP

O texto a seguir teve como base as referências [1] *e* [2].

3.1 Teoria

A metaheurística GRASP(*Greedy Randomized Adaptive Search Procedure*) é um algoritmo aplicado a problemas de otimização combinatória. É um método de busca construtiva, ou seja, partindo de uma solução vazia, constrói-se uma solução adicionando elementos a fim de melhorar a solução parcial, até que não seja mais possível melhorá-la. Sendo mais aprimorado que outros algoritmos da mesma classe como *Busca Tabu* e *Genético*, pois privilegia a geração de uma solução inicial de melhor qualidade para então utilizar a busca local apenas para pequenas melhorias.
Enquanto outros métodos preocupam-se somente em escolher sempre o melhor elemento (segundo algum critério) em cada iteração, o GRASP utiliza uma lista de candidatos a serem adicionados, e escolhe um aleatoriamente entre eles. O que faz a qualidade da solução variar devido a essas possíveis escolhas.
As etapas de construção do GRASP são:

- *Avaliação dos candidatos:* associa-se um valor para cada um dos elementos que ainda não foram adicionados, considerando a influência deles na qualidade da solução. Estabelecendo assim, o máximo (g_{max}) e o mínimo (g_{min}), entre eles.

- *Lista restrita de candidatos (RCL):* elabora-se a lista de possíveis elementos a serem adicionados, isso pode ser feito de duas maneiras:

  · *Baseado na Cardinalidade:* inclui os κ melhores candidatos da lista, onde κ é um número definido durante a programação do algoritmo.

  · *Baseado em Valor:* os candidatos pertencentes ao intervalo $[g_{min}, \mu]$, onde μ é obtido pela fórmula:

  $$\mu = g_{min} + \alpha(g_{max} - g_{min}) \tag{5}$$

  onde α é um valor entre 0 e 1 a ser definido pelo programador. Logo o tamanho da lista depende do valor de μ, que depende do parâmetro α adotado.
  Reagrupando os termos da equação (1), temos:

  $$\mu = (1 - \alpha)g_{min} + \alpha g_{max} \tag{6}$$

  Se escolhermos $\alpha = 0$ temos $\mu = g_{min}$, a construção se torna gulosa, ou seja, são escolhidos sempre os melhores elementos. Já, se escolhermos $\alpha = 1$ temos $\mu = g_{max}$, a construção é simplesmente aleatória. O tamanho da lista, influenciado pela escolha do parâmetro α, tem papel fundamental na performance do algoritmo.

- *Escolha aleatória do elemento:* seleciona-se aleatoriamente um elemento da RCL para ser adicionado solução parcial.

Repetimos estes três passos até não ser mais possível adicionar elementos. O critério de parada do algoritmo, normalmente, é um determinado número de iterações ou um determinado tempo.

3.2 Implementação

Algoritmo 1: GRASP

Data: n cidades e $MaxIte$ número máximo de iterações.
Inicialização:
Para cada aresta (i, j) do grafo, atribui-se uma distância d_{ij}, escolha aleatória de um nó para o começo do trajeto.
for $i \leftarrow 1$ **to** $MaxIte$ **do**
  for $j \leftarrow 2$ **to** n **do**
    · Determinar g_{min} e g_{max};
    · Adicionar elementos na RCL, segundo a equação:

$$\mu \leq g_{min} + \alpha(g_{max} - g_{min});$$

    · Escolha *randômica* de uma cidade pertencente a RCL;
  end
  if $L_j < L^*$ **then**
    · $L^* \leftarrow L_j$;
    · $S^* \leftarrow S_j$;
  end
end
return $S^* e\ L^*$

4 ACO

O texto a seguir teve como base as referências [3] *e* [4]*, e imagens de* [8].

4.1 Teoria

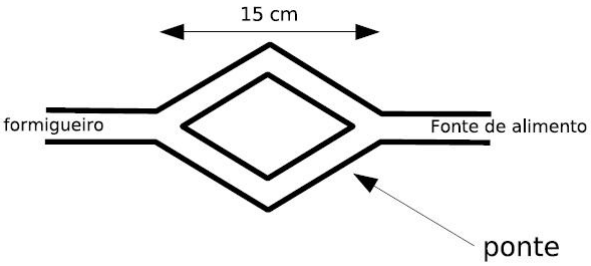
4.1.1 Inspiração Biológica

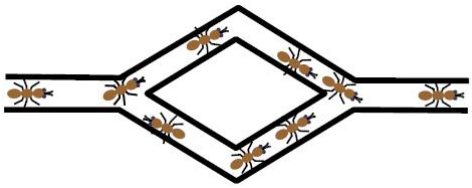
A metaheurística ACO (*Ant Colony Optimization*) cuja fonte de inspiração é o comportamento de colônias de formigas, baseia-se na observância de que as formigas são capazes de encontrar o menor caminho entre o ninho e a fonte de alimento. Esses comportamentos são explorados em colônias artificias de formigas para a obtenção de soluções aproximadas aos problemas de Otimização Combinatória, como por exemplo o problema do Caixeiro Viajante.
No mundo real, as formigas andam sem rumo (pelo menos inicialmente) até que, encontrada comida, elas retornam à colônia deixando um rastro de feromônio depositado no chão, formando uma trilha de feromônio. Se outras formigas encontram um desses rastros, elas tendem a não seguir mais caminhos aleatórios. Em vez disso, seguem a trilha encontrada, retornando e inclusive enfatizando se acharam alimento. A trilha ajuda a formiga a achar o caminho de volta e as outras a encontrar a fonte de alimentos
Com o transcorrer do tempo, entretanto, as trilhas de feromônio começam a evaporar, reduzindo, assim, sua força atrativa. Quanto mais formigas passarem por um caminho predeterminado, mais tempo será necessário para o feromônio da trilha evaporar. Analogamente, elas passarão mais rapidamente sobre um caminho curto, o que implica no aumento da densidade de feromônio depositado antes que ele comece a evaporar. A evaporação do feromônio também possui a vantagem de evitar a convergência para uma solução local ótima, se a evaporação não procedesse, todas as trilhas escolhidas pelas primeiras formigas tornariam se excessivamente atrativas para as outras e, neste caso, a exploração do espaço da solução se reduziria consideravelmente.
Todavia, quando uma formiga encontra um bom (curto) caminho entre a colônia e a fonte de alimento, outras formigas tenderão a seguir o mesmo, que eventualmente torna um determinado caminho mais interessante. A ideia do algoritmo é imitar este comportamento através de formigas artificias que caminham por um grafo que por sua vez representa o problema a ser resolvido.

4.1.2 O Experimento da Ponte Binária

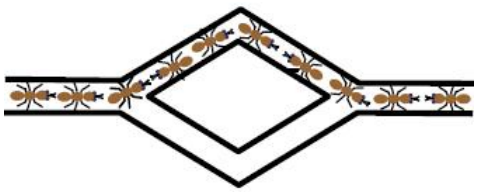
Modelo do experimento realizado por Deneubourg et al.,1990, para estudar o comportamento formigas. Inicialmente, consistiu na utilização de caminhos de tamanhos idênticos.

No início as formigas são deixadas livres para escolher o caminho aleatoriamente. Não há indicio de feromônio.



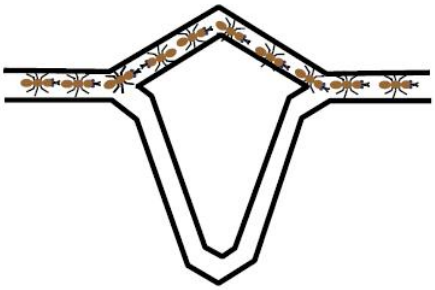


As formigas convergem para um dos caminhos com igual probabilidade.
Devido a flutuações aleatórias, um número maior de formigas selecionará um dos caminhos, que com o tempo apresentará um nível de feromônio maior que o outro. Assim, uma das pontes atrairá as formigas com maior probabilidade.
Ao final, podemos ter:



Ou:

Usando pontes de tamanhos distintos, um dos caminhos é duas vezes o tamanho do outro.
O menor caminho é de fato o preferido, pois as formigas que o escolheram são as que chegam à comida e voltam ao ninho, portanto o nível de feromônio neste é mais alto, o que estimulará outras formigas a escolherem tal caminho, resultando em uma convergência para o mesmo. Apesar de umas grande porcentagem das formigas escolherem o menor caminho, uma pequena porção delas ainda continua, por um tempo, utilizando o maior caminho. Isto pode ser interpretado como uma "exploração do ambiente"à procura de novas fontes de alimento.

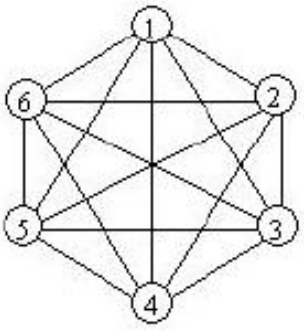


4.2 Implementação

A primeira metaheurística de otimização por colônia de formigas foi o AS (*Ant Colony*), posteriormente propuseram o algoritmo ACS (*Ant Colony System*) para o problema do Caixeiro Viajante. O sistema de colônia de formigas é uma forma de resolução de problemas que simulam o comportamento de um grupo de formigas na busca pelo menor caminho.
Em um sistema ACS, duas tarefas são principais:

- A construção de uma representação do problema de uma forma versátil, normalmente uma estrutura na forma de um grafo e que permita uma regra probabilística de transição entre os nós baseada na trilha de feromônios e no valor de cada arco.
- O desenvolvimento de uma heurística para transição de nós que possa avaliar a qualidade dos caminhos tomados.

O problema do Caixeiro Viajante busca o caminho que minimize o trajeto entre os diversos nós interligados através de arcos em uma estrutura semelhante a um grafo. Como o da figura abaixo:



Para cada arco (i, j) do grafo, é definida uma variável τ_{ij}, conhecida como trilha artificial de feromônio. Inicialmente este τ_{ij} é igual para todos os arcos da rede. Cada formiga **k** constrói uma solução a partir de um dos nós do grafo de forma randômica. Assim, a variável τ_{ij} é incrementada conforme o passar das formigas e decrementada a cada ciclo. A intensidade da trilha de feromônio definirá a utilidade desta para as formigas, isto é, quanto maior a quantidade de feromônio, melhor é esta trilha e as formigas tendem a utilizar esse percurso em detrimento dos outros.
A cada nó, a formiga artificial executa uma função estocástica para calcular a probabilidade de utilização dos arcos.
Tem-se uma fórmula probabilística onde cada formiga **k** constrói o seu caminho movendo-se através de uma sequência de locais vizinhos. O termo τ_{ij} controla a deposição do feromônio nas arestas, enquanto o η_{ij} é um valor heurístico relacionado à natureza do problema e que permite uma maior exploração. Já os parâmetros α e β controlam a intensidade do feromônio τ_{ij}, e a qualidade da aresta η_{ij}, respectivamente. A equação é dada por:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}} \tag{7}$$

onde:

- p_{ij}^{k}: é a probabilidade da formiga **k**, que se encontra na cidade 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). No caso do problema do Caixeiro Viajante, adota-se o inverso $\frac{1}{d_{ij}}$ do valor da distância entre os nós i e j;
- $l \in J_i^k$: conjunto de pontos ainda não visitados pela formiga k, que se encontra atualmente no ponto i;
- α: é um parâmetro que pondera a importância relativa da trilha de feromônio τ_{ij} na decisão de movimentação da formiga;

- β: valor heuristicamente escolhido, que pondera a influência relativa da distância η_{ij} entre os nós i e j no processo de decisão;

Podemos observar que se $\alpha = 0$, as formigas seguirão a heurística de vizinho mais próximo passo, enquanto que se $\beta = 0$, as formigas selecionarão um caminho com maior nível de feromônio e pode ocorrer uma estagnação com o passar das iterações, levando o algoritmo a pontos sub-ótimos.
A equação (7) mostra que a preferência da formiga por um determinado caminho é maior para os caminhos com maior nível de feromônio e com menor distância. A atualização deste é realizada local e globalmente.
A local é para evitar uma aresta muito forte que está sendo escolhida por todas as formigas, e ela é motivada pela evaporação do feromônio no caminho por onde cada formiga k passar. Esse processo aumenta a exploração das formigas. É dada por:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \qquad \tau_0 = \frac{1}{(n \cdot L_{nn})} \tag{8}$$

onde n é o número de nós do grafo, L_{nn} um valor heurístico de menor caminho encontrado por alguma formiga e ρ é um parâmetro que determina a velocidade da evaporação do feromônio.
Já a global é pretendido recompensar as arestas que pertencem a rotas mais curtas. Uma vez que as formigas artificias terminam seus trajetos, a melhor formiga deposita feromônio em arestas visitadas que pertencem a seu trajeto e as outras permanecem inalteradas. Assim, a cada arco (i, j) da rede, adiciona-se uma quantidade de feromônio proporcional ao tamanho da rota obtida, ou seja, quanto mais curta a rota, maior será a quantidade de feromônio depositada. Essa atualização é dada por:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}, \qquad \rho \in [0, 1] \tag{9}$$

onde,

$$\Delta\tau_{ij} = \begin{cases} 1/L_n, & se\ (i,j)\ usado \\ 0, & c.c. \end{cases} \tag{10}$$

O primeiro termo das equações (8) e (9) é responsável pela evaporação do feromônio. O parâmetro ρ é utilizado para que os caminhos menos frequentados sejam esquecidos com o passar do tempo. O segundo termo da equação (9) é responsável por aumentar a concentração de feromônio apenas nos arcos visitados pela melhor formiga, onde L^k é a distância total percorrida na rota construída pela melhor formiga da iteração. Quanto menor a rota, maior a quantidade de feromônio depositada.
Esse procedimento se repete até que um número máximo de iterações tenha sido alcançado ou caso não verifique mais melhorias nas soluções encontradas.

Algoritmo 2: ACS

Data: n cidades, $MaxIte$ e $NForm$, número máximo de iterações e formigas, respectivamente.

Inicialização:

Para cada aresta (i, j) do grafo, atribui-se uma distância d_{ij} e, estabelece-se um nível de feromônio τ_{ij}.

Para cada formiga k, escolhe-se um nó aleatório para iniciar o trajeto.

for $t \leftarrow 1$ **to** $MaxIte$ **do**

  for $k \leftarrow 1$ **to** $NForm$ **do**

    for $j \leftarrow 2$ **to** n **do**

      · Formiga k determina próxima cidade segundo a regra probabilística influenciada pelo nível de feromônio τ_{ij} e a métrica heurística de afinidade η_{ij},

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}};$$

      · Após cada transição da formiga k, aplique a regra de *atualização local*,

$$\tau_{ij}(t) \leftarrow (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0;$$

    end

    $L_k(t) \leftarrow$ tamanho da menor rota encontrada pela formiga k;

    $S_k(t) \leftarrow$ Sequencia de cidades visitadas na rota;

  end

  if $L_k(t) < L^*$ **then**

    · $L^* \leftarrow L_k(t)$;

    · $S^* \leftarrow S_k(t)$;

  end

  Aplicar a regra de *atualização global* do feromônio.

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij},$$

$$onde \qquad \Delta\tau_{ij} = \begin{cases} 1/L_k(t), & se\ (i,j)\ usado \\ 0, & c.c. \end{cases};$$

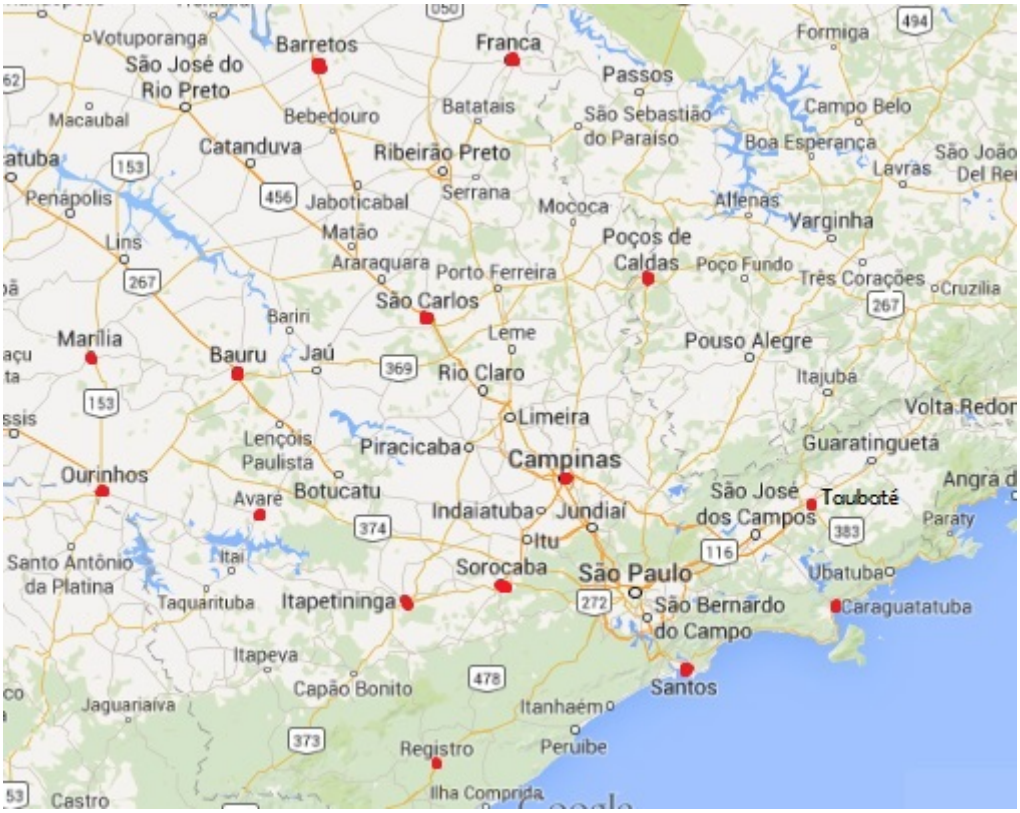
end

return $S^* e\ L^*$

5 Aplicação e Resultados

A imagem e distância rodoviária (seguindo estradas de rodagem oficias) entre as cidades foram retiradas da referência [10]

A fim de buscar uma aplicação direta do Problema do Caixeiro Viajante a situações reais. Foram escolhidas 15 cidades dispostas por todo o estado de São Paulo. Em seguida, foi mapeado as estradas e suas respectivas distâncias entre cada par destas cidades. Onde dado um ponto inicial aleatório, através das heurísticas GRASP e ACO, busca-se uma sequencia de cidades para esse trajeto ótimo, ou seja, uma rota que interligue todas as cidades minimizando o tamanho do percurso total. A disposição destas foi conforme a figura abaixo.



Para facilitar a apresentação dos dados e resultados obtidos, as cidades foram enumeradas conforme a Tabela 1. Na Tabela 2 está a distância em quilômetros, par a par, entre as cidades. Essa matriz é o dado principal para implementação dos algoritmos. É importante salientar que esta possui uma propriedade em particular, é simétrica. Isso facilita os cálculos, manipulações e atualizações feitas nos métodos implementados, já que esta pode ser tratada como triangular superior ou inferior, diminuindo significativamente a quantidade de dados armazenados e acessados na execução dos métodos.

01	Campinas	**06**	Franca	**11**	Ourinhos
02	Sorocaba	**07**	Bauru	**12**	Registro
03	Avaré	**08**	Barretos	**13**	Itapetininga
04	Santos	**09**	São Carlos	**14**	Caraguatatuba
05	Taubaté	**10**	Marília	**15**	P. de Caldas

Tabela 1: Cidades

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**
01	0	87	253	175	190	307	261	335	146	373	359	359	156	233	168
02	87	0	188	169	224	386	251	395	206	356	310	156	71	265	257
03	253	188	0	334	389	350	124	333	211	220	128	294	148	433	325
04	175	169	334	0	203	473	399	506	314	504	445	181	239	231	334
05	190	224	389	203	0	499	468	526	340	573	497	324	297	118	240
06	307	386	350	473	499	0	297	140	186	358	415	565	408	541	251
07	261	251	124	399	468	297	0	258	156	108	124	373	221	494	300
08	335	395	333	506	526	140	258	0	193	298	379	596	410	568	327
09	146	206	211	314	340	186	156	193	0	269	272	373	223	379	169
10	373	356	220	504	573	358	108	298	269	0	95	490	335	610	414
11	359	310	128	445	497	415	124	379	272	95	0	350	216	538	418
12	259	156	294	181	324	565	373	596	373	490	350	0	158	366	412
13	156	71	148	239	297	408	221	410	223	335	216	158	0	335	322
14	233	265	433	230	118	541	494	568	379	610	538	366	335	0	388
15	168	257	325	334	240	251	300	327	169	414	418	412	322	388	0

Tabela 2: Matriz de distância entre as cidades

Ambos os métodos foram rodados em uma máquina com processador Intel(R) Core(TM) i3-2330M CPU @ 2.20 GHz 2.20 GHz.
No método GRASP o dado de entrada é exatamente a Tabela 2. O número limite de iterações foi fixado em 100, valor determinado experimentalmente. Verificando que, é um número suficiente para o "aquecimento" do algoritmo, ou seja, para que obtenha uma aproximação inicial e, não haja uma estagnação do método. Para cada ponto inicial, devido ao caráter aleatório na escolha do elemento da lista de candidatos, o algoritmo foi executado 20 vezes em busca de uma melhor solução.
Assim, dada a cidade inicial, obtêm-se a rota de menor caminho S^*, o tamanho do percurso, em quilômetros, L^* e o respectivo tempo, t, de execução do programa para achar essa solução, em segundos. Conforme apresentado na Tabela 3.

Cidade Inicial	**Percurso S***	**L* (km)**	**t(s)**
Campinas	1 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 8 - 6 - 15 - 5 - 14 - 4 - 12	2146	0.1291
Sorocaba	2 - 1 - 15 - 9 - 6 - 8 - 7 - 10 - 11 - 3 - 13 - 12 - 4 - 5 - 14	2147	0.1215
Avaré	3 - 11 - 10 - 7 - 9 - 1 - 2 - 13 - 12 - 4 - 14 - 5 - 15 - 6 - 8	2110	0.1216
Santos	4 - 12 - 2 - 1 - 13 - 3 - 11 - 10 - 7 - 9 - 8 - 6 - 15 - 5 - 14	2157	0.1220
Taubaté	5 - 14 - 4 - 12 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 1 - 15 - 6 - 8	2096	0.1235
Franca	6 - 8 - 9 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 15 - 5 - 14 - 4 - 12	2063	0.1232
Bauru	7 - 3 - 11 - 10 - 8 - 6 - 9 - 15 - 1 - 2 - 13 - 12 - 4 - 5 - 14	2126	0.1237
Barretos	8 - 6 - 9 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 15 - 5 - 14 - 4 - 12	2056	0.1209
São Carlos	9 - 6 - 8 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 15 - 5 - 14 - 4 - 12	2158	0.1205
Marília	10 - 11 - 3 - 7 - 9 - 1 - 2 - 13 - 12 - 4 - 14 - 5 - 15 - 6 - 8	2126	0.1223
Ourinhos	11 - 10 - 7 - 3 - 2 - 13 - 12 - 4 - 14 - 5 - 1 - 15 - 9 - 6 - 8	2127	0.1213
Registro	12 - 4 - 14 - 5 - 15 - 1 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 6 - 8	2057	0.1214
Itapetininga	13 - 2 - 1 - 15 - 9 - 6 - 8 - 7 - 10 - 11 - 3 - 12 - 4 - 5 - 14	2206	0.1204
Caraguatatuba	14 - 5 - 4 - 12 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 1 - 15 - 6 - 8	2069	0.1207
P. de Caldas	15 - 9 - 6 - 8 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 5 - 14 - 4 - 12	2109	0.1204

Tabela 3: GRASP - Percursos ótimos

O parâmetro α foi variado em todo o seu intervalo $[0, 1]$, a fim de determinar qual valor obteria os melhores resultados para as rotas. Este valor encontrado foi de $\alpha = 0.2$, utilizado para os dados apresentados na Tabela 3.
Para α igual a 0.0 e 0.1 obtemos boas soluções, mas para esses valores o algoritmo se comporta de forma gulosa, tendo uma visão das soluções local, não olhando as cidades globalmente, o que prejudica a solução construída. Para α com valores maior ou igual a 0.4, o método se torna aleatório, e quanto mais próximo de 1.0 é esse valor mais se evidencia esse comportamento. Isso faz com que as soluções oscilem entre boas e ruins, sendo necessário, em certos momentos, executar o programa pelo menos 20 vezes, para que uma solução satisfatória seja encontrada.
O tempo de execução do método ficou em torno de um décimo de segundo (0.1 s).

No método ACS há dois dados de entrada para a inicialização. A matriz de feromônio τ, começando com todos os arcos tendo o feromônio igual a 1 e posteriormente modificados conforme as atualizações, local e global. A matriz η que representa a atratividade de cada arco em relação a distância entre eles, não podendo ser utilizada diretamente como na Tabela 2. Isso se deve ao fato das distâncias serem da ordem de 10^2 e o valor heurístico utilizado ser o seu inverso. Causando uma diferença nos dados de feromônio e distância de duas casas decimais, fazendo com que a cada iteração os valores, devido as atualizações, se tornassem cada vez mais próximos de zero e diminuindo a importância do valor heurístico da distância na busca por rotas.
A solução encontrada para igualar a escala dos valores τ_{ij} e η_{ij} em cada arco foi normalizar a matriz de distâncias segundo a norma infinito descrita abaixo.

Dado $A = [a_{ij}]_{m \times n}$ uma matriz $m \times n$ A norma infinito ou norma do máximo da matriz A, denotada por $\|A\|_\infty$, é o número não negativo:

$$\|A\|_\infty = max \sum_{j=1}^{n} |a_{ij}| \qquad para\ i = 1, 2, ..., m;$$

(A maior soma absoluta das linhas).
Mantendo o número máximo de iterações igual a 100 e a mesma máquina com o processador já descrito. Para o Método ACS, as melhores rotas obtidas seguem conforme a Tabela 4.

Cidade Inicial	**Percurso S***	**L* (km)**	**t(s)**
Campinas	1 - 15 - 6 - 8 - 9 - 7 - 10 - 11 - 3 - 13 - 2 - 12 - 4 - 5 - 14	2116	2.0677
Sorocaba	2 - 1 - 15 - 6 - 8 - 9 - 7 - 10 - 11 - 3 - 13 - 12 - 4 - 5 - 14	2134	2.0164
Avaré	3 - 11 - 10 - 7 - 9 - 8 - 6 - 15 - 1 - 2 - 13 - 12 - 4 - 5 - 14	2057	2.0266
Santos	4 - 12 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 8 - 6 - 15 - 1 - 5 - 14	2103	1.9963
Taubaté	5 - 14 - 4 - 12 - 13 - 2 - 1 - 15 - 6 - 8 - 9 - 7 - 10 - 11 - 3	2084	1.9963
Franca	6 - 8 - 9 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 15 - 5 - 14 - 4 - 12	2063	1.9718
Bauru	7 - 10 - 11 - 3 - 13 - 2 - 12 - 4 - 5 - 14 - 1 - 15 - 9 - 6 - 8	2104	2.0057
Barretos	8 - 6 - 9 - 7 - 10 - 11 - 3 - 13 - 2 - 1 - 15 - 5 - 14 - 4 - 12	2056	2.0251
São Carlos	9 - 7 - 10 - 11 - 3 - 13 - 2 - 12 - 4 - 14 - 5 - 1 - 15 - 6 - 8	2141	2.0045
Marília	10 - 11 - 3 - 7 - 9 - 8 - 6 - 15 - 1 - 2 - 13 - 12 - 4 - 5 - 14	2073	2.0804
Ourinhos	11 - 10 - 7 - 3 - 13 - 2 - 12 - 4 - 5 - 14 - 1 - 15 - 9 - 6 - 8	2100	1.9997
Registro	12 - 4 - 14 - 5 - 15 - 1 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 6 - 8	2057	1.9772
Itapetininga	13 - 2 - 1 - 15 - 6 - 8 - 9 - 7 - 10 - 11 - 3 - 12 - 4 - 5 - 14	2193	1.9823
Caraguatatuba	14 - 5 - 4 - 12 - 13 - 2 - 1 - 15 - 6 - 8 - 9 - 7 - 10 - 11 - 3	2057	2.0079
P. de Caldas	15 - 1 - 5 - 14 - 4 - 12 - 2 - 13 - 3 - 11 - 10 - 7 - 9 - 6 - 8	2075	1.9985

Tabela 4: ACS - Percursos ótimos

O parâmetro ρ foi variado em todo o seu intervalo $[0, 1]$, a fim de determinar qual valor obteria os melhores resultados para as rotas. Este valor encontrado foi de $\rho = 0.7$, utilizado para os dados apresentados na Tabela 4. Ele foi determinado através de testes computacionais. Para valores iguais ou abaixo de 0.5 a taxa de evaporação do feromônio é muito rápida, tirando significativamente sua influência na determinação das rotas, o algoritmo acaba se baseando nas distâncias. Para valores acima de 0.7 a taxa de evaporação é tão lenta que torna as rotas viciadas, diminuindo a exploração das formigas na procura de melhores percursos.
Quanto aos valores de α e β que ponderam a importância do feromônio e da distância, respectivamente, na decisão de movimentação da formiga, foram determinados por testes computacionais. Chegando a $\alpha = 1.0$ e $\beta = 3.0$. Estes valores, tomados dessa forma, ponderam de forma equilibrada a influência de cada fator na fórmula probabilística do processo de decisão da formiga em escolher a próxima cidade. Obtendo as melhores soluções a partir de cada cidade.
O tempo de execução do método ficou, em geral, na casa de dois segundos (2.0 s).

Conclusão

Tanto o método GRASP como o ACS, apesar de não apresentarem garantia de que sempre encontrarão soluções ótimas e um tempo eficiente, apresentaram uma rota satisfatória em um tempo resolução aceitável para esse problema do Caixeiro Viajante aplicado a 15 cidades do Estado de São Paulo.
A diferença no tamanho do percurso total entre os métodos foi, em geral, de 10 a 20 quilômetros, chegando a ser de 34 quilômetros em um dos casos, sendo as melhores soluções encontradas pela heurística ACS. Isso se deve ao fato desse algoritmo apresentar uma exploração global de maior qualidade quando comparado ao método GRASP, que se caracteriza por uma busca local melhor.
Quanto ao tempo de execução da rotina dos métodos, enquanto o GRASP obtém a solução em um décimo de segundo, o ACS faz o mesmo em dois segundos (em um espaço amostral de 15 cidades e fixando em 100 iterações). A diferença ocorre pelo fato da heurística ACS apresentar um loop interno a mais, referente a quantidade de formigas que buscam uma rota.
Devido a aleatoriedade presente na escolha de elementos presentes na lista de candidatos em ambos os métodos, há necessidade executar as rotinas mais de uma vez, quando se busca melhorar ainda mais as soluções. Entretanto, essa diferença chega a ser no máximo de 50 quilômetros no tamanho total da rota.
Como forma mais eficiente para se atacar problemas do mesmo tipo, com maior escala, 100 cidades por exemplo, fica a proposta de um algoritmo híbrido GRASP+ACS, utilizando a melhor performance na busca local do método GRASP e o refinamento global na solução que o método ACS oferece.

Referências Bibliográficas

[1] G. Zäpfel, R. Braune & M. Bögl. Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics. Springer. 2010.

[2] T. Feo and M. G. Resende; Greedy Randomized Adaptive Search Procedures. J. of Global Optimizations 6: 109-13.1995

[3] M. Dorigo e T. Stützle. Ant Colony Optimization. A Bradford Book, 2004.

[4] M. Dorigo, V. Maniezzo e A. Colorni. The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics B, 26(1):29-41, 1996.

[5] A. Croes. A Method for Solving Traveling-Saleman Problem. Opns. Res. 3, 791-812. 1958

[6] M. J. Souza Freitas. Otimização Combinatória. Universidade Federal de Ouro Preto, Departamento de Computação.

[7] Tese de mestrado. C. Zim Zapelini. Um estudo Abrangente sobre Metaheurística, incluindo um Histórico. Universidade de São Paulo, Instituto de Matemática e Estatística.

[8] Slide. Estéfane G. M. de Lacerda. A otimização Colônia de Formigas. UFRN, Departamento de Engenharia da Computação e Automação. 2008.

[9] Artigo. Pataki, G. Teaching Interger Programming Formulations Using the Traveling Salesman Problem. SIAM REVIEW, Vol. 45, No. 1, pp. 116-123, 2003.

[10] Site: www.google.com.br/maps