Análise Empírica da Utilização de Metaheurísticas ao Problema do Caixeiro Viajante

Omitido para revisão

[1]Omitido para revisão

Abstract. *This paper has the purpose to make a empirical research on the application of metaheuristics on Traveling Salesman Problem (TSP), which is a NP-Hard problem. Greedy Randomized Adaptive Search Procedure (GRASP) and Simulated Annealing (SA) metaheuristics were studied. GRASP uses a global search followed by a local search on the founded solutions, being the SA applied at this second stage. In order to validate this approach, the experiments were conducted in different Traveling Salesman Problem instances, from the library TSPLIB. The results were compared, based on the solutions quality and computational time, with references values from TSPLIB and other approaches, such as Tabu Search.*

Resumo. *Este artigo tem por objetivo realizar um estudo empírico da aplicação de metaheurísticas no Problema do Caixeiro Viajante, que faz parte da classe NP-completo. As metaheurísticas estudadas a Greedy Randomized Adaptive Search Procedure (GRASP) e o Simulated Annealing (SA). O GRASP realiza uma busca global, seguido de uma busca local nas soluções encontradas, sendo o SA incorporado nesta etapa. Para validar esta proposta, foram realizados experimentos em diferentes instâncias do Problema do Caixeiro Viajante, presentes na biblioteca TSPLIB. Os resultados obtidos foram comparados com os relação à qualidade da solução final e no tempo de execução, com valores de referência da TSPLIB e com outras abordagens, como a Busca Tabu.*

1. Introdução

O Problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP) é um problema clássico de otimização combinatória [Dantzig et al. 1954, Black 2005]. Ele é conhecido por fazer parte da classe de problemas NP-completo, pois, até hoje não existe um algoritmo que o resolva em tempo polinomial. Além disso, este problema serve como base para diversas aplicações na área de transporte, projeto de circuitos integrados, robótica, entre outros [Gutin and Punnen 2002].

Um caixeiro viajante deseja visitar n cidades de uma certa região, onde duas cidades estão conectadas entre si por um caminho. Dessa maneira, ele deverá passar por cada uma das cidades apenas uma vez, percorrendo o menor caminho. Existem diferentes variações definidas para este problema, como por exemplo, o caixeiro viajante simétrico e o assimétrico. Na primeira considera-se que a distância para ir de uma cidade A até B é a mesma de B para A, e na segunda estas podem diferir [Gutin and Punnen 2002].

Por ser um problema de difícil tratamento, três principais estratégias de ataque podem ser destacadas: (a) Algoritmos exatos, (b) Heurísticas, e (c) Casos especiais. Em (a) a solução exata é encontrada, porém, o tempo computacional aumenta exponencialmente

de acordo com o número de cidades. Na segunda estratégia, soluções são encontradas rapidamente, e com grandes probabilidades de serem ótimas (nem sempre são ótimas). A última estratégia considera casos especiais, com os quais é mais fácil encontrar heurísticas apropriadas. Um exemplo é o caso do TSP assimétrico.

Neste trabalho foram estudados duas metaheurísticas, o Procedimento de Busca Adaptativo, Aleatório e Guloso (GRASP - *Greedy Randomized Adaptive Search Procedure*) e o Recozimento Simulado (SA - *Simulated Annealing*), para serem aplicadas ao problema do caixeiro viajante.

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é um algoritmo usualmente aplicado em otimização combinatorial [Pitsoulis and Resende 2002, Resende 2001, Silveira 1999]. Em cada iteração uma solução gulosa aleatória é contruída e em seguida é refinada através de uma busca local.

O SA (*Simulated Annealing*) é um algoritmo de busca local que a partir de uma solução candidata, ele se movimenta iterativamente para uma solução vizinha que melhor resolve o problema [Kirkpatrick et al. 1983]. A principal diferença do SA para outros algoritmos de busca local, como por exemplo o *Hill Climbing*, é a possibilidade que o SA tem de aceitar uma solução pior que a atual (também conhecida como solução candidata) durante o processo iterativo. Isto faz com que o SA não fique "preso" em mínimos locais e consiga atingir seu objetivo, que é uma solução próxima à do mínimo global.

A abordagem a ser estudada é a integração entre o GRASP e o SA. A idéia principal é utilizar o SA para realizar a busca local nas soluções gulosas geradas pelo GRASP. Para validar esta proposta foram realizados experimentos utilizando diferentes instâncias do problema do caixeiro viajante assimétrico presentes na TSPLIB[1]. Os resultados foram comparados com a utilização do GRASP combinado com a Busca Tabu.

Na seção 2 é apresentado o modelo de representação do problema. Nas seções 3, 4 e 5 são discutidas as metaheurísticas estudadas. Os experimentos realizados e os resultados obtidos estão na seção 6, seguido de algumas considerações finais na seção 7.

2. Representação do Problema

A modelagem do problema é uma etapa importante para a sua resolução.Dependendo da representação das soluções algumas operações a serem aplicadas pelas metaheurísticas podem ser inviabilizadas. Alguns autores propõem estruturas de dados que diminuem o tempo computacional do algoritmo [LEE 2006].

Neste trabalho, uma solução é representada como um vetor de n posições, onde n é o número de cidades. Em cada posição são armazenadas as cidades na ordem em que são visitadas. Com esta estrutura, a função de energia (ou função objetivo) pode ser calculada com a soma das distâncias entre as cidades adjacentes.

3. Greedy Randomized Adaptive Search Procedure (GRASP)

A metaheurística GRASP possui um grande destaque na literatura para problemas de otimização, isso devido aos bons resultados obtidos e à grande facilidade de adaptação

[1] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

para resolução de problemas. O GRASP é composto por duas etapas principais: (a) construção da solução, onde uma solução possível é criada, e (b) busca local, que encontra uma solução ótima local [Pitsoulis and Resende 2002, Resende and Ribeiro 2003].

3.1. Descrição do Algoritmo

A Fig. 1 ilustra o bloco principal do GRASP [Resende and Ribeiro 2003].

```
1. f* = ∞
2. Repita até alcançar 10 iterações sem melhorar x*
     a) Gera uma solução gulosa aleatória x
     b) Melhora a solução x com um procedimento de busca
        local
     c) Se f(x) < f*
          (i)  f* = f(x)
          (ii) x* = x
3. Retorna x*
```

Figura 1. Pseudo-código para a metaheurística GRASP.

3.2. Construção da Solução Semi-Gulosa Aleatória

Para construir uma solução, o processo é iniciado com a escolha de uma cidade inicial qualquer. A partir desta cidade cria-se uma lista com seus vizinhos, ordenada pelas distâncias. A lista é então restringida para conter apenas a porcentagem α dos vizinhos mais próximos, da qual é escolhida uma cidade aleatoriamente. A Fig. 2 ilustra o pseudo-código para a construção da solução.

```
1. S = ∅
2. Escolhe a primeira cidade e insere na solução S = S∪{0}
3. Para i = 1 até n
     a) Cria uma lista ordenada pela distância dos
        vizinhos da cidade i
     b) Restringe a lista para com os α vizinhos mais
        próximos e não visitados
     c) Escolhe um candidato s aleatório da lista
        restrita
     d) S = S ∪ {s}
4. Retorna S
```

Figura 2. Pseudo-código para a construção da solução semi-gulosa aleatória.

Com $\alpha = 0$ a busca é estritamente gulosa, pois, apenas o vizinho mais próximo e não visitado de uma cidade é escolhido. Isso gera uma convergência rápida, porém com uma diversidade menor entre as soluções candidatas. Utilizando $\alpha = 1$ a busca é totalmente aleatória, sendo escolhido qualquer um dos vizinhos. Dessa maneira a a convergência se torna mais lenta, porém com uma grande diversidade. Neste trabalho foi escolhido $\alpha = 0.15$.

4. Simulated Annealing (SA)

O *Simulated Annealing* (SA) é uma metaheurística de otimização baseado em busca local. Uma das principais características do SA é a possibilidade de aceitar uma solução pior que a solução atual, o que faz com que o SA não fique preso em mínimos locais.

O algoritmo foi inspirado em um fenômeno real conhecido como recozimento (*Annealing*), que é o processo de aquecimento de metais seguido de resfriamento lento e gradual, cujo objetivo é tornar o material mais rígido. Dessa maneira, o SA explora a analogia entre o modo como um metal se resfria e se congela numa estrutura cristalina de energia mínima e a busca por um mínimo num sistema qualquer [Kirkpatrick et al. 1983].

4.1. Descrição do Algoritmo

Para que seja possível resolver um problema utilizando o SA, devem ser definidas duas funções principais:

1. **Função de energia:** também conhecida como função de objetivo, avalia as soluções do problema indicando o quanto elas o satisfazem.
2. **Função de incremento:** retorna uma solução vizinha a partir de uma dada solução, permitindo explorar a vizinhança de uma solução candidata.

Além dessas funções, existem quatro parâmetros fundamentais para o sucesso do algoritmo. São elas: (a) Temperatura inicial, (b) Temperatura final, (c) Decremento da temperatura e (d) Número de iterações em cada temperatura.

A temperatura inicial deve ser quente o suficiente para permitir a aceitação de quase todas as soluções de uma dada vizinhança. Caso isso não aconteça, a solução final pode ser a própria solução inicial, ou muito próxima desta e, nesse caso, a implementação seria praticamente idêntica a do *Hill Climbing*. Entretanto, se a temperatura for muito elevada, todo e qualquer vizinho será aceito e, portanto, a busca será praticamente aleatória.

A temperatura final é utilizada como parâmetro de controle e estabelece quando o algoritmo deverá ser encerrado. Normalmente, a temperatura é reduzida até que atinja o valor zero. No entanto, isto pode fazer com que o algoritmo seja executado por um tempo muito maior que o necessário, dependendo do método de decremento de temperatura utilizado.

O decremento da temperatura controla o modo como será feito o resfriamento do sistema e possibilita chegar na temperatura final (critério de parada) a partir da temperatura inicial. O número de iterações que deve ser realizado em cada temperatura deve ser suficiente para que o sistema estabilize. Nas temperaturas baixas, são necessárias muitas iterações para que o mínimo onde o SA se encontra (seja local ou global) seja totalmente explorado. Já nas temperaturas mais altas, podem se realizar um número menor de iterações, já que a vizinhança a ser explorada é muito grande.

O método proposto por Lundy [Lundy and Mees 1986] é fazer com que a temperatura caia rapidamente nas primeiras iterações do algoritmo, e depois comece a diminuir lentamente. O método de resfriamento é definido como:

$$T_f = \frac{T}{1 + T \times r}$$

onde, T é a temperatura atual, r o fator de resfriamento, e T_f a temperatura final. A Fig. 3 apresenta o comportamento de diminuição da temperatura.

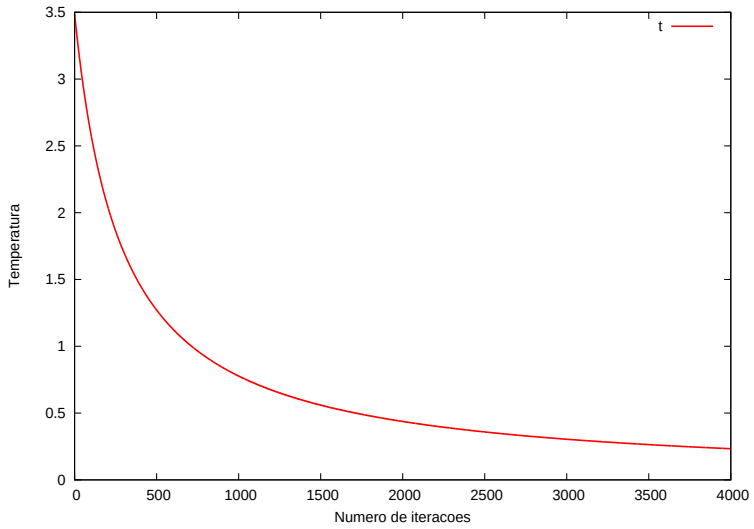

Figura 3. Diminuição da temperatura com o passar do tempo.

A Fig. 4 ilustra o bloco principal do GRASP em pseudo-código [Kirkpatrick et al. 1983].

1. Define um candidato inicial x
2. $T = Temperatura_{inicial}$
3. Repita até alcançar a temperatura mínima
    a) Escolhe aleatoriamente um vizinho v do candidato x
    b) $\Delta E = f(v) - f(x)$
    c) Se $\Delta E <= 0$
        (i) $x = v$
    d) Senão $x = v$ com probabilidade $e^{\frac{-\Delta E}{T}}$
    e) Decrementa a temperatura T

Figura 4. Pseudo-código do Simulated Annealing.

4.2. Parâmetros do Simulated Annealing

Uma vez estabelecida a estrutura de vizinhança, como função de incremento foram utilizadas três abordagens diferentes:

a) **Inversão**: é escolhido um segmento de quatro cidades quaisquer, e inverte a ordem de visita dessas cidades.

$$[\,0\,1\,2\,\underline{\mathbf{3\,4\,5\,6}}\,7\,8\,9\,]$$

$$[\,0\,1\,2\,\underline{\mathbf{6\,5\,4\,3}}\,7\,8\,9\,]$$

b) **Translação**: é escolhido um par de cidades vizinhas que são inserida entre outras duas cidades quaisquer.

$$[\,0\,1\,2\,\underline{\mathbf{3}\,\mathbf{4}}\,5\,6\,7\,8\,9\,]$$

$$[\,0\,1\,2\,5\,6\,7\,8\,\underline{\mathbf{3}\,\mathbf{4}}\,9\,]$$

c) **Permutação**: duas cidades aleatórias são trocadas de lugar.

$$[\,0\,1\,\underline{\mathbf{2}}\,3\,4\,5\,6\,\underline{\mathbf{7}}\,8\,9\,]$$

$$[\,0\,1\,\underline{\mathbf{7}}\,3\,4\,5\,6\,\underline{\mathbf{2}}\,8\,9\,]$$

As duas primeiras abordagens permitem uma busca global na vizinhança da solução, garantindo uma maior diversidade e fuga de mínimos locais. Na segunda abordagem a busca é local e garante que a solução atual seja refinada. A cada chamada da função de incremento uma das três abordagens é escolhida aleatoriamente, com 25% de probabilidade de escolher a primeira, 25% a segunda, e 50% de escolher a terceira.

Os parâmetros do SA foram determinados empiricamente. Foi estabelecido um conjunto de testes nos quais os parâmetros foram ajustados, de modo que o algoritmo se comportasse corretamente. Os valores que foram definidos são:

- Temperatura Inicial: 3.5
- Temperatura Mínima: 0.0001
- Resfriamento: 0.001
- Iterações em cada temperatura: 900

Como critérios de parada, duas condições são verificadas: (a) é atingida a temperatura mínima, ou (b) foram executadas 4000 iterações sem melhorar a melhor solução encontrada.

5. Busca Tabu

A Busca Tabu (*Tabu Search* - TS) é uma metaheurística baseada em busca local [Glover and Laguna 1993]. A Busca Tabu foi utilizada no procedimento de busca local do GRASP, como proposto por [Laguna and González-Velarde 1991]. Esta abordagem não foi o foco principal deste trabalho, por esse motivo maiores detalhes foram omitidos.

A principal característica desta metaheurística é a existência de uma lista tabu, que armazena as soluções já foram visitas em um passado recente. Dessa maneira, as soluções presentes nesta lista não são consideradas em novas buscas. Existem dois parâmetros principais: (a) Tamanho da lista tabu, e (b) número de iterações congeladas. Os valores definidos empiricamente foram n (onde n é o número de cidades) e 9000, respectivamente.

6. Resultados Experimentais

O problema do caixeiro viajante assimétrico foi escolhido como base para os experimentos. As instâncias utilizadas fazem parte da TSPLIB, que é um *benchmark* para validação de algoritmos que visam resolver o TSP. Ao todo foram utilizados 19 conjuntos de testes, com diferentes números de cidades.

Para cada uma destes problemas os algoritmos estudados foram executados 25 vezes. Destas execuções 3 foram realizadas em um ambiente controlado para análise do tempo computacional e as demais foram realizadas em um cluster.

Na Tabela 1 são apresentados os resultados obtidos. A primeira coluna é o nome do arquivo da TSPLIB utilizado, seguido do número de cidades contidas nesta instâncias arquivo e da melhor solução já encontrada. A quarta coluna apresenta a melhor solução encontrada com o GRASP + SA, e na quinta coluna está o desvio padrão, σ_1 das soluções encontradas, considerando as 25 execuções. Na sexta coluna está a melhor solução encontrada para a abordagem GRASP + TS, seguido do desvio padrão σ_2. O desvio padrão mede o grau de dispersão entre os dados, desta maneira, quanto menor é o desvio padrão, maior é a estabilidade do algoritmo em sempre convergir para a mesma solução.

Arquivo	**Cidades**	**Ref.**	**GRASP+SA**	σ_1	**GRASP+TS**	σ_2
br17	17	39	39	0	39	0
ftv33	33	1286	1286	15.8997	1385	33.5384
ftv35	35	1473	1475	5.93107	1522	12.4979
ftv38	38	1530	1530	8.61162	1605	22.4161
p43	43	5620	5620	0	5623	0.72
ftv44	44	1613	1615	16.9723	1704	17.9571
ftv47	47	1776	1824	20.0856	1948	39.1435
ry48p	48	14422	14718	153.504	14706	60.6789
ft53	53	6905	8097	211.06	7962	223.696
ftv55	55	1608	1649	23.5285	1787	34.8343
ftv64	64	1839	1897	32.1114	2041	46.1185
ftv70	70	1950	2038	28.7745	2118	42.8929
ft70	70	38673	41387	463.992	40596	258.493
kro124p	124	36230	40990	912.398	42946	906.376
ftv170	170	2755	3319	70.7575	3556	116.869
rbg323	323	1326	1361	5.73634	1466	21.1991
rbg358	358	1163	1188	5.32165	2465	246.396
rbg403	403	2465	2465	0	2475	4.19027
rbg443	443	2720	2720	1.09982	2816	37.3518

Tabela 1. Resultados obtidos nos experimentos, com as abordagens estudadas: GRASP + SA e GRASP + TS.

Analisando os resultados obtidos com a abordagem GRASP + SA é possível observar que o algoritmo apresenta um bom comportamento na resolução do problema. Dos 19 problemas avaliados, em 11 deles a solução ótima foi alcançada. Observando o desvio padrão, σ_1, é possível observar que o algoritmo consegue encontrar uma solução próxima da ótima. Mesmo assim, com exceção dos casos kro124p e ft70, que apresentam um alto desvio padrão, as soluções encontradas foram próximas da solução ótima global.

Com o aumento da complexidade dos problemas, houve um aumento da degradação dos resultados. Entretanto, nas instâncias rbg403 e rbg443, essa degradação não aconteceu. Nestes dois casos ocorreu uma convergência para o mínimo global, como o exemplo rbg403, no qual a solução ótima foi encontrada em todas as 25 execuções

($\sigma_1 = 0$). Por este motivo, este problema apresentou um tempo médio inferior ao exemplo rbg358, que possui um número menor de cidades (ver Tabela 2).

Uma das razões pelas deste comportamento, e por que a solução ótima não foi sempre encontrada é devido à solução gulosa aleatória. Quando é atribuído um valor muito baixo para α a solução torna-se gulosa, direcionando a busca local para um mínimo local. Foi observado com o caso ftv35, que se utilizarmos uma busca gulosa ($\alpha = 0$), o SA não consegue alcançar o mínimo global. Entretanto, aumentando a diversidade na construção da solução ($\alpha = 0.15$), o SA passa a encontrar o mínimo global.

Os resultados obtidos com o GRASP + TS, não apresentaram um desempenho superior. Como pode ser observado na Tabela 1, os resultados foram muito próximos da abordagem GRASP + SA, sendo na maioria dos casos um pouco menos preciso. Além disso, esta abordagem apresentou um tempo computacional muito maior (ver Tabela 2).

A Tabela 2 apresenta a média do tempo execução dos dois algoritmos estudados. É possível observar, que o tempo aumenta consideravelmente a medida que a complexidade dos problemas é incrementada.

Arquivo	**Cidades**	**GRASP+SA**	**GRASP+TS**
br17	17	15s	2s
ftv33	33	25s	19s
ftv35	35	26s	24s
ftv38	38	33s	31s
p43	43	40s	26s
ftv44	44	34s	52s
ftv47	47	32s	58s
ry48p	48	34s	1m 49s
ft53	53	36s	1m 51s
ftv55	55	37s	1m 49s
ftv64	64	37s	2m 25s
ftv70	70	47s	2m 52s
ft70	70	42s	3m 29s
kro124p	124	54s	7m 21s
ftv170	170	2m 28s	35m 51s
rbg323	323	11m 45s	6h
rbg358	358	14m 09s	9h 05m
rbg403	403	9m 04s	12h 10m
rbg443	443	16m 48s	12h 38m

Tabela 2. Comparação com relação ao custo computacional.

Analisando os resultados da Tabela 2, podemos observar que o SA tem um desempenho maior que a Busca Tabu em relação ao tempo de execução. Para problemas com um número pequeno de cidades, a Busca Tabu é melhor que o SA, mas a medida que a complexidade aumenta o SA passa mantém um tempo consideravelmente inferior. Essa grande diferença ocorre por que a Busca Tabu precisa verificar e atualizar a sua lista tabu com freqüência. Por este motivo, quanto maior a lista, maior é o tempo de execução.

7. Conclusão

Neste trabalho foi apresentado a combinação de duas metaheurísticas, o GRASP e SA, para a resolução do problema do caixeiro viajante. O algoritmo apresentou um bom comportamento nos experimentos realizados utilizando os problemas da TSPLIB. Além disso, dada a complexidade do problema, o foi possível encontrar uma solução exata ou próxima ao mínimo global com um baixo tempo computacional.

Algumas melhorias podem ser adicionadas ao algoritmo proposto. Uma delas é a manipulação do valor α, utilizado na criação da solução semi-gulosa aleatória do GRASP, para se adaptar ao problema em questão. Outra melhoria possível é utilização do *Adaptive Simulated Annealing* (ASA), no qual os parâmetros são ajustados automaticamente.

Referências

Black, P. E. (2005). traveling salesman. Dictionary of Algorithms and Data Structures [online], Available from: http://www.nist.gov/dads/HTML/travelingSalesman.html.

Dantzig, G. B., Fulkerson, R., and Johnson, S. M. (1954). Solution of a large-scale traveling salesman problem. *Operations Research*, (2):393–410.

Glover, F. and Laguna, M. (1993). Tabu search. *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–141.

Gutin, G. and Punnen, A. (2002). *Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Laguna, M. and González-Velarde, J. (1991). A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing*, 2:253–260.

LEE, S. H. (2006). Greedy randomized adaptive search procedure for traveling salesman problem. Master's thesis, Texas A&M University. http://handle.tamu.edu/1969.1/3735.

Lundy, M. and Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming: Series A and B*, 34(1):111–124.

Pitsoulis, L. and Resende, M. (2002). Greedy randomized adaptive search procedures. In Pardalos, P. and Resende, M., editors, *Handbook of Applied Optimization*, pages 178–183. Oxford University Press.

Resende, M. (2001). Greedy randomized adaptive search procedures (GRASP). In Floudas, C. and Pardalos, P., editors, *Encyclopedia of Optimization*, volume 2, pages 373–382. Kluwer Academic Publishers.

Resende, M. and Ribeiro, C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers.

Silveira, C. (1999). GRASP – Uma heuristíca para resolução de problemas de otimização combinatoria. Technical report, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil.