Universidade Federal de São Paulo
Instituto de Ciência e Tecnologia
Projeto e Análise de Algoritmos

Problema do Caixeiro Viajante

Alunos: André Vitor Leiniö
Davi Melo Morales
Lucas Santana Lellis

Dezembro
2015

1 Introdução

O problema do caixeiro viajante consiste em um problema NP-Completo de otimização combinacional [1], que possui grande importância em pesquisa operacional e em ciência da computação teórica. Seu enunciado é o seguinte:

> "Dada uma lista de cidades e as distâncias entre elas, qual seria a menor rota possível a qual o caixeiro viajante visitasse cada uma delas exatamente uma vez e ao final retornasse à cidade de origem?"

Sua primeira formulação foi feita em 1930 e hoje é um dos problemas mais estudados na área de otimização, sendo utilizado como *benchmark* para uma série de métodos. Suas aplicações práticas podem abranger áreas como planejamento e logística.

Há diversas possíveis soluções para o problema, tanto de maneira exata - tais como força bruta e *backtracking* - quanto de maneira aproximada, por meio de heurísticas, como o *Simulated Annealing*.

2 Objetivos

Implementar soluções para o problema do caixeiro viajante e realizar comparações entre a de forma aproximada e a de forma exata, levando em consideração os tempos de execução e a qualidade das soluções aproximadas.

3 Métodos

Para tal, foram feitas três abordagens para a resolução do problema: a solução por força bruta, por backtracking e pela heurística *Simulated Annealing*. Para os testes dos algoritimos utilizou-se uma máquina com as especificações descritas na Tabela 3, com as instâncias listadas na Tabela 3.

CPU	Intel i7 990X
Cores	6
Threads	12
Clock	3.47 GHz
Cache	12 MB
RAM	20 GB
SO	Ubuntu 14.04

Tabela 1: Especificações da Máquina

Nome	**Tamanho**	**Sol. Exata**
Berlim52	52	7,542
Pr76	76	108,159
Ch150	150	6,528

Tabela 2: Instâncias utilizadas[1]

Testes foram feitos utulizando o algoritimo *Simulated Annealing* variando o parametro α entre o valores 0.85, 0.90 e 0.99 e executando o algoritimo duas vezes para cada entrada e anotado o tempo de execução e o resultado.

Testou-se também uma instância de tamanho menor a fim de comparar os tempos de resolução dos algoritmos, assim como a qualidade das soluções obtidas; essa instância apresenta os 13 primeiros valores de Berlim52 e foi chamada de *littleboy*. A Tabela 3 apresenta as especificações da máquina usada para tais testes.

[1]Instâncias e soluções exatas obtidas no endereço `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

CPU	Intel® Core™2 Duo CPU E7400
Cores	2
Threads	2
Clock	2.80 GHz
RAM	4 GB
SO	Fedora 22

Tabela 3: Especificações da Máquina 2

3.1 Considerações

A entrada do problema consiste em um arquivo que possui um número sequencial de cidades em que cada linha apresenta uma identificação e as coordenadas nos eixos x e y de cada cidade.

A representação do problema se deu através de um vetor, no qual as cidades foram identificadas através de uma *id* própria, representada pela Figura 1.

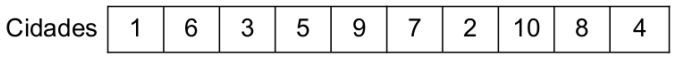

Figura 1: Exemplo de representação do problema.

Um pré-processamento é realizado de modo que calculam-se as distâncias euclideanas entre todas as cidades e estes resultados são armazenados em uma tabela para consulta.

3.2 Força Bruta

A abordagem por força bruta considera todas as possíveis permutações entre as cidades, calculando a cada iteração o valor da rota presente; comparam-se então os valores para que se encontre o melhor (menor).

Levando-se em conta que nesta implementação é necessário considerar cada permutação entre as n cidades de entrada, serão feitas comparações entre as $n!$ possíveis soluções e para cada permutação os n elementos são percorridos para somar as distâncias. Portanto, conclui-se que este algoritmo é da ordem $O(n.n!)$.

3.3 *Backtracking*

A solução por backtracking implementada foi uma adaptação de um algoritimo de permutações, porém a cada iteração o valor da distância é somado a uma variável usada para fazer a poda caso seja maior que a distância da rota mínima calculada até o momento. Como o algoritimo de backtracking tem a ordem de *O(n!)*, esta é a ordem do nosso algoritimo no pior caso, porém com a poda o caso médio apresenta melhora razoável.

3.4 *Simulated Annealing*

O *Simulated Annealing*, ou recozimento simulado, foi a meta-heurística utilizada na abordagem aproximada.

Proposto por Scoot Kirkpatrick, ele realiza o processo de busca por soluções ao simular o processo de recozimento de metais. Neste processo, o sólido é aquecido além de seu ponto de fusão e resfriado. Um resfriamento rápido conduz a produtos meta-estáveis, de maior energia interna, ao passo que um resfriamento lento conduz a produtos mais estáveis e de menor energia. Assim, o processo passa por vários estados possíveis ao longo do recozimento.

A analogia com um problema combinatório se dá pela Tabela 4.

Estados possíveis	**Soluções** do espaço de busca
Energia	**Função objetivo**
Energia mínima	**Solução ótima** local, possivelmente local

Tabela 4: Analogia com problema combinatório.

A cada iteração, gera-se um novo estado a partir do estado corrente por uma pequena modificação aleatória; a troca de posição entre duas cidades, no caso presente. Caso esse novo estado apresente melhora em relação ao anterior, ele se torna o estado corrente. Em caso de piora, a probabilidade de se mudar do estado corrente para um novo estado é de $e^{-\Delta/(T)}$, onde Δ é a diferença entre as soluções e T é o tempo corrente, o qual fornece proporcionalidade em relação à etapa do processo de recozimento [2].

Em altas temperaturas, a chance de se aceitar um estado de piora é alta, de modo que cada solução possui praticamente a mesma probabilidade de ser a solução corrente. Ao longo do resfriamento, esta chance diminui, e a baixas temperaturas somente soluções com baixos valores terão alta probabilidade de se tornarem a solução corrente.

Esta técnica permite que mínimos locais sejam evitados, o que garante uma solução melhor.

O comportamento do SA é descrito através de uma série de parâmetros variáveis. São eles:

- **Temperatura inicial (T_0)**: Representa a temperatura inicial, a qual deve ser alta o suficiente para permitir movimentos livres entre soluções vizinhas.
- **Taxa de resfriamento (α)**: é a taxa de resfriamento, responsável pela velocidade na qual a temperatura resfria.
- **SAmax**: número de interações a cada nível de temperatura.
- **Tamanho do vetor (n)**: número de cidades envolvidas no problema.
- **Temperatura final T_f**: uma temperatura mínima, próxima a zero, que indica a condição de parada.

O tempo de execução do algoritmo dependerá diretamente destes quatro parâmetros, uma vez que para uma temperatura inicial, haverá um decrescimento desta temperatura até a temperatura final; a velocidade deste decrescimento é determinada de modo que a cada iteração, a temperatura corrente é multiplicada pela taxa de resfriamento. Assim, este laço externo será executado $log_\alpha T_f/T_0$ vezes.

Para cada temperatura T há SAmax iterações, dentro das quais calcula-se o valor da função objetivo (distância total percorrida) n vezes.

Desta maneira, temos um algoritmo na ordem de O(pqn), onde p = $log_\alpha T_f/T_0$ e q = SAmax.

4 Resultados e Discussão

Foram obtidos resultados apenas para o algoritimo *Simulated Annealing*, não foi possivel obter resultados em tempo hábil para os algoritimos de backtracking e força bruta. Desta forma, os resultados obtidos estão na Tabela 5.

Input	**Alpha**	**Solução**	**Tempo**
Berlin52	99	13,620.6	749.9325
Berlin52	99	14,404.7	885.378
Berlin52	90	23,307.3	66.7996
Berlin52	90	26,611.4	67.2125
Berlin52	85	23,303.1	43.8778
Berlin52	85	26,525.0	43.4335
Pr76	99	282,312	802.966
Pr76	99	321,372	803.845
Pr76	90	440,157	77.211
Pr76	90	460,295	76.9392
Pr76	85	470,609	49.345
Pr76	85	497,850	49.7831
Ch150	99	29,289.8	1,270.89
Ch150	99	29,641.3	1,106.22
Ch150	90	47,363.3	108.695
Ch150	90	46,897.4	105.163
Ch150	85	48,828	70.6719
Ch150	85	48,609.6	70.3704

Tabela 5: Resultados

Calculando as médias para cada variação dos algoritimos obtemos os resultados apresentados pela Tabela 6.

As Figuras de 2 a 7 ilustram os dados da Tabela 6:

Input	Alpha	Sol. Exata	Solução	Random	Tempo(s)
Berlin52	99	7,542	14,012.65	29,948.3	802.6553
Berlin52	90	7,542	24,959.35	29,948.3	67.0061
Berlin52	85	7,542	24,914.05	29,948.3	43.6557
Pr76	99	108,159	301,842	574,570	803.272
Pr76	90	108,159	450,226	574,570	77.075
Pr76	85	108,159	484,230	574,5'70	49.564
Ch150	99	6,528	29,465.6	53,864	1,188.555
Ch150	90	6,528	47,130.35	53,864	106.929
Ch150	85	6,528	48,718	53,864	70.5212

Tabela 6: Média dos Resultados

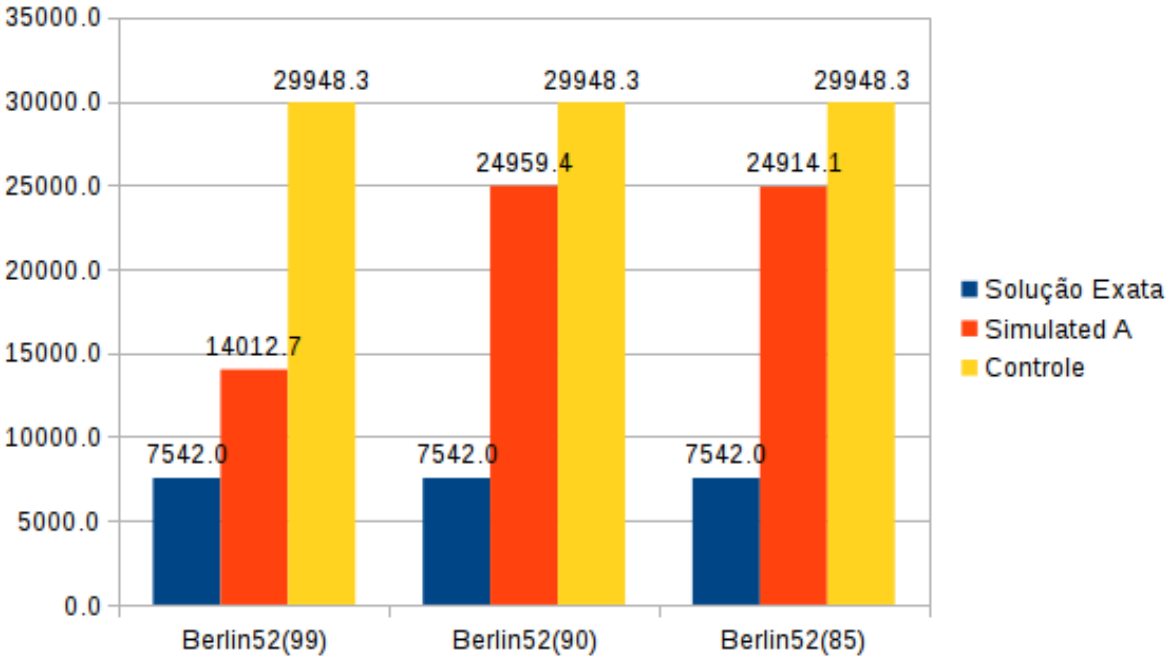

Figura 2: Gráfico comparativo de resultados - Berlin56.

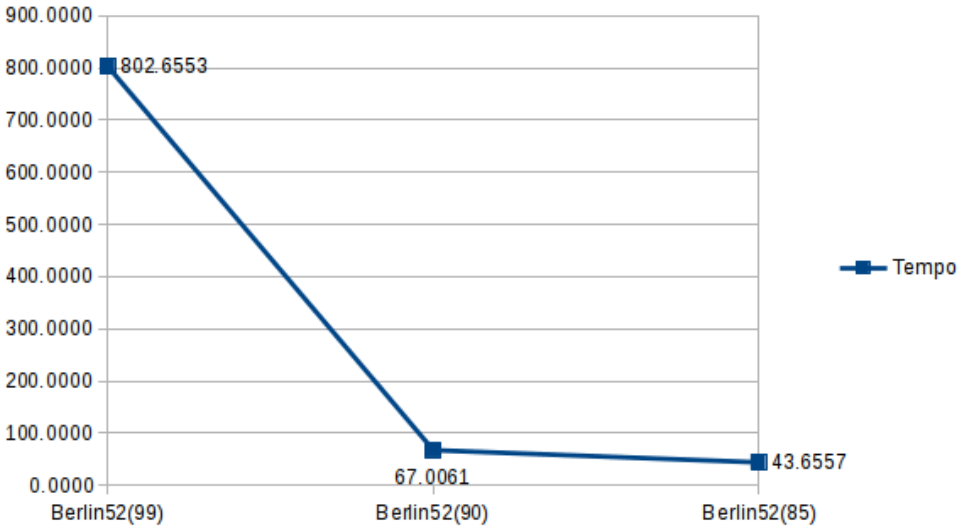

Figura 3: Gráfico comparativo de tempo - Berlin56.

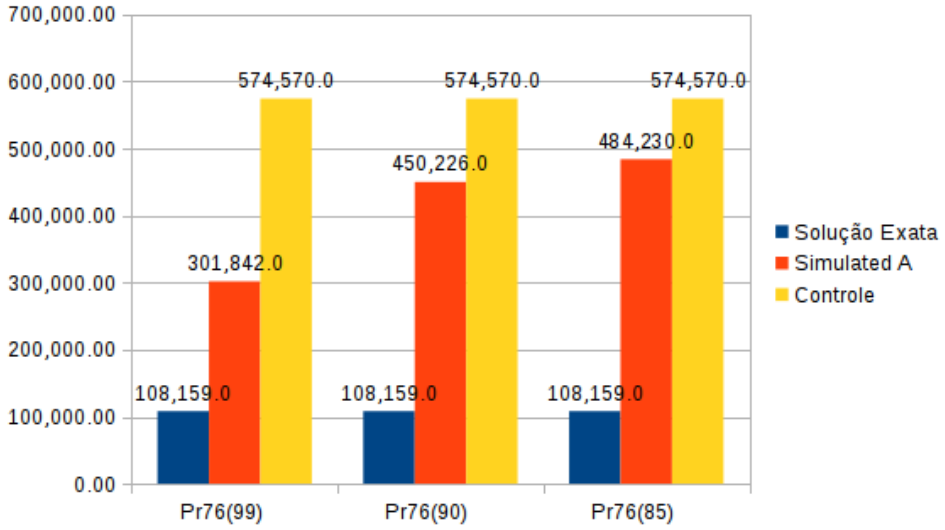

Figura 4: Gráfico comparativo de resultados - Pr76.

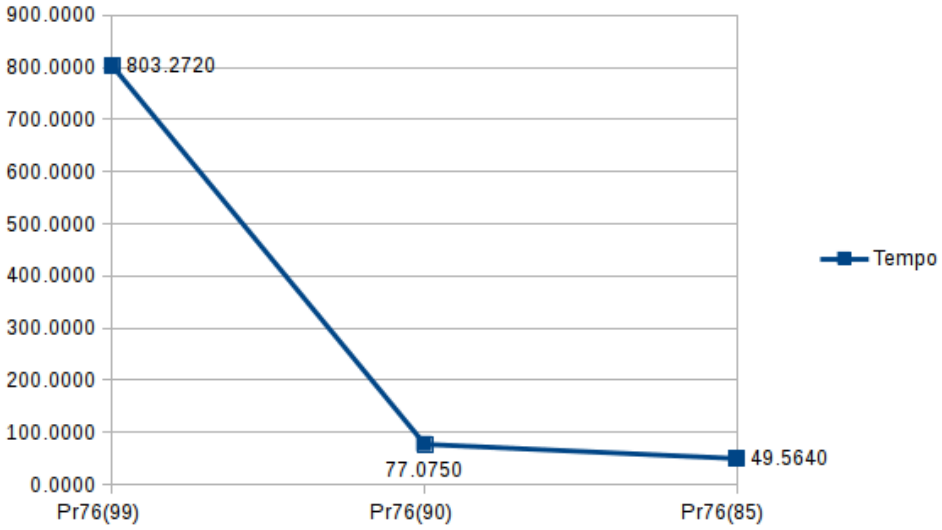

Figura 5: Gráfico comparativo de tempo - Pr76.

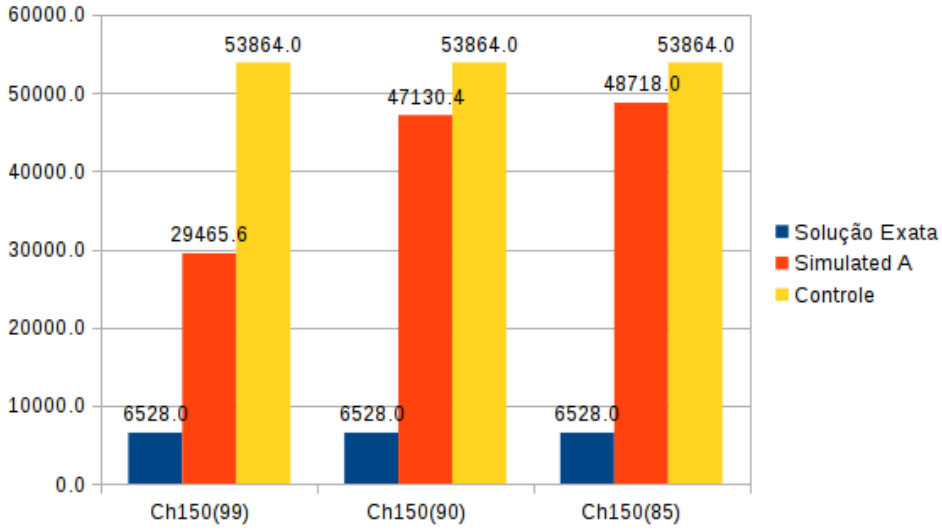

Figura 6: Gráfico comparativo de resultados - Ch150.

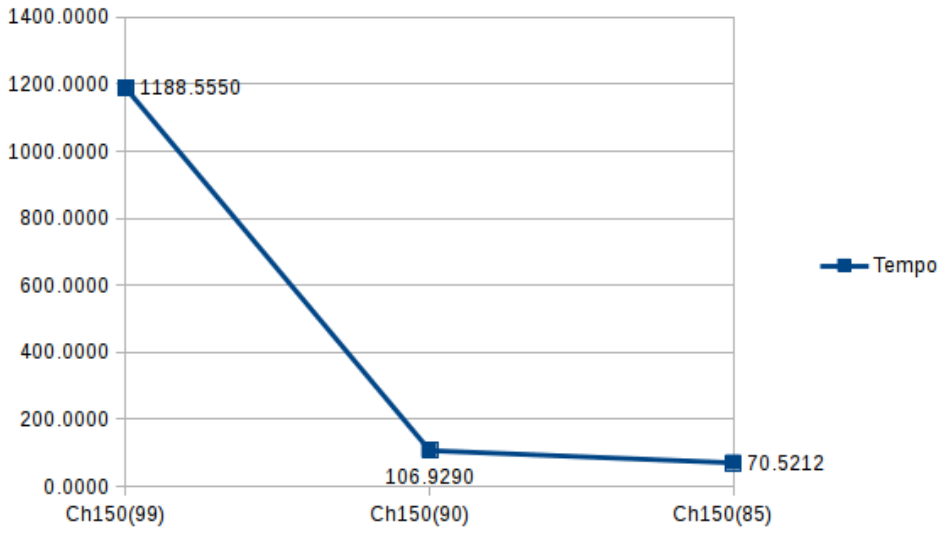

Figura 7: Gráfico comparativo de tempo - Ch150.

Conclui-se a partir dos dados analisados que a solução por *simulated annealing*, em termos de qualidade de solução, sempre apresenta uma melhora em relação a valores aleatórios, porém piora significativa em relação à solução ótima. Por conta da manutenção dos dos parâmetros do *simulated annealing* ao longo das execuções, instâncias menores apresentaram soluções de qualidade melhor que instâncias maiores.

A partir da análise dos dados referentes a execuções com alterações no parâmetro α, percebe-se que para $\alpha = 0,99$, os tempos de execução são bastante elevados, ao passo que para os demais valores testados os tempos de execução são parecidos.

A qualidade das soluções encontradas acompanha, também, este padrão; portanto há soluções de qualidade razoavelmente melhor para $\alpha = 0,99$ em relação às demais soluções, as quais apresentam qualidade parecida. Observa-se, assim, a grande influência desse parâmetro no tempo de execução e qualidade das soluções.

4.1 Comparação com instâncias menores

Como não foi possivel fazer os testes dos algoritimos exatos utilizando as entradas da literatura, foram feitos testes utilizando uma entrada menor. Na Tabela 7 apresentam-se as melhores soluções encontradas por nossos algoritmos exatos, a qual será utilizada como referência para a verificação da qualidade das soluções aproximadas, assim como a solução média gerada aleatoriamente:

Melhor valor encontrado	4564.46
Melhor rota	1 6 2 7 8 9 10 12 11 3 5 4 0 1
Média das soluções aleatórias	11502.8

Tabela 7: Informações a respeito de soluções exatas para a instância *littleboy*

Soluções foram calculadas a partir da instância para os métodos aqui apresentados, o que permitiu verificar o tempo gasto pelos algoritmos. A Tabela 8 apresenta uma comparação entre os resultados obtidos, onde o parâmetro α da solução aproximada foi colocado como *0,99*:

Input	**Abordagem**	**Solução**	**Random**	**Tempo(s)**
littleboy	naive	4564.46	11502.8	3349.47
littleboy	naive	4564.46	11502.8	3360.01
littleboy	backtracking	4564.46	11502.8	22.4283
littleboy	backtracking	4564.46	11502.8	22.5526
littleboy	simulated annealing	4660.45	11502.8	726.597
littleboy	simulated annealing	4614.02	11502.8	726.275

Tabela 8: Resultados obtidos para *littleboy*

A partir dos dados apresentados anteriormente é possível perceber que, para esta instância, a solução apresentada pelo *simulated annealing* é bastante satisfatória; a partir de um valor médio dessas soluções, é possível perceber uma diferença de apenas 1,6% em relação à solução exata, ao passo que a média de soluções aleatórias apresenta uma diferença de 152%. Seu tempo de execução, porém, se mostrou maior que o tempo de execução da solução por *backtracking*, mas ainda razoavelmente menor que para a abordagem *naive*.

A fim de se verificar a melhora de tempo das abordagens *backtracking* e *simulated annealing*, calculou-se o *speedup* da média destas em relação à abordagem *naive*, onde $speedup = t_{naive}/t_{bt,sa}$. Os resultados obtidos apresentam-se na Tabela 9 e na Figura 8:

abordagem	**tempo médio(s)**	**speedup**
naive	3354,74	1x
backtracking	22,49045	149x
simulated annealing	726,436	4,6x

Tabela 9: Dados a respeito do tempo de execução

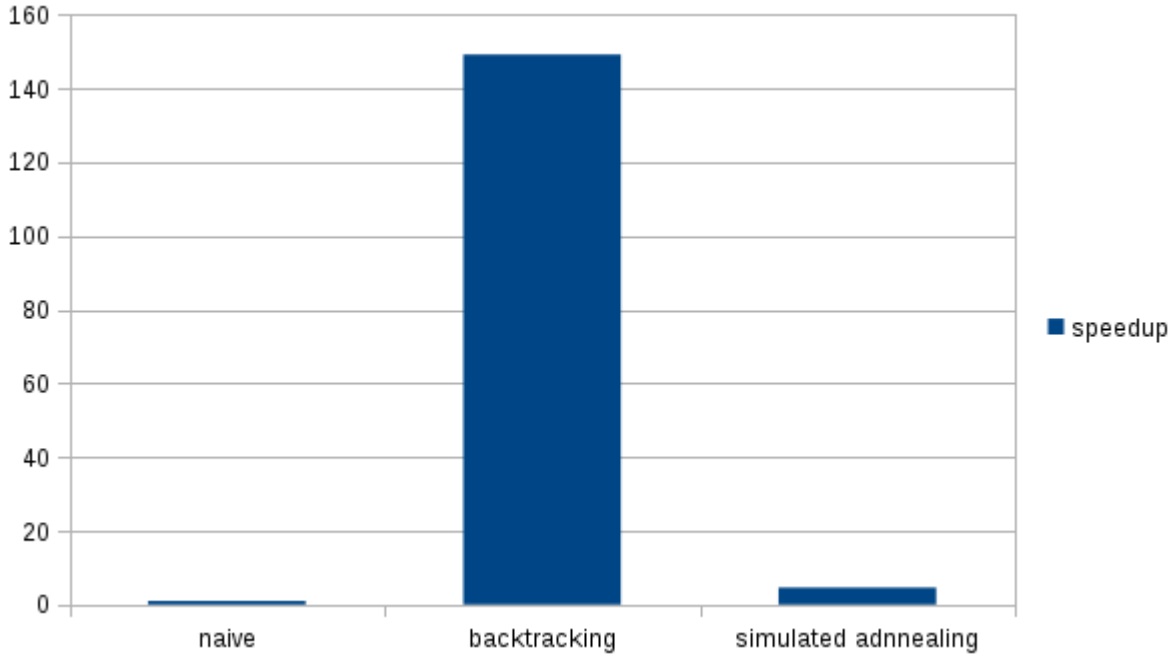

Figura 8: Gráfico comparativo de tempos de execução - *littleboy*

Desta maneira, conclui-se que para instâncias de tamanho pequeno a solução por *backtracking* é mais eficiente que a por *simulated annealing*; conclui-se, também, que a solução aproximada apresenta qualidade muito boa e, portanto, mostra-se como uma boa alternativa à solução *naive*.

Propõe-se, para trabalhos futuros, testes similares para o *simulated annealing* com valores diferentes de α, a fim de se verificar relações diferentes de custo-benefício quanto ao tempo de execução e a qualidade das soluções. A análise de tal valor, porém, estaria sujeita ao propósito das soluções em um problema do mundo real.

5 Conclusão

A partir dos testes realizados foi possível observar que as soluções exatas apresentam um tempo de execução bastante elevado, o que as torna impraticáveis para instâncias suficientemente grandes.

O método utilizado para soluções aproximadas apresentou resultados melhores do que a média de soluções aleatórias, mas distantes da solução ótima, especialmente em instâncias numerosas.

Para instâncias menores, porém, as soluções aproximadas apresentaram boa qualidade, mas uma vantagem menos expressiva em relação às soluções exatas.

Observou-se, também, resultados a partir da alteração de parâmetros dentro da solução aproximada.

Para um projeto futuro, seria interessante um refinamento mais criterioso da solução aproximada, de maneira que as soluções se aproximassem mais dos resultados conhecidos da literatura.

Referências

[1] T.H. Cormen. *Introduction to Algorithms*. MIT Press, 2009.

[2] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680.