Otimização Combinatória: implementação e resultados

Luerson de Albuquerque Silva Filho

7 de Março, 2024

1 Introdução

"Otimização Combinatória"corresponde ao texto elaborado por professores e pesquisadores da UFPB especialistas em pesquisa e desenvolvimento de software para a área de otimização, o qual disserta sobre 5 diferentes algoritmos de otimização a serem implementados pelo leitor. Este documento tem como objetivo ilustrar os resultados obtidos pela implementação do autor e compará-las com os resultados médios esperados.

Para obter acesso ao texto "Otimização Combinatória", é necessário entrar em contato com integrantes do ramo Log do laboratório LASER, localizado no Centro de Informática da UFPB. O código fonte do autor pode ser acessado pelo link: https://github.com/Luerson/Kit_Log/tree/main.

2 Procedimentos

2.1 Leitor de instâncias

O código foi implementado na linguagem `C++` e elaborado em cima de uma implementação previamente construída, a qual permite acesso a todas as instâncias utilizadas em "Otimização Combinatória": https://github.com/cvneves/kit-opt/tree/master/GILS-RVND-TSP/leitor-instancias.

2.2 Instâncias

A solução ótima das instâncias utilizadas nos testes podem ser encontradas em: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html

2.3 Resultados

Para avaliar os resultados, obteve-se a média de 10 execuções de cada instância em um Intel® Core™ i5-11400H 2.70GHz antes de compará-la com a média esperada no benchmark. Para facilitar a comparação, foi calculada a diferença relativa dos resultados obtidos com o esperado pelo benchmark no formato:

$$\%\_de\_melhora = \frac{benchmark - obtido}{benchmark} \times 100 \tag{1}$$

Portanto, um valor percentual positivo indica uma melhora no desempenho do algoritmo implementado em relação ao benchmark, enquanto um valor negativo indica piora na mesma proporção. Os valores de tempo são medidos em segundos, o custo em uma unidade arbitrária e ambos foram arredondados para a última casa decimal ilustrada.

3 Heurísticas

3.1 abordagem meta-heurística para o TSP

A meta-heurística consiste basicamente na implementação do método heurístico *Iterated Local Search* (ILS), a qual consiste em um algoritmo de construção seguido de métodos de busca para encontrar a melhor solução local. Essa solução local é então perturbada e novas buscas são feitas até que um marco de parada seja atingido. Após isso, a melhor solução encontrada é retornada.

O autor demorou aproximadamente 20 dias para fazer sua primeira implementação desse método, correspondendo ao tempo de desenvolvimento do código, seguido de otimizações para diminuir o tempo de execução. A comparação dos resultados obtidos com a média esperada em "Otimização combinatória" segue na tabela abaixo:

Table 1: Comparação de Desempenho

Instância	% de Melhora		Benchmark		Implementação	
	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo
a280.tsp	38.6	0.000	96.623	2579.0	59.315	2579.0
ali535.tsp	25.3	-0.020	1525.000	202384.0	1139.699	202423.7
att48.tsp	43.7	0.000	0.300	10628.0	0.169	10628.0
att532.tsp	1.4	-0.006	1778.960	27731.0	1753.600	27732.6
bayg29.tsp	32.6	0.000	0.043	1610.0	0.029	1610.0
bays29.tsp	38.0	0.000	0.050	2020.0	0.031	2020.0
berlin52.tsp	42.5	0.000	0.374	7542.0	0.215	7542.0
bier127.tsp	38.3	0.000	10.209	118282.0	6.303	118282.0
brazil58.tsp	41.8	0.000	0.479	25395.0	0.279	25395.0
brg180.tsp	41.3	0.000	12.824	1950.0	7.522	1950.0
burma14.tsp	50.0	0.000	0.004	3323.0	0.002	3323.0
ch130.tsp	38.4	0.000	10.910	6110.0	6.726	6110.0
ch150.tsp	41.8	0.000	10.430	6528.0	6.075	6528.0
d198.tsp	45.0	0.000	33.639	15780.0	18.511	15780.0
d493.tsp	45.6	-0.009	1132.480	35042.0	615.581	35045.2
dantzig42.tsp	40.4	0.000	0.161	699.0	0.096	699.0
eil101.tsp	41.1	0.000	4.436	629.0	2.614	629.0
eil51.tsp	39.3	0.000	0.369	426.0	0.224	426.0
eil76.tsp	41.3	0.000	1.549	538.0	0.909	538.0
fl417.tsp	43.0	0.000	365.503	11861.0	208.208	11861.0
fri26.tsp	36.4	0.000	0.033	937.0	0.021	937.0
gil262.tsp	39.6	-0.004	82.271	2378.6	49.725	2378.7
gr120.tsp	40.3	0.000	9.065	6942.0	5.416	6942.0
gr137.tsp	41.2	0.000	11.348	69853.0	6.667	69853.0
gr17.tsp	37.5	0.000	0.008	2085.0	0.005	2085.0
gr202.tsp	44.2	0.000	37.105	40160.1	20.702	40160.1
gr21.tsp	35.7	0.000	0.014	2707.0	0.009	2707.0
gr229.tsp	43.8	0.009	61.498	134625.2	34.576	134613.0
gr24.tsp	39.3	0.000	0.028	1272.0	0.017	1272.0
gr431.tsp	42.3	-0.012	721.745	171509.8	416.536	171530.0
gr48.tsp	43.0	0.000	0.314	5046.0	0.179	5046.0
gr96.tsp	43.7	0.000	3.475	55209.0	1.957	55209.0
hk48.tsp	45.2	0.000	0.336	11461.0	0.184	11461.0
kroA100.tsp	40.2	0.000	3.468	21282.0	2.075	21282.0
kroA150.tsp	42.1	0.000	11.751	26524.0	6.804	26524.0
kroA200.tsp	42.1	0.000	32.951	29368.0	19.068	29368.0
kroB100.tsp	38.8	0.000	3.748	22141.0	2.295	22141.0
kroB150.tsp	37.0	0.000	10.634	26130.0	6.697	26130.0
kroB200.tsp	41.2	0.003	35.530	29438.1	20.905	29437.2
kroC100.tsp	41.2	0.000	3.568	20749.0	2.097	20749.0
kroD100.tsp	43.0	0.000	4.114	21294.0	2.345	21294.0
kroE100.tsp	37.7	0.000	3.745	22068.0	2.335	22068.0
lin105.tsp	42.4	0.000	4.355	14379.0	2.510	14379.0
lin318.tsp	43.4	-0.015	188.780	42039.4	106.795	42045.7
linhp318.tsp	44.1	-0.016	187.536	42046.5	104.864	42053.1

Continua na próxima página

Table 1 – continuação

Instância	% de Melhora		Benchmark		Implementação	
	Tempo	Custo	Tempo	Custo	Tempo	Custo
pcb442.tsp	42.4	-0.037	597.431	50857.4	344.380	50876.0
pr107.tsp	36.7	0.000	4.582	44303.0	2.900	44303.0
pr124.tsp	42.6	0.000	7.021	59030.0	4.030	59030.0
pr136.tsp	39.0	0.000	13.632	96772.0	8.318	96772.0
pr144.tsp	39.1	0.000	10.479	58537.0	6.381	58537.0
pr152.tsp	38.6	0.000	8.708	73682.0	5.346	73682.0
pr226.tsp	42.6	0.000	45.270	80369.0	25.973	80369.0
pr264.tsp	41.2	0.000	64.758	49135.0	38.070	49135.0
pr299.tsp	41.9	-0.008	130.098	48191.0	75.622	48194.8
pr76.tsp	42.1	0.000	1.366	108159.0	0.791	108159.0
rat195.tsp	37.8	-0.108	28.046	2323.6	17.454	2326.1
rat99.tsp	40.2	0.000	4.115	1211.0	2.460	1211.0
rd100.tsp	39.2	0.000	3.983	7910.0	2.423	7910.0
rd400.tsp	45.1	0.050	498.288	15303.8	273.361	15296.1
si175.tsp	42.4	0.000	17.333	21407.0	9.987	21407.0
si535.tsp	38.4	-0.014	758.534	48460.0	467.054	48466.8
st70.tsp	38.6	0.000	1.030	675.0	0.632	675.0
swiss42.tsp	40.0	0.000	0.155	1273.0	0.093	1273.0
ts225.tsp	41.0	0.000	28.869	126643.0	17.031	126643.0
tsp225.tsp	38.2	0.000	45.368	3916.0	28.046	3916.0
u159.tsp	38.8	0.000	10.828	42080.0	6.628	42080.0
ulysses16.tsp	50.0	0.000	0.008	6859.0	0.004	6859.0
ulysses22.tsp	36.8	0.000	0.019	7013.0	0.012	7013.0

3.2 Concatenação de subsequências (MLP)

Esse método consiste numa abordagem heurística para o problema da mínima latência (*Minimum Latency Problem*, MLP). Para resolver o problema, a heurística ILS é readaptada para receber uma matriz de subsequências, que permitirá realizar operações da busca local de maneira ágil e sem perda indevida de informação.

O autor implementou a metodologia proposta ao longo de 4 dias de desenvolvimento. A agilidade se deve à reciclagem do código elaborado para a abordagem anterior, apenas readaptado para o MLP. A comparação dos resultados obtidos com a média esperada em "Otimização combinatória" segue na tabela abaixo:

Table 2: Comparação de Desempenho

Instância	% de Melhora		Benchmark		Implementação	
	Tempo (s)	Custo	Tempo (s)	Custo	Tempo (s)	Custo
dantzig42.tsp	62.2	0.000	0.160	12528.0	0.061	12528.0
swiss42.tsp	65.9	0.000	0.160	22327.0	0.055	22327.0
att48.tsp	70.2	0.000	0.320	209320.0	0.095	209320.0
gr48.tsp	67.9	0.000	0.330	102378.0	0.106	102378.0
hk48.tsp	65.1	0.000	0.300	247926.0	0.105	247926.0
eil51.tsp	70.3	-0.186	0.490	10178.0	0.145	10196.9
berlin52.tsp	73.1	0.000	0.460	143721.0	0.124	143721.0
brazil58.tsp	73.8	0.000	0.780	512361.0	0.204	512361.0
st70.tsp	74.7	0.000	1.650	20557.0	0.417	20557.0
eil76.tsp	77.3	-0.034	2.640	17976.0	0.600	17982.1
pr76.tsp	78.3	0.000	2.310	3455242.0	0.501	3455242.0

Continua na próxima página

Table 2 – continuação

Instância	% de Melhora		Benchmark		Implementação	
	Tempo	Custo	Tempo	Custo	Tempo	Custo
gr96.tsp	79.5	0.000	6.190	2097170.0	1.270	2097170.0
rat99.tsp	80.7	0.000	11.270	57986.0	2.176	57986.0
kroA100.tsp	81.4	0.000	8.590	983128.0	1.601	983128.0
kroB100.tsp	81.9	0.000	9.210	986008.0	1.668	986008.0
kroC100.tsp	80.2	0.000	8.170	961324.0	1.620	961324.0
kroD100.tsp	79.0	0.000	8.460	976965.0	1.781	976965.0
kroE100.tsp	75.9	0.000	8.310	971266.0	1.999	971266.0
rd100.tsp	80.4	0.000	8.520	340047.0	1.672	340047.0
eil101.tsp	81.9	-0.021	12.760	27513.0	2.315	27518.7
lin105.tsp	82.6	0.000	8.420	603910.0	1.463	603910.0
pr107.tsp	83.0	0.000	10.890	2026626.0	1.847	2026626.0

Observando os dados, fica evidente que a diferença em tempo de execução obtida pela implementação proposta contra a média oferecida pelo benchmark é ainda mais acentuada do que observado na subseção 3.1. Tendo em vista que a implementação atual é apenas uma readaptação do código ILS anterior para o problema proposto no capítulo, é razoável imaginar que um fator externo esteja influenciando o aumento de desempenho relativo.

Uma hipótese levantada pelo autor, é de que um desbalanço no poder dos processadores das máquinas utilizadas é um fator relevante nessa diferença de desempenho. Na subseção anterior, um computador com processador Intel® Core™ i7-3770 3.40GHz foi utilizado para gerar os dados do benchmark, enquanto o benchmark da subseção atual foi gerado por uma máquina com Intel® Core™ i7 2.93 GHz (demais detalhes não informados), o que pode explicar a aumento relativo de desempenho. Entretanto, não é desconsiderada a possibilidade de que a qualidade da implementação seja a principal causa desse fenômeno, assim como outras potenciais razões.

4 Agradecimentos

Meus agradecimentos vão principalmente para meus colegas Petrus George Leal Ismael da Costa Neves e Guilherme Dantas Pinto, os quais são também integrantes do ramo Log do laboratório LASER e que me deram suporte durante o projeto. Gostaria também de agradecer aos professores Bruno Bruck e Anand Subramanian, que me apresentaram esse primeiro desafio.