Soluções 1

Ambiente de execução das soluções: um PC com processador AMD Ryzen 9 39000X com 12 cores de 3.8 GHz, 32 GB RAM, e Ubuntu Linux. Código: aqui.

Exercício 1 (Vizinhanças, 2 pt)

1-swap. A vizinhança 1-swap é *simétrica*, porque trocando os mesmos vértices uma segunda vez obtemos o estado original. Vamos aproveitar essa observações simples para introduzir um pouco de notação. Para uma partição (P, Q) balanceada (i.e. $||P|-|Q|| \leq 1$) temos $(P, Q) \rightleftharpoons (p, q) \rightleftharpoons (q, p) = (P, Q)$, onde $(P, Q) \rightleftharpoons (p, q)$ para um par de vértices $p \in P, q \in Q$ é $(P \setminus \{p\} \cup \{q\}, Q \setminus \{q\} \cup \{p\})$. Ainda vamos escrever $|(P, Q)| = |P| + |Q|$ e $(P, Q) \oplus (R, S) = (P \oplus R, Q \oplus, S)$.
A vizinhança 1-swap não é *conectada*: as duas partes sempre tem a mesma quantidade $\lceil n/2 \rceil$ e $\lfloor n/2 \rfloor$ de vértices, logo não é possível alcançar um estado onde a primeira parte tem um vértice a menos. Mas essa definição é rígida demais: podemos identificar a partição (P, Q) com (Q, P). Agora a vizinhança é conectada: dado duas partições $S = (P_S, Q_S), T = (P_T, Q_T)$ podemos supor que elas tem o mesmo número de vértices em $|P_S| = |P_T|$ (*). Logo existe um vértice $p \in P_S$, $p \notin P_T$ senão, $P_T \subseteq P_S$ e (*) implica $P_S = P_T$, i.e. as soluções ficam idênticas. Por um argumento similar também existe $q \in Q_S$, $q \notin Q_T$. Trocando p e q em S reduz a distância: $|(S \rightleftharpoons (p, q)) \oplus T| = |S \oplus T| - 2$, e um argumento por indução mostra que existe um sequência de 1-swaps que transforma S em T. Com isso a vizinhança também está *fracamente otimamente conectada*.
Finalmente ela não é *exata*:  é um mínimo local (não estrito) de valor 2, mas a solução ótima  tem valor 1.
2-swap. A vizinhança 2-swap é *simétrica*, pelo mesmo argumento. Ela não é *conectada*, mesmo identificando soluções (P, Q) e (Q, P), porque podemos ter uma distância 2, mas um 2-swap muda a distância sempre por 4. Também não é *fracamente otimamente conectada* (por exemplo, do estado  não podemos alcançar o estado ótimo ) e por isso também não *exata*.

Exercício 2 (Busca local para um problema polinomial, 4pt)

O processo de geração sugerido no exercício tende a produzir envoltórias convexas com poucos pontos: ele foi substituído por um processo que gera pontos (r, φ) com $r = 0.5$ e $\varphi \in U[0, 2\pi]$. Um exemplo é dado na Figura 1 (esquerda).
Vamos comparar buscas local primeira e melhor melhora, PM e MM, com o algoritmo direto D, que ordena os pontos no sentido anti-horário. As instâncias de teste tinham $n = 100[16]$ pontos. Cada experimento foi replicado 30 vezes. O tempo do Algoritmo D foi, em todas instâncias, menos que 1 ms. Tabela 1 mostra para cada n o tempo t em segundos e o número de iterações s, ambos com desvio padrão σ para PM e MM. Figura 1 (meio, direita) mostra gráficos exemplários para PM.
Podemos ver que o tempo em função do número de iterações segue aproximadamente o esperado: cada iteração precisa no caso pessimista analisar $\Theta(n^2)$ vizinhos, para verificar um mínimo local todos vizinhos tem que ser analisados uma vez. Então esperamos um complexidade de $\Omega(n^2)$. Para MM isso é a complexidade por iteração. De fato, a

complexidade empírica, determinada usando uma hipótese polinomial de acordo com a seção 6.5 da notas de aula, produz $T_{\text{PM}}(n) = 3.0 \cdot n^{2.8}$ ns e $T_{\text{MM}}(n) = 2.5 \cdot n^{3.1}$ ns.

Exercício 3 (Busca local para um problema NP-completo, 4pt)

a) O desafio está numa implementação razoavelmente eficiente das buscas. Para acelerar as buscas estamos mantendo um valor g_v para todo vértice $v \in V$, que representa a mudança na função objetivo caso o vértice v passa para o outro parte. Com isso a troca de vértices u, v tem valor $g_u + g_v + 2[uv]$, onde a expressão $[uv]$ é 1 caso existe uma aresta entre u e v, e 0 caso contrário. Uma troca pode ser avaliada em tempo $O(\log \Delta)$, com Δ o maior grau, buscando na lista de vizinhos de vértice u por v para determinar o valor de $[uv]$. Depois de trocar u e v temos que atualizar os valores de u, v e todos vizinhos dos dois vértices em tempo $O(\delta(u) + \delta(v))$.

No caso da melhor melhora isso ainda implica numa complexidade de $\Theta(\Delta n^2)$ para encontrar o melhor par, um tempo proibitivo em grafos maiores. Então neste caso vamos usar *buckets* para acelerar a busca. Nota que $-\Delta \leq g_v \leq \Delta$, para todos vértices. Logo, para uma partição balanceada (P, Q) de V vamos manter p-buckets p_i e q-buckets q_i para $i \in [-\Delta, \Delta]$, onde cada p-bucket p_i contém os vértices em $v \in P$ com $g_v = i$ e cada q-bucket os vértices $v \in Q$ com $g_v = i$. Escreve $|p_i|$ e $|q_i|$ para o número de vértices nos buckets. Para determinar a melhor troca, vamos encontrar o menor i com $|p_i| > 0$ e o menor j com $|q_j| > 0$. Estes índices sempre existem porque os partes são não-vazios. Qualquer par de vértices $u \in p_i$, $v \in q_j$ tem valor $i + j + 2[uv]$. Então caso $[uv] = 0$ encontramos uma troca ótima. Senão ainda é possível encontrar uma troca de menor valor com $u \in p_{i+1}$, $v \in q_j$ ou com $u \in p_i$, $v \in q_{j+1}$, onde $[uv] = 0$. Isso é suficiente para encontrar a melhor troca. Além das atualização dos valores g como no caso da primeira melhores, teremos que atualizar ainda os buckets: cada vértices que troca de valor g troca de bucket.

Tabela 2 mostra para cada instância o melhor valor conhecido, e para as duas buscas o número de iterações(± desvio), o tempo (± desvio) e o desvio relativo (± desvio) do valor encontrado. Podemos observar que os resultados ficam razoáveis (em comparação com soluções aleatórias), mas ainda distantes das melhores

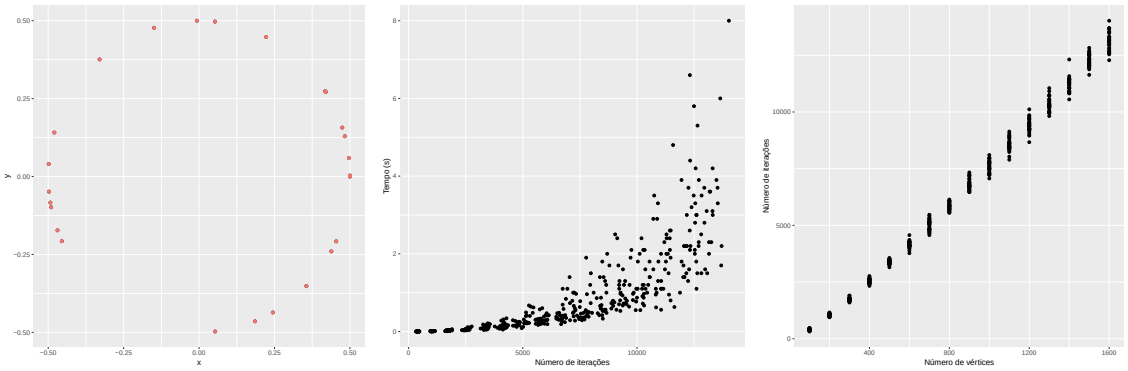

Figura 1: Esquerda: exemplo de uma instância, Meio: tempo em função de número de iterações, Direita: número de iterações em função do número de vértices.

Tabela 1: Resultados das busca locais.

	PM				MM			
n	t (s)	$\pm\sigma$	s	$\pm\sigma$	t (s)	$\pm\sigma$	s	$\pm\sigma$
100	0.00	0.00	402.1	38.8	0.00	0.00	120.9	7.2
200	0.01	0.00	1039.1	53.7	0.03	0.00	262.1	20.0
300	0.02	0.01	1738.2	70.5	0.12	0.01	411.8	25.5
400	0.06	0.03	2547.9	102.6	0.28	0.02	553.4	34.5
500	0.11	0.05	3382.3	93.3	0.58	0.03	724.6	41.2
600	0.16	0.08	4154.4	147.9	1.02	0.08	876.8	58.3
700	0.29	0.16	5028.7	240.9	1.58	0.09	1008.9	51.7
800	0.31	0.11	5843.5	163.6	2.42	0.13	1185.5	63.8
900	0.57	0.32	6818.8	240.4	3.48	0.21	1345.7	77.7
1000	0.60	0.38	7597.5	268.5	4.77	0.26	1488.9	78.1
1100	0.95	0.60	8559.0	280.6	6.51	0.34	1671.6	86.2
1200	1.03	0.44	9399.9	297.0	8.70	0.44	1874.6	91.6
1300	1.44	0.78	10325.8	295.2	10.73	0.69	1978.6	97.3
1400	1.90	1.24	11210.2	331.2	13.50	0.68	2134.1	99.8
1500	2.68	1.27	12280.0	276.8	16.83	0.91	2310.1	120.8
1600	2.86	1.43	13091.6	416.6	20.10	3.21	2413.8	383.7

soluções. O tempos para as maiores instâncias são grandes, mas nunca mais que 10 minutos. Uma curiosidade é que o busca melhor melhora é mais rápida que a primeira melhora, porque implementas uma estrutura de dados com operações eficientes que encontra a melhor melhora sem analisar todas trocas individualmente.

b) A BLMR é uma busca melhor melhora com randomização, então não precisa nenhuma aceleração específica. Porém nos evitamos de selecionar uma das melhores trocas aleatoriamente por razões de eficiência. A tabela 3 mostra os resultados para diferentes valores de p: número de iterações "#i" (em M), desvio relativo r (em %), e tempo t (em s). Podemos observar que o número de iterações diminui de aprox. $1\,M/s$ para $10^5/s$ com p crescente. O desvio relativos para $p = 0$ ficam mais ou menos idênticos com a busca local melhor melhora, como esperado, mas melhoram com p crescente, até aprox. $p = 0.5$ e depois pioram. O melhor desvio relativo médio sobre todas instâncias é observado para $p = 0.5$ com um valor de 1080.8.

c) Nenhuma das buscas encontra os melhores valores conhecidos. Então aqui vamos determinar a complexidade empírica das buscas primeira e melhor melhora em função do tamanho da instância. (O BLMR tem tempo fixo.) Em ambos os casos uma hipótese polinomial é mais adequado (com R^2 de 0.97 e 0.83, respectivamente)

Tabela 2: Resultados das buscas locais.

			FI						BI					
Inst	n	BKV	$\#i$	$\pm\sigma$	r	$\pm\sigma$	t (s)	$\pm\sigma$	$\#i$	$\pm\sigma$	r	$\pm\sigma$	t (s)	$\pm\sigma$
add20	2395	596	1305.5	91.6	93.9	15.6	0.0	0.0	464.0	34.2	57.4	15.3	0.0	0.0
data	2851	189	2105.6	129.3	859.0	94.8	0.0	0.0	612.5	16.3	629.8	67.9	0.0	0.0
3elt	4720	90	2258.0	53.9	2007.1	104.3	0.1	0.0	785.2	26.0	2022.2	93.5	0.0	0.0
uk	4824	19	1924.9	50.0	4939.6	132.5	0.1	0.0	766.9	16.6	4602.1	138.1	0.0	0.0
add32	4960	11	2371.9	93.5	9880.6	636.1	0.1	0.0	875.9	22.1	7298.2	520.8	0.0	0.0
bcsstk33	8738	10171	12338.7	1408.1	157.1	58.1	0.9	0.7	2229.3	70.8	42.0	36.2	0.1	0.0
whitaker3	9800	127	4863.5	130.6	3754.4	95.3	0.2	0.0	1648.7	19.3	3029.0	85.0	0.0	0.0
crack	10240	184	5877.6	140.4	2216.4	51.9	0.3	0.0	1662.7	17.9	2234.4	47.4	0.0	0.0
wingnodal	10937	1707	10687.7	639.3	215.5	65.8	1.2	0.3	2946.5	150.8	278.3	58.1	0.0	0.0
fe4elt2	11143	130	5176.8	147.9	3781.1	120.8	0.3	0.1	1937.5	28.1	3191.6	125.9	0.0	0.0
vibrobox	12328	10343	12402.0	457.0	84.7	13.4	1.3	0.5	3197.5	84.9	39.2	10.9	0.1	0.0
bcsstk29	13992	2843	13230.5	657.9	890.1	114.7	1.0	0.3	3350.7	49.0	438.4	61.0	0.1	0.0
4elt	15606	139	7117.0	120.1	4317.4	101.0	1.3	0.3	2593.0	28.6	4478.8	85.7	0.0	0.0
fesphere	16386	386	6630.7	115.1	1686.4	32.8	1.0	0.3	2687.7	35.2	1699.6	37.2	0.0	0.0
cti	16840	334	8837.5	214.0	2642.0	70.9	0.9	0.3	2840.3	38.8	2386.2	79.2	0.0	0.0
memplus	17758	5499	9538.7	172.0	57.5	3.4	1.0	0.2	3196.8	44.9	64.5	3.0	0.0	0.0
cs4	22499	369	7610.3	348.3	2186.8	47.4	1.6	0.5	2688.1	30.7	2292.7	22.7	0.0	0.0
bcsstk30	28924	6394	34071.1	922.5	1640.6	118.0	4.1	1.2	7178.9	87.5	307.9	159.5	0.4	0.0
bcsstk31	35588	2762	37912.9	865.7	1792.2	123.9	4.8	1.5	8519.5	120.9	1000.0	144.7	0.2	0.0
fepwt	36519	340	19951.8	348.7	5306.7	117.9	4.9	1.5	7027.0	79.4	5048.9	135.4	0.1	0.0
bcsstk32	44609	4667	44251.7	1285.9	2393.4	119.0	9.1	1.4	10755.1	92.8	907.2	161.1	0.3	0.0
febody	45087	262	27986.7	1104.0	8236.9	189.8	9.4	2.2	9015.5	81.7	6690.9	139.1	0.1	0.0
t60k	60005	79	21960.8	160.4	16219.6	122.1	24.8	12.9	10406.0	62.0	13788.9	131.9	0.1	0.0
wing	62032	789	23710.2	1345.3	2723.5	41.0	19.3	2.4	7325.1	52.2	3033.3	23.1	0.1	0.0
brack2	62631	731	64107.9	1144.7	5417.3	216.1	19.4	4.9	15322.5	177.3	4431.5	219.9	0.2	0.0
finan512	74752	162	41834.3	1264.8	29300.2	421.9	33.1	10.3	14000.9	143.7	22190.7	544.2	0.1	0.0
fetooth	78136	3816	78715.1	1203.3	1198.6	30.2	33.2	6.3	18977.2	264.6	1015.1	49.3	0.2	0.0
ferotor	99617	2098	95356.7	1986.9	2550.0	120.3	52.4	13.6	25644.5	267.6	2606.3	107.8	0.4	0.0
598a	110971	2398	110406.1	1705.7	1969.3	72.6	176.2	48.0	29271.1	258.1	2467.2	87.8	0.6	0.0
feocean	143437	464	74396.1	1815.4	15249.2	312.1	217.5	107.8	24111.5	95.4	14659.6	123.6	0.2	0.0
144	144649	6486	146677.5	3023.4	1144.0	52.8	261.5	116.1	38392.5	344.3	1283.5	38.0	0.9	0.0
wave	156317	8677	164300.9	3693.1	1043.7	66.0	337.4	137.1	41784.5	515.8	970.1	32.1	0.6	0.0
m14b	214765	3836	230879.5	3760.5	2996.9	139.7	495.2	125.9	58047.9	620.5	3274.6	119.4	1.4	0.0
auto	448695	10101	464066.2	7143.9	2355.6	66.1	2715.7	797.0	121709.6	724.2	2618.2	72.5	3.8	0.5

e obtemos

$$T_{\mathrm{FI}}(n) = 0.3 \cdot n^{2.3}\,\mathrm{ns}$$
$$T_{\mathrm{BI}}(n) = 187 \cdot n^{1.3}\,\mathrm{ns}$$

Tabela 3: Resultados da busca BLMR para diferentes valores de p.

	0			0.05			0.25			0.5			0.75			1		
Inst	$\#i(M)$	r	t (s)	$\#i(M)$	r	t (s)	$\#i(M)$	r	t (s)	$\#i(M)$	r	t (s)	$\#i(M)$	r	t (s)	$\#i(M)$	r	t (s)
add20	373.5	57.7	300.4	344.1	10.0	300.5	180.8	7.7	300.9	117.5	15.6	300.9	99.2	192.0	300.9	87.0	523.4	300.5
data	251.9	616.8	300.9	212.9	215.7	300.8	111.3	196.1	300.8	71.6	164.9	300.8	76.0	2495.8	300.6	67.4	3899.0	300.6
3elt	355.7	2084.2	300.8	230.9	133.3	300.7	87.2	132.4	300.8	51.1	278.9	300.5	72.5	5047.8	300.6	55.8	7503.3	300.6
uk	847.2	4594.7	300.8	344.4	358.9	300.9	101.7	232.6	300.8	55.8	1036.8	300.8	71.5	11860.0	300.4	60.0	18004.2	300.8
add32	782.7	6718.2	300.7	379.2	3138.2	300.6	161.3	2276.4	300.5	90.7	1630.9	301.1	84.7	25005.5	300.4	67.6	43170.9	300.4
bcsstk33	32.5	38.0	300.8	27.4	34.2	300.8	19.5	27.2	300.8	14.4	50.6	300.9	15.1	879.5	300.6	13.1	1335.1	300.8
whitaker3	498.3	3033.4	300.8	152.6	257.6	300.9	39.4	201.6	300.6	21.1	357.2	301.1	40.6	7757.5	300.7	29.2	11346.6	301.0
crack	253.3	2222.0	300.9	149.4	237.3	300.4	55.6	199.6	300.8	30.8	191.5	300.5	36.6	5806.7	300.7	26.8	8146.0	300.7
wingnodal	272.8	249.2	300.7	145.6	126.5	300.6	62.5	96.8	300.5	36.3	28.7	300.5	38.8	1426.8	300.5	27.7	2115.9	301.0
fe4elt2	436.6	3210.0	301.1	176.4	454.5	301.0	49.1	375.1	300.7	25.4	319.2	300.7	40.5	8579.4	300.8	29.0	12558.0	301.0
vibrobox	69.6	38.7	300.9	51.1	29.4	300.7	39.3	24.7	300.5	28.6	24.2	300.5	24.8	408.1	301.2	19.1	699.3	300.6
bcsstk29	56.7	471.3	300.7	39.2	472.2	300.6	25.4	432.0	301.0	19.3	271.9	300.6	19.0	3323.3	300.5	15.4	5218.4	300.5
4elt	408.5	4409.2	301.1	85.4	566.3	300.9	20.9	484.2	300.8	10.7	335.0	300.7	24.5	11014.1	300.8	16.6	16357.8	300.7
fesphere	348.1	1694.8	301.1	60.6	64.0	300.3	13.9	68.2	300.8	7.4	142.6	300.7	19.4	4194.3	300.5	13.2	6268.0	300.9
cti	333.0	2382.8	300.8	109.3	589.6	301.0	29.1	554.3	300.8	12.5	222.3	300.9	25.9	4854.3	301.0	18.4	7128.6	300.5
memplus	213.2	63.2	300.7	99.7	36.7	300.5	32.8	34.1	300.6	18.3	38.9	300.7	21.1	180.9	300.9	16.6	393.5	300.6
cs4	445.0	2288.3	300.6	70.9	730.9	300.5	16.3	678.3	300.9	6.5	204.1	300.9	12.0	3857.1	300.9	9.3	5826.0	300.5
bcsstk30	202.7	260.0	300.7	58.1	263.4	300.6	18.9	256.5	300.8	10.7	363.3	301.0	11.4	4899.6	301.2	8.7	7773.9	300.9
bcsstk31	82.6	1060.1	300.6	54.1	964.6	301.0	26.7	913.9	300.4	15.3	667.8	301.0	16.6	5648.9	300.7	11.1	10270.2	301.2
fepwt	253.3	5057.2	300.4	31.7	1068.2	300.5	6.8	966.5	300.9	3.4	839.7	301.1	9.9	14419.5	300.6	6.2	21171.6	301.0
bcsstk32	77.1	902.2	301.2	51.0	921.1	301.0	21.0	1011.3	301.0	12.3	582.2	301.3	12.1	6423.9	300.9	8.8	10452.4	300.9
febody	744.9	6733.1	300.7	105.0	3157.6	301.4	22.4	2782.6	301.1	10.6	1533.4	300.9	13.5	20918.3	300.8	7.8	31113.4	300.7
t60k	766.7	13725.3	301.0	20.8	4341.5	300.5	3.8	3683.8	301.2	1.8	2791.4	300.8	3.0	37624.3	301.0	2.2	56596.7	300.7
wing	723.1	3048.9	300.9	25.9	1057.7	301.1	5.2	1025.9	300.9	2.4	703.1	300.5	3.6	5011.4	301.0	2.7	7597.6	300.8
brack2	159.1	4429.6	300.6	81.2	2490.2	300.7	25.5	2309.2	301.0	13.2	976.1	301.2	12.8	16790.6	301.2	6.3	24970.6	301.2
finan512	308.6	22171.6	300.7	54.8	10156.4	300.7	13.2	9629.5	300.4	6.7	8772.5	300.8	6.3	50247.3	300.9	3.4	80575.3	300.8
fetooth	243.0	1016.7	300.7	70.7	629.3	300.7	20.9	560.3	300.9	10.9	192.1	300.8	10.6	3902.9	300.4	5.1	5836.4	300.9
ferotor	214.1	2682.8	300.7	43.7	1572.6	300.9	11.1	1517.3	301.1	5.5	999.0	300.9	7.7	10754.6	300.7	3.7	15690.2	301.0
598a	91.0	2486.6	301.7	25.6	1499.5	301.1	8.3	1402.9	301.2	4.5	761.4	300.8	5.2	10380.1	300.6	3.0	15365.9	301.0
feocean	383.6	14633.5	300.9	12.3	6120.5	301.0	2.3	5883.7	300.9	0.9	5479.1	301.0	1.8	29943.5	300.6	1.3	44053.1	300.8
144	86.8	1290.6	301.6	22.3	864.8	300.8	5.7	841.3	301.0	3.8	570.9	301.5	4.0	5722.8	300.8	1.7	8181.4	301.0
wave	119.9	985.3	301.1	19.9	617.4	301.4	5.0	601.7	300.9	2.4	427.8	300.9	2.8	4129.4	300.8	1.6	6000.9	300.7
m14b	92.6	3211.5	301.2	15.4	2321.4	301.7	3.6	2319.2	301.4	1.6	2277.5	301.2	1.6	15177.6	300.8	0.6	21780.2	301.1
auto	83.2	2639.5	302.1	4.9	2164.9	302.2	0.9	2244.3	302.0	0.7	3495.0	302.2	0.5	11864.0	301.9	0.3	16301.3	302.2
Avgs	312.1	3544.3	300.9	99.3	1402.0	300.9	36.7	1293.3	300.9	21.3	1080.8	300.9	24.9	10315.9	300.8	19.0	15712.5	300.8