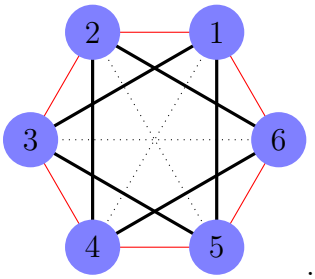
Soluções 1

Exercício 1 (Vizinhanças no PCV)
A vizinhança é simétrica, porque podemos desfazer um movimento por remover a última cidade deslocada e inserí-la na sua posição anterior. Ela é fracamente otimamente conectada, porque qualquer permutação de cidades pode ser produzida: para cidades $c_1, c_2, \dots c_n$, remove da rota atual c_2 e insere depois de c_1, remove c_3 insere depois de c_2, etc. Ela não é exata: considere o grafo



em que as arestas pontilhadas possuem peso 1, as vermelhas peso 2, e as arestas pretas peso 3. A rota vermelha possui peso total 12, o rota mínimo é 1452361 com peso total 9. Removendo qualquer cidade vai remover duas arestas vermelhas incluir uma aresta preta na solução que diminui o peso total por 1. Inserir em qualquer outra posição entre duas cidades vai remover uma aresta vermelha e inserir uma preta e uma vermelha ou uma pontilhada. No melhor caso isso resulta num incremento por 1. Logo todos vizinhos possuem peso igual ou maior e a rota é um mínimo local não-estrito (para obter um mínimo local estrito o peso das arestas pretas pode ser aumentado para 4).
Uma outra abordagem: Como o número de vizinhos é $O(n^2)$, pelo teorema 2.3 da notas, a vizinhança só pode ser exata caso $\mathsf{P} = \mathsf{NP}$.

Exercício 2 (Aproximação do PCV por busca local, 2.5pt)
(Pela dificuldade de decidir o "gap problem".) Supõe que temos tal busca local. Caso $\mathsf{P} \neq \mathsf{NP}$ em contradição com a NP-completude de RHC podemos resolver o problema RHC em tempo polinomial como segue. Para uma instância do RHC $G = (V, A)$ com caminho Hamiltoniano p constrói uma instância do PCV com arestas de distância 1 caso $a \in A$ e arestas de distância $\alpha(n)n$ caso contrário. Isso é possível em tempo polinomial porque podemos computar $\alpha(n)$ em tempo polinomial. Essa instância possui valor ótimo n, caso o RHC possui uma solução. Caso contrário a solução c obtida por fechar o caminho p possui o valor ótimo $n - 1 + \alpha(n)n$. Mas caso uma busca local com a vizinhança desejada existe, podemos decidir RHC em tempo polinomial: c possui um vizinho melhor sse RHC possui uma solução.

Exercício 3 (Busca local para $Pm \mid\mid C_{\max}$, 2.5pt)
A Tabela 1 mostra os resultados. Os desvios relativos são do limite inferior simples $L = \max\{\max_i p_i, \sum_i p_i/m\}$. O arquivo com os resultados detalhados é disponível em `www.inf.ufrgs.br/msc/pmmm.dat`, uma implementação em `www.inf.ufrgs.br/msc/pcmax.cpp`.

Tabela 1: Resultados para $P \mid\mid C_{\max}$.

Alg.	Tempo (ms)	Iterações	Desvio relativo (%)
PM	29.37 ± 80.18	6536.66 ± 15271.18	6.05 ± 6.89
MM	212.56 ± 583.41	5886.02 ± 13895.30	5.60 ± 6.57
LS	0.00 ± 0.00	-	20.39 ± 17.83

a) Podemos observar que a "melhor melhora" produz ligeriamente melhores resultados que a "primeira melhora", com um desvio relative médio aprox. 0.4 menor, mas precisa uma fator mais que 7 mais tempo. As complexidades empíricas dos algoritmos são

$$\begin{aligned} T_{\mathrm{MM}}(n) &= 82 \times n^{2.16} ns, \\ T_{\mathrm{PM}}(n) &= 48 \times n^{1.96} ns, \\ I_{\mathrm{PM}}(n) &= 0.55 \times n^{1.40}, \\ I_{\mathrm{MM}}(n) &= 0.91 \times n^{1.44}. \end{aligned}$$

b) Em comparação com as buscas locais o "list scheduling" é bem mais rápido: todos casos de teste precisaram menos que um 1ms. A qualidade com um desvio relativo de 20% é significativamente pior, mas muito melhor que a garantia teórica (de 100%).

c) Considere um mínimo local da busca local acima (independente da estratégia). Escolhe uma máquina crítica c (i.e. $C_{\max} = C_c$) da solução correspondente e uma tarefa de menor tempo de processamento p_j nessa máquina. Por ser um mínimo local, movendo a tarefa j para qualquer outra máquina i produz um tempo total pelo menos $C_{\max}$: $C_i + p_j \geq C_{\max}$. Logo

$$C_{\max} \leq \sum_{i\in[m]} (C_i + p_j)/m \leq \sum_{i\in[m]} C_i/m + p_{\max} = \sum_{j\in[n]} p_j/m + p_{\max} \leq C^*_{\max} + C^*_{\max} = 2C^*_{\max}.$$

Uma análise mais detalhada mostra que a qualidade de aproximação de fato é ligeiramente melhor. Escolha a máquina crítica c com o menor número de tarefas. Caso c possui somente uma tarefa, a solução é ótima. Logo podemos supor que c possui pelo menos duas tarefas, e como j é a tarefa de menor tempo, sabemos que $p_j \leq C_{\max}/2$. Com isso obtemos

$$\begin{aligned} C_{\max} &\leq \sum_{i\in[m]\setminus\{c\}} (C_i + p_j)/m + C_c/m \leq \sum_{i\in[m]\setminus\{c\}} C_i/m + (m-1)p_j/m + C_c/m \\ &= \sum_{i\in[m]} C_i/m + (m-1)C_{\max}/2m \leq C^*_{\max} + (m-1)C_{\max}/2m. \end{aligned}$$

Logo

$$(m+1)/2mC_{\max} \leq C^*_{\max} \to C_{\max} \leq (2 - 2/(m+1))C^*_{\max}.$$

Tabela 2: Resultados para QBP. Médias de 5 replicações.

Alg	p	Tempo (ms)	Iterações	Desvio relativo (%)
BL	0.00	22.70 ± 33.55	8264.65 ± 12742.39	0.00 ± 0.00
BL	0.05	20.20 ± 22.07	7959.40 ± 8864.07	0.00 ± 0.00
BL	0.25	25.45 ± 34.68	9271.35 ± 10722.92	0.00 ± 0.00
BL	0.50	150854.55 ± 153028.53	36662803.45 ± 39844800.58	7.13 ± 10.87
BL	0.75	227074.05 ± 129624.46	84068141.80 ± 62933312.00	29.01 ± 21.92
BL	1.00	300000.00 ± 0.00	174437888.90 ± 68957421.98	51.41 ± 21.75
MM	-	0.65 ± 0.88	102.50 ± 87.27	6.25 ± 5.12
PM	-	0.85 ± 1.31	212.40 ± 219.09	8.59 ± 8.98

(Em geral, caso existe uma máquina crítica com $k \geq 2$ tarefas, o list scheduling consegue uma $1 + (m-1)/((k-1)m+1)$-aproximação. Então um algoritmo pode dar uma garantia melhor, verificando qual a máquina crítica com o maior número de tarefas no mínimo local.)

Exercício 4 (Busca local para QBP, 3.5pt)
A Tabela 2 mostra os resultados. Os desvios relativos são da solução ótima. O arquivo com os resultados detalhados é disponível em `www.inf.ufrgs.br/msc/bqp.log`, uma implementação em `www.inf.ufrgs.br/msc/bqp.cpp`.

a) Neste caso a MM produz melhores resultados que PM, em menos tempo e iterações.

b) A BLMR é o melhor método, resolve todas instâncias otimamente para $p \leq 0.25$, sendo $p = 0.05$ ligeiramente melhor em tempo e número de iterações.

c) A complexidade empírica só pode ser determinada para $p \leq 0.25$. Obtemos

$$T_{\text{BLMR}/0.05}(n) = 368.1 \times n^{0.71} \mu s,$$
$$T_{\text{BLMR}/0.25}(n) = 414.5 \times n^{0.69} \mu s,$$
$$I_{\text{BLMR}/0.05}(n) = 36.2 \times n^{0.91},$$
$$I_{\text{BLRM}/0.25}(n) = 13.4 \times n^{1.08}.$$