

Soluções 1

Exercício 1 (Vizinhanças)

A vizinhança 2-flip é simétrica (porque a distância Hamming é simétrica), não fracamente otimamente conectada (porque 00 não é alcançável por 11), e logo não exata. O mesmo se aplica para um k -flip com k par (porque o número de bits setados sempre difere por $2k$ do número de bits iniciais). Isso se aplica ainda para k ímpar caso temos exatamente k bits. Porém com k ímpar e pelos menos $k + 1$ bits, podemos aplicar uma sequência de k k -flips, tal que o k -flip $i \in [k]$ troca os bits $[k + 1] \setminus \{k + 2 - i\}$. Por exemplo para $k = 5$ vamos trocar os bits $\{1, 2, 3, 4, 5\}$, $\{1, 2, 3, 4, 6\}$, $\{1, 2, 3, 5, 6\}$, $\{1, 2, 4, 5, 6\}$, $\{1, 3, 4, 5, 6\}$. Isso troca cada bit $2, \dots, k + 1$ um número par de vezes, e o bit 1 um número ímpar de vezes, logo inverte o bit 1. Por consequência com pelo menos $k + 1$ bits a vizinhança é simétrica e fracamente otimamente conectada. Mas ela ainda não é exata. Supõe, por exemplo, que o número cláusulas satisfeitas é igual a número de variáveis com valor 1 (i.e. $\varphi = x_1 \wedge x_2 \wedge \dots \wedge x_n$). Para uma solução com distância Hamming $d \in [1, k]$ da solução ótima $x_1 = \dots = x_n = 1$, cada vizinho é pior.

Exercício 2 (PCV)

Como a aresta de menor distância é $1/n\alpha$ um ciclo é ótimo somente se todas suas arestas são *curtas* (i.e. possuem distância $1/n\alpha$). Temos $n + k$ arestas curtas: $n - 1$ arestas *circulares* $(2, 3), \dots, (n - 1, n), (n, 1)$, uma aresta *terminal* $(k + 3, 2k + 4)$ e k arestas *pontes* $(j, 2k + 4 - j)$ para $j \in [k]$. Restrito para tais arestas, os vértices incidentes a arestas pontes exceto 2 e $k + 4$ (“vértices pontes”) e os vértices da aresta terminal possuem grau 3, o restantes $n - 2k$ vértices grau 2.

Afirmamos que todas arestas pontes necessariamente fazem parte de uma solução ótima. Isso exclui $k - 1$ arestas curtas incidentes as vértices pontes. Além disso a aresta $(2k + 3, 2k + 4)$ não faz parte de uma solução ótima, porque o caminho $(2k + 3, 1, n, n - 1, n - 2, \dots, 2k + 5, 2k + 4)$ cujos vértices interiores possuem grau 2 precisa fazer, e $(2k + 3, 2k + 4)$ fecharia um subciclo. Isso exclui k das $n + k$ arestas candidatas, logo a solução ótima é única.

Resta provar a afirmação. Supõe que j é o menor índice tal que a j -ésima aresta ponte $(j, 2k + 4 - j)$ não faz parte de uma solução ótima. Temos $j > 2$ porque $(1, 2k + 3)$ e $(2, 2k + 2)$ são necessárias, já que os vértice 1 e 2 possuem grau 2. Como a j -ésima aresta ponte falta, os quatro arestas incidentes às vértices j e $2k + 4 - j$ que sobram fazem parte da solução ótima e as arestas $(j - 2, j - 1)$ e $(2k + 4 - j + 2, 2k + 4 - j + 1)$ não. Para $j = 3$ isso exclui a aresta $(2k + 3, 2k + 2)$ e força $(2k + 3, 2k + 4)$, uma contradição (ver acima). Para $j > 3$ a exclusão de $(j - 2, j - 1)$ e $(2k + 4 - j + 2, 2k + 4 - j + 1)$ força as arestas $(j - 3, j - 2)$ e $(2k + 4 - j + 3, 2k + 4 - j + 2)$ que fecha um subciclo com as arestas ponte $j - 2$ e $j - 3$, novamente uma contradição.

Exercício 3 (Busca local para um problema polinomial)

Uma implementação é disponível em <http://www.inf.ufrgs.br/~mrpritt/e01.zip>. Em comparação a “melhor melhora” (MM) precisa, como esperado, menos passos que a “primeira melhora” (PM). Experimentalmente o número de passos da MM é 65% do número de passos da PM. Porém a redução em número de passos não compenso o tempo

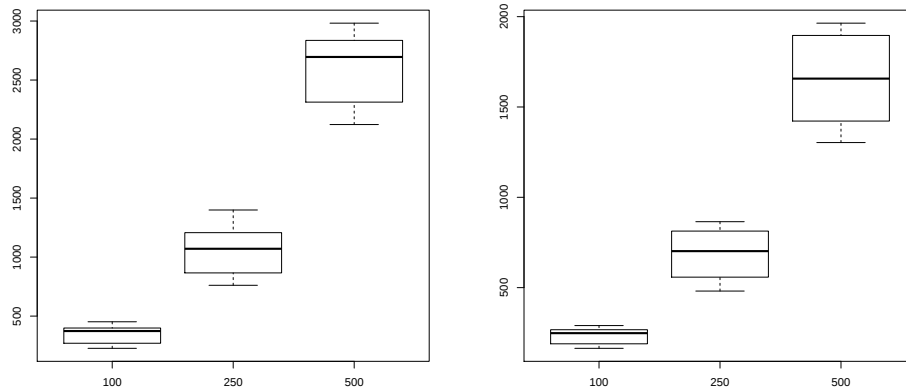


Figura 1: Número de passos “primeira melhora” (esquerda) e “melhor melhora” (direita) para o AGM.

adicional para encontrar o melhor movimento. A MM é um fator entre aproximadamente 4 e 17 vezes mais lento, com uma diferença crescente com o número de vértices. Uma outra observação é a variação considerável do número de passo (e consequentemente tempo). Com pesos em $[1, 100]$ o valor da função objetivo fica entre $n - 1$ e $100(n - 1)$, logo o número de iterações é limitado por $99(n - 1)$. Isso é consistente com o número observada de passos, que é ligeiramente superlinear.

Para determinar a complexidade empírica, vamos supor que os dois algoritmos são polinomiais (que é justificado aqui porque o número de passos é $O(n)$ e cada iteração é $O(n^2)$.) Uma regressão linear para $\log T \sim \log n$ produz as complexidades empíricas

$$T_b(n) = 4.1\mu s \cdot n^{3.8}$$

$$T_f(n) = 27\mu s \cdot n^{3.1}$$

$$S_b(n) = 0.85 \cdot n^{1.2}$$

$$S_f(n) = 1.06 \cdot n^{1.2}.$$

Isso confirma um número de passos ligeiramente superlinear, e uma complexidade maior da MM. Em comparação o algoritmo de Prim tem complexidade $O(m + n \log n) = O(n^2)$, i.e. a busca local é empiricamente, comparado com o pior caso do Prim entre $n^{1.1}$ e $n^{1.8}$ pior. (Mas observe que a complexidade empírica do algoritmo de Prim também é menos que $O(n^2)$.)

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

Uma implementação é disponível em <http://www.inf.ufrgs.br/~mrpritt/e01.zip>. Vale a pena considerar também implementações do Knuth em <http://www-cs-faculty.stanford.edu/~uno/programs.html>.

A Tabela 1 apresenta os resultados do GSAT e WalkSAT do exercício, para um tempo limite de 120s, e os resultados do WalkSAT/SKC, uma variante com uma seleção da

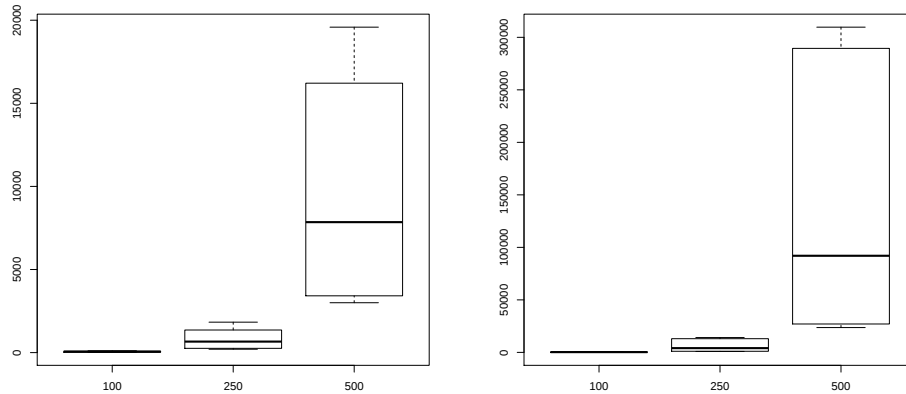


Figura 2: Tempo (em s) “primeira melhora” (esquerda) e “melhor melhora” (direita) para o AGM.

variável invertida melhor. O GSAT consegue resolver todas instâncias com tempo e número de passos menor que o WalkSAT, que resolve somente as duas menores instâncias. O WalkSAT/SKC sempre é melhor que os outros algoritmos. Podemos ver ainda que a tabela é consistente com a estatística: com uma probabilidade p para encontrar uma solução num dado período, a número esperado de períodos segue uma distribuição geométrica com valor esperado $1/p$ e desvio padrão $\sqrt{1-p}/p \approx 1/p$ (para p pequeno) em concordância com desvio padrão na ordem do valor médio. A probabilidade de encontrar uma solução num dado período para o WalkSAT/SKC e flat100-1, por exemplo, é $\approx 1/(615700/900) \approx 0.0015$ (porque o período é $3 \times 300 = 900$).

Todas instâncias possuem $k = 3$, logo o teorema do Schönig garante uma complexidade $T(n) \sim n^{3/2}(4/3)^n$. Para comparar vamos supor uma complexidade empírica $T(n) \sim ab^n$. Para o WalkSAT é impossível determinar a complexidade empírica, porque (i) só temos duas medidas e (ii) enquanto par8-5-1 tem menos variáveis a instância é mais difícil, i.e. nestes dois pontos a complexidade em função de n diminui.

Para o GSAT e WalkSAT/SKC uma regressão linear $\log T \sim n$ limitada para as instâncias flat produz as complexidades empíricas

$$T_{\text{GSAT}}(n) = 0.245 \cdot 1.036^n$$

$$T_{\text{WalkSAT/SKC}}(n) = 0.026 \cdot 1.037^n$$

Logo a complexidade empírica é bem melhor que a complexidade pessimista do Schönig (que não é uma surpresa: estamos comparando o caso médio de algumas instâncias com o caso pessimista). Além disso o modelo provavelmente ainda é pessimista: a predição para WalkSAT/SKC e flat200-1 são 18 horas, mas um teste (limitado) revela que o algoritmo resolve essa instância em ≈ 174 s.

Tabela 1: Resultados Max-SAT para tempo limite 120s. Em parênteses: número de instâncias resolvidas, caso diferente de 15.

Inst.	GSAT		WalkSAT	
	T (ms)	# passos (K)	T (ms)	# passos (K)
flat50-1	44.1 ± 46.6	7.8 ± 8.5	17503 ± 14473	12704 ± 10524
par8-5-c	189.8 ± 149.2	51.5 ± 40.8	57119 ± 45532 (3)	41830 ± 33355
flat75-1	974.6 ± 908.7	122.1 ± 112.0	-	-
flat100-1	9178.7 ± 8603.8	911.8 ± 853.1	-	-

WalkSAT/SKC		
Inst.	T (ms)	# passos (K)
flat50-1	4.7 ± 6.5	3.4 ± 4.1
par8-5-c	137.8 ± 214.9	98.0 ± 153.8
flat75-1	134.7 ± 98.7	85.2 ± 62.8
flat100-1	1050.1 ± 817.1	615.7 ± 480.3

Observação 1 (Regressão linear para polinômios e funções exponenciais)

Para determinar a complexidade empírica, supondo que a complexidade possui a forma $T(n) = c_1 n^{c_2}$: gera um arquivo `x.dat` com um tamanho da instância e um tempo por linha. Por exemplo

```
n T
100 233.0000
250 689.7667
500 1655.8667
```

Chama “R” (GNU R) e usa os comandos

```
d<-read.table("x.dat",header=T)
lm(log(T)~log(n),data=d)
```

Com coeficientes a e b obtidos, o modelo $\log T = a + b \log n$ ou $T = e^a n^b$. Para a hipótese $T(n) = c_1 c_2^n$ usa

```
lm(log(T)~n,data=d)
```

com coeficientes a e b obtidos o modelo é $\log T = a + bn$ ou $T = e^a e^{bn}$ ◇