Heurísticas Modernas para Problemas de Otimização Combinatória

Luiz A N Lorena
Pesquisador Titular
LAC/INPE
lorena@lac.inpe.br
http://www.lac.inpe.br/~lorena

Problemas de Otimização Combinatória aparecem quando é necessário selecionar de um conjunto discreto e finito de dados, o melhor subconjunto que satisfaz a determinados critérios. Por exemplo, selecionar o melhor conjunto de itens indivisíveis a serem transportados em um veículo de capacidade limitada.

BUSCA TABU

Busca Tabu (Algoritmo básico)

s_0 = solução inicial;
f_melhor = *f* (s_0);
niter = 0 (número de iterações sem melhora);
melhiter = 0 (número da iteração da melhor solução encontrada);
nbmax = número máximo de iterações sem obter uma melhora na solução;
Inicializar a *lista Tabu L* ;
Inicializar a função do *Critério de Aspiração*, *A* (*f* (s_0));
$s = s_0$;
Enquanto (*niter - melhiter* < *nbmax*) ***faça***
f_atual = *f_melhor*;
niter = *melhiter* + 1;
// Pesquisa de vizinhança //
Gerar um conjunto V_1 de soluções s_i em *N* ;
Escolha a melhor solução s_i^+ em *V* que não é tabu ou *f* (s_i^+) < *A*(*f* (s) ;
Atualize a lista tabu *L* e a função critério de aspiração *A* ;
Se *f* (s_i^+) < *f_melhor* ***então***
f_melhor = *f* (s_i^+),
$s = s_i^+$;
fim_se
Se *f_melhor* < *f_atual* ***então*** *melhiter* = *niter* ;
fim_enquanto.

Fred Glover (1989)

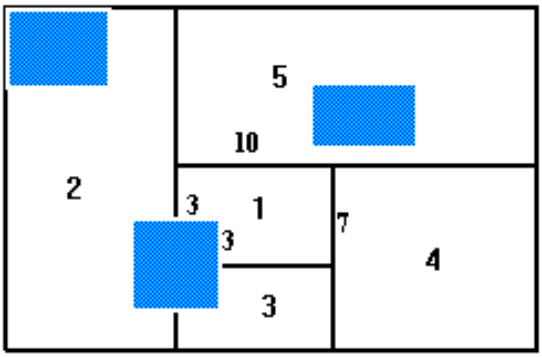
LEIAUTE

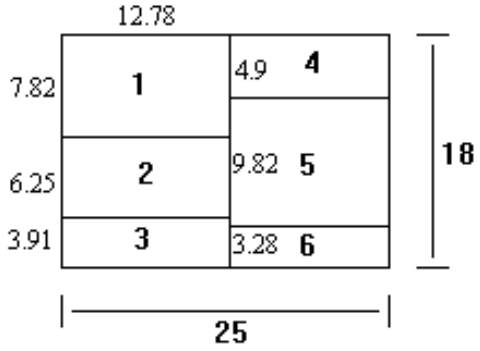
$$Min \sum_{i=1}^{n}\sum_{j=1}^{n} qij\, d_{ij}\,, \quad x \hat{I} x$$

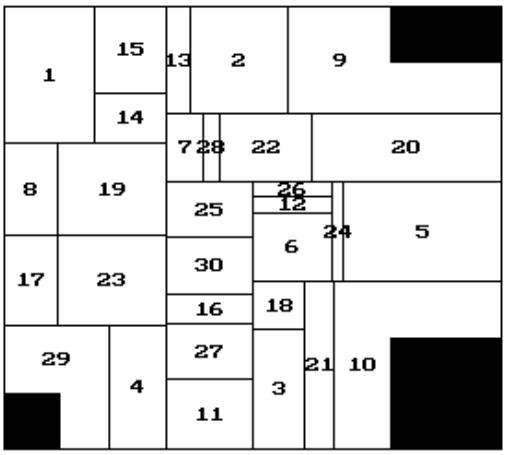
sujeito a: $a_{i(min)} \pounds a_i \pounds a_{i(max)}$

$$0 \pounds s_i \pounds s_{i(max)}$$

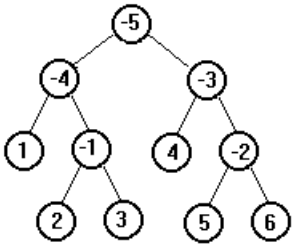
$$i = 1, 2, \ldots, n.$$

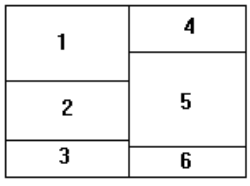






Funcao objetivo (global):24152

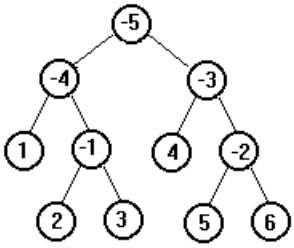


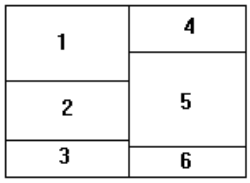


(1,2,3,4,5,6)

Busca Tabu

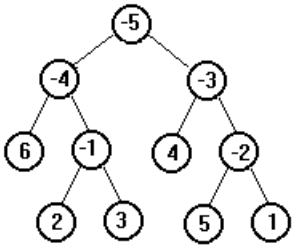
→ Duas vizinhanças

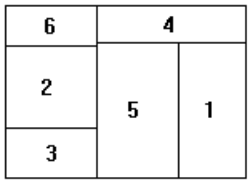




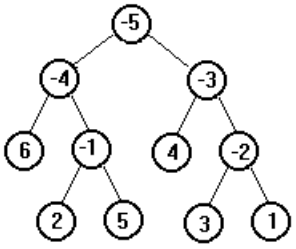
s = (1,2,3,4,5,6) ou ((1(23))(4(56)))

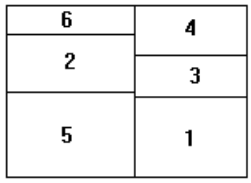
1ª Vizinhança:
Troca de duas folhas





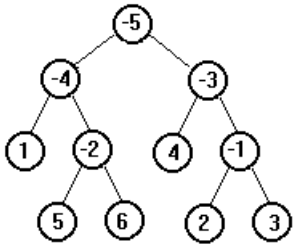
s' = (6,2,3,4,5,1) ou ((6(23))(4(51)))

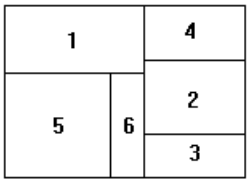




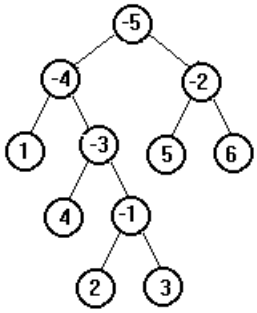
s' = (6,2,5,4,3,1) ou ((6(25))(4(31)))

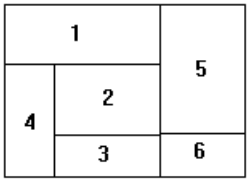
2ª Vizinhança:
Troca de dois nós internos





s = (1,5,6,4,2,3) ou ((1(56))(4(23)))





s' = (1,4,2,3,5,6) ou ((1(4(23)))(56))

SISTEMAS DE MANUFATURA FLEXÍVEIS

Problema de “Scheduling” com restrições

- Cada parte requer um conjunto de ferramentas.
- A máquina possui um magazine com capacidade limitada. Esta restrição é crítica, pois restringe o número de ferramentas que podem ser carregadas e, conseqüentemente o número de partes que podem ser produzidas continuamente.
- Não existe “loop” no sistema. Uma parte que começa a ser processada é finalizada e não retorna. Isto significa que todas as partes são processadas e que o conjunto de ferramentas necessário para processar cada parte deve estar no magazine antes do início do processamento.
- Os tempos de “setup” são considerados separados dos tempos de processamento das partes, sendo constituídos por dois períodos: período de preparação do sistema para reinício de processamento (preparação da máquina e de dispositivos, limpeza da área de trabalho e manutenções) e o período de tempo gasto na troca de ferramentas. O tempo de preparação do sistema para o reinício de processamento é considerado fixo. O tempo de troca de ferramentas é considerado proporcional ao número de ferramentas trocadas.
- Os turnos de produção são respeitados; no momento de término de um turno, nenhuma parte pode estar em processamento ou vir a ser processada.

Para uma dada seqüência s_i de partes a serem processadas, a função objetivo a ser minimizada é definida pela soma dos seguintes termos:

$$f(s_i) = P_1 . Tp(s_i) + P_2 . At(s_i) + P_3 . Sp(s_i) + P_4 . Sf(s_i) + P_5 . Tr(s_i),$$

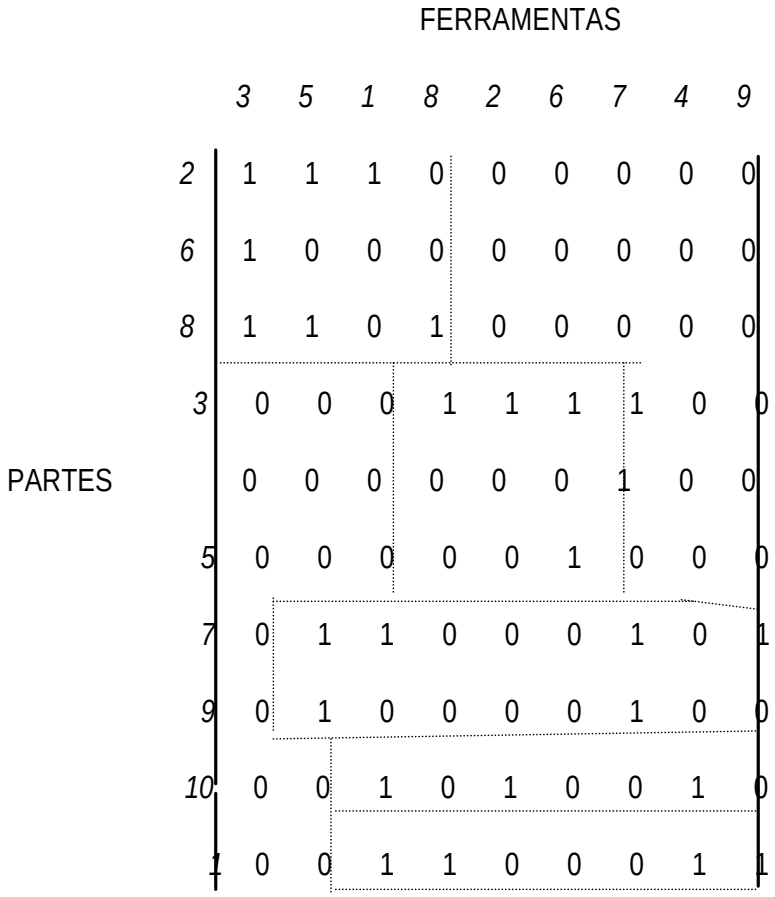
onde:
$Tp(s_i)$ é o tempo total de produção ("makespan"),
$At(s_i)$ é o tempo total de atraso,
$Sp(s_i)$ é o tempo total de "setup" de preparação para reinício de operação do sistema,
$Sf(s_i)$ é o tempo total de "setup" referente a troca de ferramentas,
$Tr(s_i)$ é o tempo total ocioso dos turnos.

A cada parcela da função objetivo é associado um peso. Com relação as parcelas acima P_1, P_2, P_3, P_4 e P_5. Sendo que, conforme os valores definidos para os pesos a função objetivo poderá refletir várias estratégias.

		FERRAMENTAS								
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
PARTES	*1*	1	0	0	1	0	0	0	1	1
	2	1	0	1	0	1	0	0	0	0
	3	0	1	0	0	0	1	1	1	0
	4	0	0	0	0	0	0	1	0	0
	5	0	0	0	0	0	1	0	0	0
	6	0	0	1	0	0	0	0	0	0
	7	1	0	0	0	1	0	1	0	1
	8	0	0	1	0	1	0	0	1	0
	9	0	0	0	0	1	0	1	0	0
	10	1	1	0	1	0	0	0	0	0

Capacidade do magazine de *quatro* ferramentas.

Algoritmo de Identificação de Grupos



FP ← Família de partes
FF ← Família de ferramentas
FP1 = {2,6,8} e FF1 = { 1,3,5,8}.
FP2 ={3,4,5} e FF2 ={2,6,7,8}.
FP3={7,9} e FF3 = {1,5,7,9}.
FP4 ={10} e FF4 = {1,2,4}.
FP5={1} e FF5 = {1,4,8,9}.

Seqüência s = (6,3,4,5,2,8,9,7,10,1)

PARTES	FP	MAGAZINE TIPOS DE FERRAMENTAS	Nº DE TROCAS
6	1	1 3 5 8	-
3 , 4 , 5	2	2 6 7 8	3
2 , 8	1	1 3 5 8	3
9 , 7	3	1 5 7 9	2
10	4	1 2 4 9	2
1	5	1 4 8 9	1
Nº de Sp = 5			Nº de Sf = 11

5 instantes de parada para troca de ferramentas (Sp) e
11 trocas de ferramentas (Sf).

Busca Tabu

ALGORITMO TSS

s_0 = seqüência inicial;
f_melhor = *f* (s_0);
niter = 0 (número de iterações sem melhora);
melhiter = 0 (número da iteração da melhor solução encontrada);
nbmax = número máximo de iterações sem obter uma melhora na solução;
Inicializar as *listas Tabu* para duas pesquisas, L_1 e L_2.
Inicializar a função do *Critério de Aspiração*, *A* (*f* (s_0));
$s = s_0$;
Enquanto (*niter - melhiter* < *nbmax*) ***faça***
f_atual = *f_melhor*;
niter = *melhiter* + 1;
// Primeira pesquisa de vizinhança //
Gerar um conjunto V_1 de seqüências s_i em N_1 ;
Escolha a melhor seqüência s_i^+ em V_1 que não é tabu ou $f(s_i^+) < A(f(s)$;
Atualize a lista tabu L_1 e a função critério de aspiração *A* ;
Se $f(s_i^+) <$ *f_melhor* ***então*** *f_melhor* = $f(s_i^+)$,
$s = s_i^+$;
fim_se
// Segunda Pesquisa de Vizinhança //
Gerar um conjunto V_2 de seqüências s_i em N_2 ;
Escolha a melhor seqüência s_i^+ em V_2 que não é tabu ou $f(s_i^+) < A(f(s)$;
Atualize a lista tabu L_2 e a função critério de aspiração *A* ;
Se $f(s_i^+) <$ *f_melhor* ***então*** *f_melhor* = $f(s_i^+)$,
$s = s_i^+$;
fim_se
Se *f_melhor* < *f_atual* ***então*** *melhiter* = *niter* ;
fim_enquanto.

→ Duas vizinhanças

s = (6,3,4,5,2,8,9,7,10,1)

1ª Vizinhança:
representação de lotes → s = [(6),(3,4,5), (2,8), (9,7), (10), (1)]

posicionamento de dois lotes é alterado por um movimento de troca,

por exemplo → s' = [(6), (10), (2,8), (9.7), (3,4,5), (1)]

2ª Vizinhança:
representação de partes → s = (6,10,2,8,9,7,3,4,5,1)
uma parte passa a ser processada em outro lote por movimentos de retirada e de inserção

por exemplo → s' = (6,10,8,9,2,7,3,4,5,1)
ou s' = [(6), (10), (8), (9), (2), (7), (3,4,5), (1)]

por exemplo → s' = (10,6,2,8,9,7,3,4,5,1)
ou s' = [(10), (6,2,8), (9,7), (3,4,5), (1)]

PARTES DE	FP	MAGAZINE TIPOS DE FERRAMENTAS				Nº DE Sf
1	5	1	4	8	9	-
10	4	1	2	4	8	1
3 , 4 , 5	2	2	6	7	8	2
6 , 8 , 2	1	1	3	5	8	3
7 , 9	3	1	5	7	9	2

Nº de Sp = 4 Nº de Sf = 8

PARTES DE	FP	MAGAZINE TIPOS DE FERRAMENTAS				Nº Sf
1	5	1	4	8	9	-
10	4	1	2	4	8	1
3 , 4 , 5	2	2	6	7	8	2
6 , 8	1	3	5	7	8	2
2	6	1	3	5	7	1
7 , 9	3	1	5	7	9	1

Nº de Sp = 5 Nº de Sf = 7

Mesma de seqüência de partes:

s = (1,10,3,4,5,6,8,2,7,9)

→ 4 instantes de parada para troca de ferramentas com 8 trocas de ferramentas
→ 5 instantes de parada para troca de ferramentas com 7 trocas de ferramentas

ALGORITMO GENÉTICO CONSTRUTIVO

PROBLEMA DAS *P*-MEDIANAS

Problema clássico de localização.

→ objetivo: localizar *p* facilidades (medianas), de forma a minimizar a soma das distâncias de cada vértice a sua facilidade mais próxima.

Seja m_{ij} (≥0) o custo de atribuir o vértice *j* ao vértice *i*,
V o conjunto de vértices.

x_{ij} = 1 se o vértice *j* é atribuído ao vértice *i*,
x_{ij} = 0 caso contrário.

$$Min \quad z = \sum_{i \in V} \sum_{j \in V} m_{ij} x_{ij}, \tag{1}$$

sujeito a

$$\sum_{i \in V} x_{ij} = 1 \quad " j \hat{I} V \tag{2}$$

$$\sum_{i \in V} x_{ii} = p, \tag{3}$$

$$x_{ij} \leq x_{ii}; i, j \in V \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V, j \in V. \tag{5}$$

(2) e (4) asseguram que cada vértice *j* seja atribuída somente a um vértice *i*, que necessariamente deve ser uma mediana.
(3) determina o número exato de medianas que devem ser atribuídas a *p*.

Algoritmo Genético

AG {Algoritmo Genético}
t := 0 ;
Inicializar P_t ; {população inicial}
Avaliar P_t ;
Enquanto (não satisfeita condição de parada) **fazer**
    t := t + 1 ;
    Selecionar P_t **de** P_{t-1} ; {operador reprodução}
    Recombinar P_t ; {operador "crossover" e mutação}
    Avaliar P_t ; {avaliar adaptação}
Fim_Enquanto

→ Conjunto de variáveis denominadas *estruturas* . Na sua forma mais simples, a estrutura é representada por uma cadeia ("string") de dados elementares, possivelmente uma cadeia binária;

→ Estruturas combinam formando uma *população*.

→ Cada estrutura é avaliada através de uma *função de adaptação*.

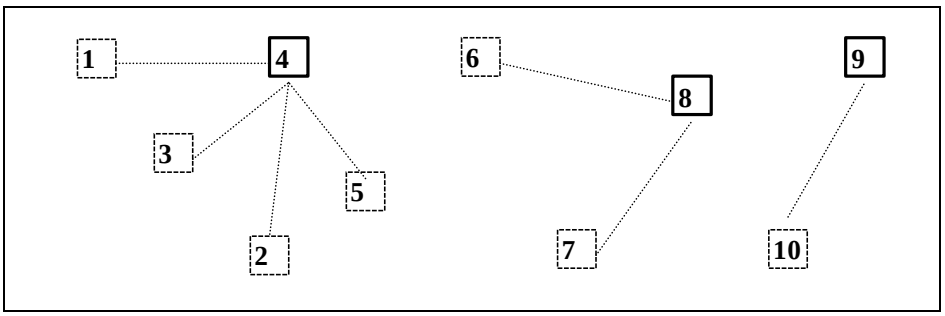
→ O principal operador no algoritmo genético é a *recombinação*. Na recombinação, duas estruturas da população são escolhidas, através de um critério de *seleção*, para produzir duas novas estruturas.

→ *"crossover"* quando escolhe um ou mais pontos nas estruturas pais, de tal forma que exista a mistura de ambas partes nas estruturas filhas. Cada elemento da cadeia, que forma a estrutura, pode ainda ser modificada, o que é denominado *mutação*.

Exemplo:

$$S_i = (2\ 2\ 2\ 1\ 2\ 2\ 2\ 1\ 1\ 2)$$

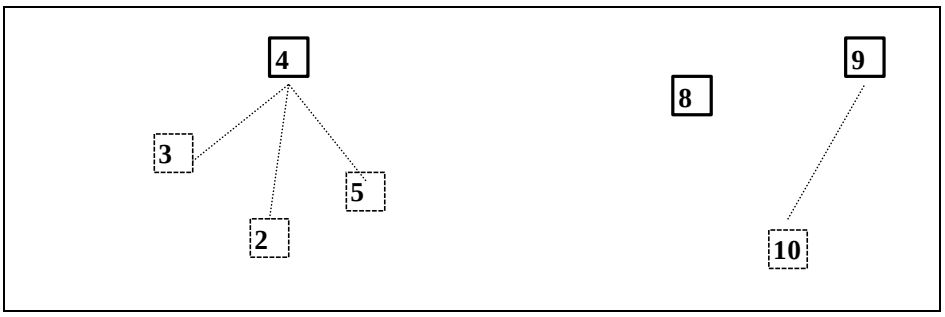
- V_1 : conjunto de vértices que representam as medianas $\rightarrow V_1 = \{4.8.9\}$
- V_2 : conjunto de vértices que não são medianas $\rightarrow V_2 = \{1,2,3,5,6,710\}$

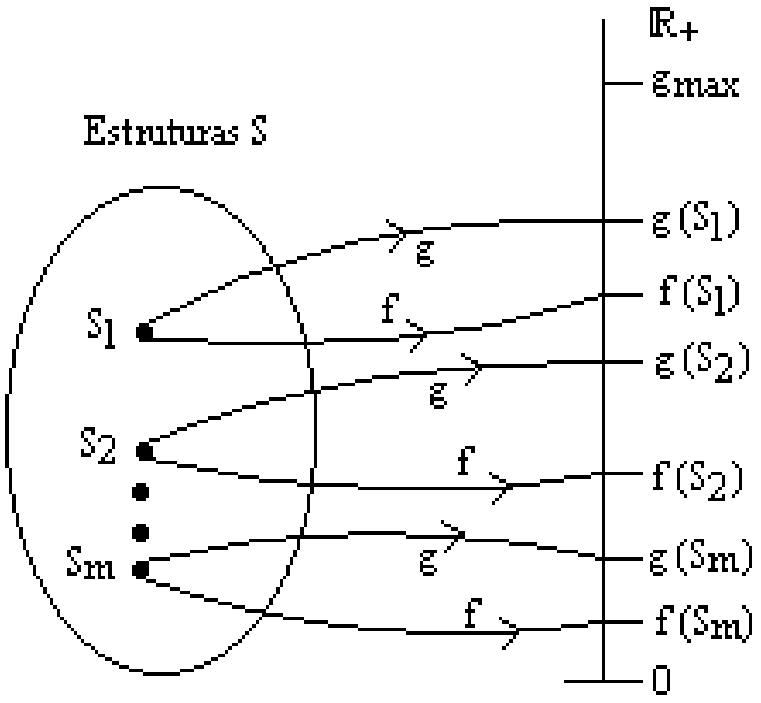


Algoritmo Genético Construtivo

População de esquemas:

$$S_i = (\#\ 2\ 2\ 1\ 2\ \#\ \#\ 1\ 1\ 2)$$





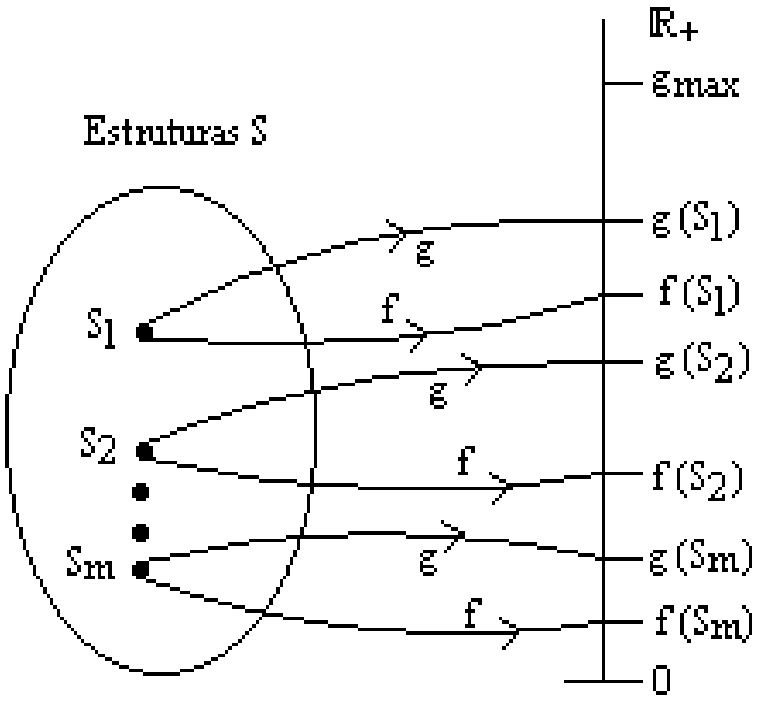
→ Baseado no Algoritmo A* ;

$$\rightarrow d_i = \frac{g(s_i) - f(s_i)}{g(s_i)}, i = 1, \ldots, m.$$

$$\rightarrow d\ g_{max} \leq d_i\ g(s_i) + a\ d\ (g_{max} - g(s_i))$$

$$\rightarrow a \geq \frac{dg_{max} - d_i g(s_i)}{d(g_{max} - g(s_i))} = d(s_i)$$

- V_1 : conjunto de vértices que representam as medianas;
- V_2 : conjunto de vértices que não são medianas;
- V_3 **:** conjunto de vértices que ainda não estão representados na estrutura.



$\rightarrow g(s_k) = \sum_{j \in V_2} \min_{i \in V_1} \{m_{ij}\}$;

$\rightarrow$ Sejam $l_j = \min_{i \in V_2} m_{ij}, j \in V_1$ e seja C_j o conjunto formado pela mediana *j* e seus vértices associados. $|C_j|$ é o número de elementos do conjunto C_j. Então:

$\rightarrow f(s_k) = \sum_{j=1}^{p} l_j |C_j|$.

Operador seleção

$\rightarrow$ População ordenada pela equação:

$\rightarrow \Delta(s_k) = \frac{1 + d_k}{V_1 + V_2}$.

Operador 1: As estruturas s_i e s_j são selecionadas na população inteira;
Operador 2: A estrutura s_i é selecionada nas k_1 melhores estruturas e s_j é selecionada na população inteira;
Operador 3: As estruturas s_i e s_j são selecionadas nas k_1 melhores estruturas;

Operador 4: A estrutura s_i é selecionada nas k_2% melhores estruturas e s_j é selecionada nas k_1 melhores estruturas;
Operador 5: As estruturas s_i e s_j são selecionadas nas k_2% melhores estruturas, onde para o problema das *p*-medianas k_1 é igual ao número de vértices do problema e k_2 é igual a 30% das melhores estruturas.

Operador recombinação

Caso 1:
base: (1 2 0 2 1 0 1 0)
guia: (0 2 1 1 1 0 2 2)
resultante: (_ 2 _ _ 1 0 _ _)

Caso 2:
base: (1 2 0 2 1 2 1 0)
guia: (0 2 1 1 1 0 2 2)
resultante: (1 _ _ _ _ 2 _ _)

Caso 3:
base: (1 2 0 2 1 2 1 0)
guia: (0 2 1 1 1 0 2 2)
resultante: (_ _ _ _ _ _ _ 2)

Caso 4:
base: (1 2 0 2 1 0 1 0)
guia: (0 2 1 1 1 0 2 2)
resultantes: (2 2 1 2 1 0 1 0)
(1 2 1 2 2 0 1 0)
(1 2 1 2 1 0 2 0)

Caso 5:
base: (1 2 0 2 1 0 1 0)
guia: (0 2 1 1 1 0 2 2)
resultantes: (1 1 0 2 1 0 2 0)
(1 2 0 1 1 0 2 0)

Operador mutação

→ Apenas uma estrutura é selecionada (operadores 4 e 5). Cada mediana da estrutura permuta de posição com os vértices que não pertencem ao conjunto das medianas, gerando novas estruturas ($|V|-|V_1|$ estruturas).

Resultados Computacionais

Problema	Vértices	Medianas	Solução ótima	Função objetivo	Diferença %	Tempo segundos
pmed1	**100**	**5**	**5819**	**5819**	**0**	**28**
pmed2	**100**	**10**	**4093**	**4093**	**0**	**37**
pmed3	**100**	**10**	**4250**	**4250**	**0**	**34**
pmed4	**100**	**20**	**3034**	**3034**	**0**	**230**
pmed5	**100**	**33**	**1355**	**1360**	**0.36**	**375**
pmed6	**200**	**5**	**7824**	**7824**	**0**	**172**
pmed7	**200**	**10**	**5631**	**5631**	**0**	**238**
pmed8	**200**	**20**	**4445**	**4454**	**0.20**	**1055**
pmed9	**200**	**40**	**2734**	**2754**	**0.73**	**3331**
pmed10	**200**	**67**	**1255**	**1257**	**0.15**	**4325**
pmed11	**300**	**5**	**7696**	**7696**	**0**	**369**
pmed12	**300**	**10**	**6634**	**6637**	**0.04**	**677**
pmed16	**400**	**5**	**8162**	**8162**	**0**	**555**
pmed21	**500**	**5**	**9138**	**9138**	**0**	**1875**

Tabela 1: Resultados computacionais dos testes realizados.

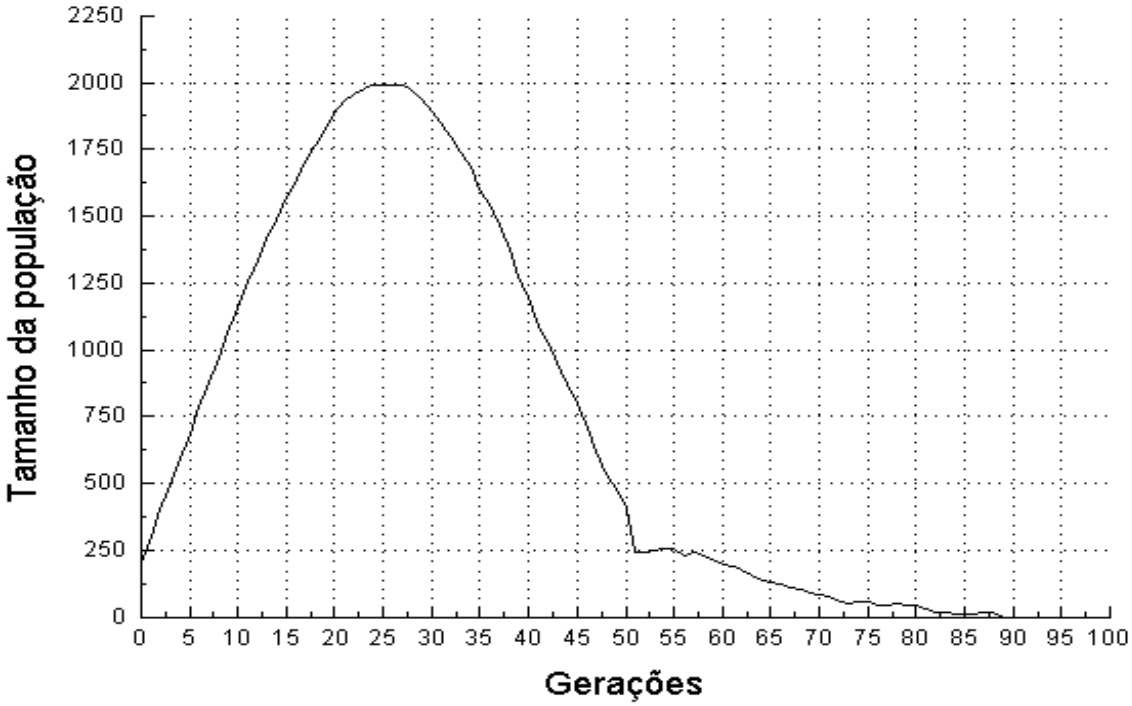


Gráfico 1: Tamanho da população de estruturas com o decorrer das gerações para o problema pmed1.

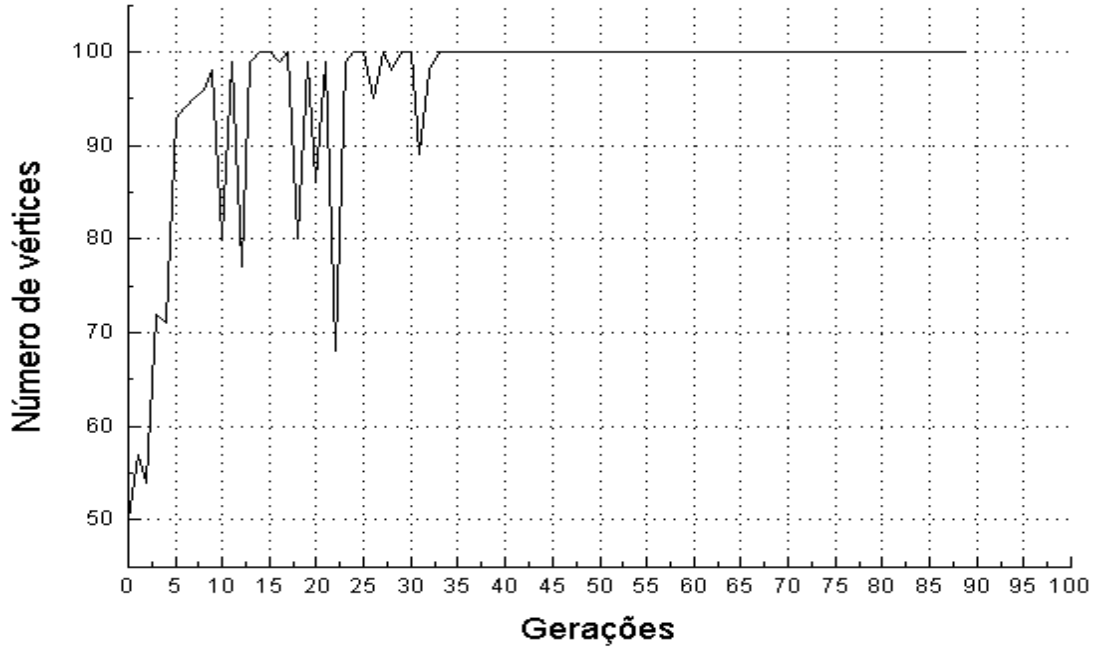


Gráfico 2: Número de vértices que participam da melhor estrutura gerada a cada geração para o problema pmed1.

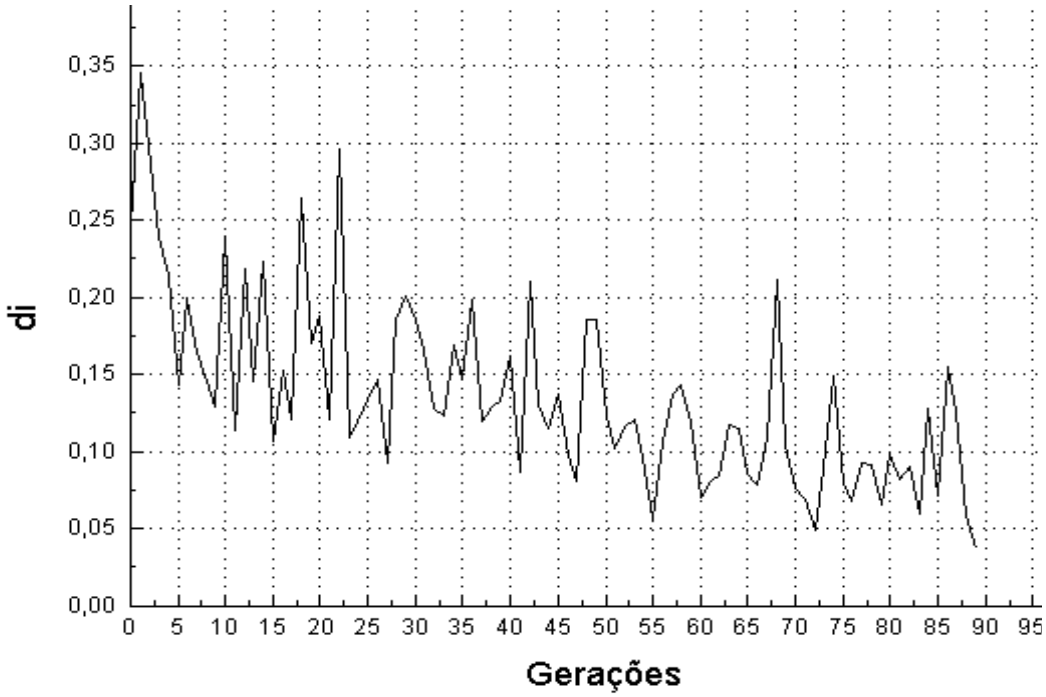


Gráfico 3: Gráfico de d_i em função do número de gerações, para o problema pmed1.

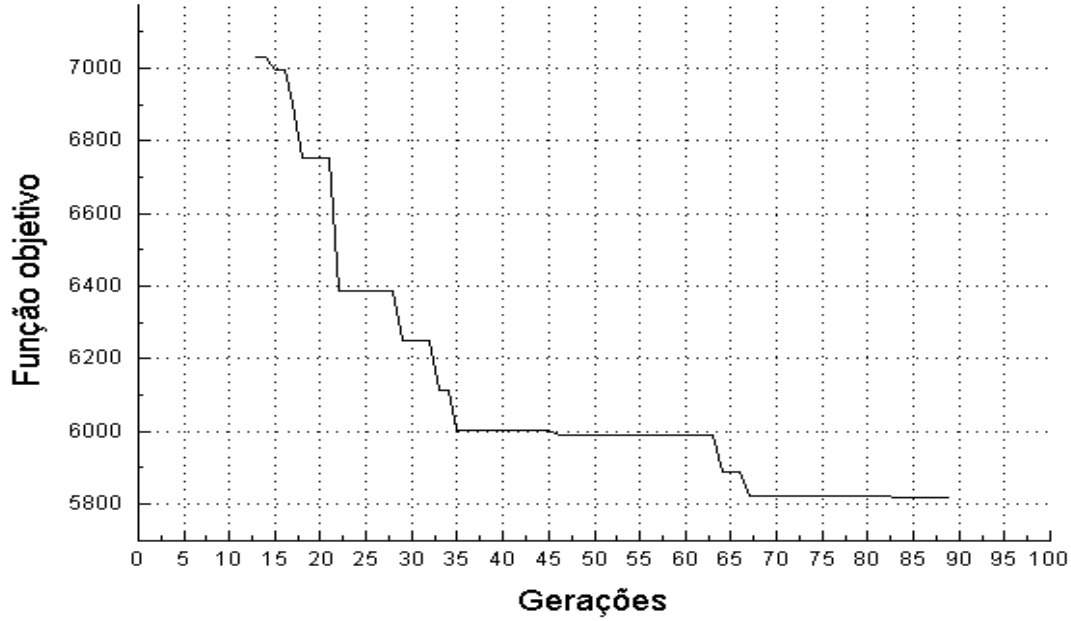
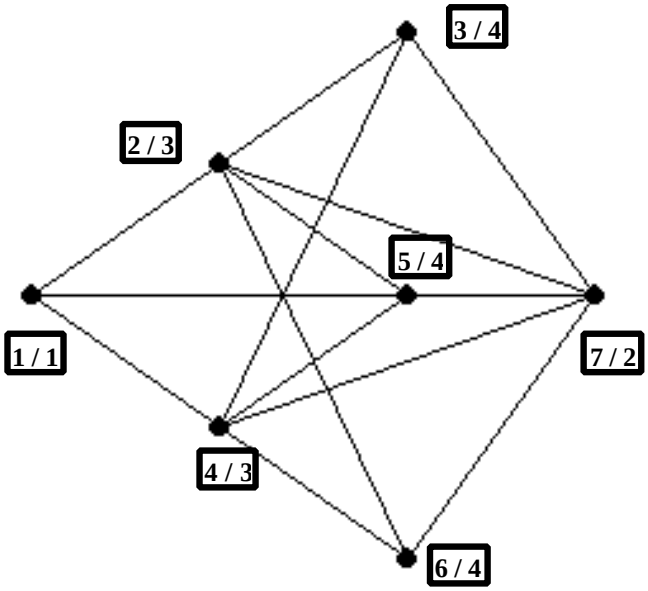
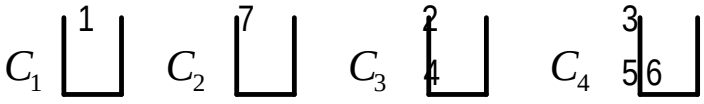


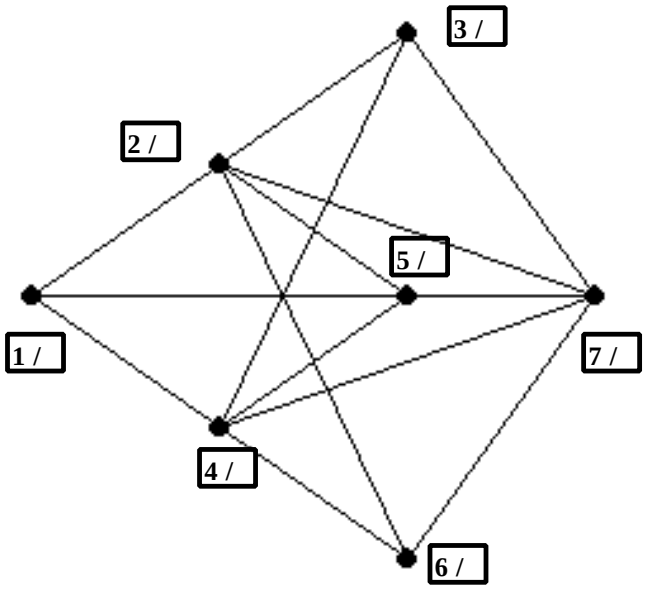
Gráfico 4: Função objetivo em função das gerações para o problema pmed1.

Coloração de Grafos

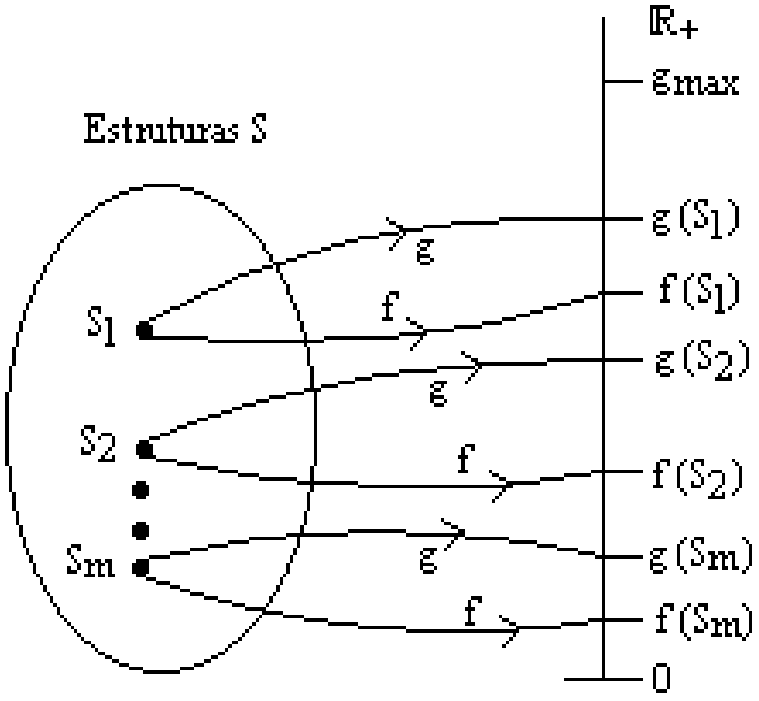


Estrutura: s = (1,3,4,3,4,4,2)





Esquema: s = (1,#,4,3,#,#,2)



$$\rightarrow f(s) = \sum_{p=1}^{k} \left\{ \left[\frac{\left(\left|C_p\right| - 1\right).\left|C_p\right|}{2} \right] - \left|E\left(C_p\right)\right| \right\},$$

$$\rightarrow g(s) = \sum_{p=1}^{k} \left[\frac{\left(\left|C_p\right| - 1\right).\left|C_p\right|}{2} \right],$$

$$\rightarrow g_{\max} = mult.k.\left[\frac{\left(\left\lfloor \frac{m}{k} \right\rfloor - 1\right).\left\lfloor \frac{m}{k} \right\rfloor}{2} \right],$$

$$\rightarrow \Delta(s) = \frac{1 + \sum_{p=1}^{k} \left[\left|E\left(C_p\right)\right|\right]}{nv}.$$

Graph Coloring Instances

Each instance includes the information: (nodes, edges), optimal coloring, source.

DSJC1000.1.col.b (1000,99258), ?, DSJ
DSJC1000.5.col.b (1000,499652), ?, DSJ
DSJC1000.9.col.b (1000,898898), ?, DSJ
DSJC125.1.col.b (125,1472), ?, DSJ
DSJC125.5.col.b (125,7782), ?, DSJ
DSJC125.9.col.b (125,13922), ?, DSJ
DSJC250.1.col.b (250,6436), ?, DSJ
DSJC250.5.col.b (250,31366), ?, DSJ
DSJC250.9.col.b (250,55794), ?, DSJ
DSJC500.1.col.b (500,24916), ?, DSJ
DSJC500.5.col.b (500,125249), ?, DSJ
DSJC500.9.col.b (500,224874), ?, DSJ
DSJR500.1.col.b (500,7110), ?, DSJ
DSJR500.1c.col.b (500,242550), ?, DSJ
DSJR500.5.col.b (500, 117724), ?, DSJ
flat1000_50_0.col.b (1000,245000), 50, CUL
flat1000_60_0.col.b (1000,245830), 60, CUL
flat1000_76_0.col.b (1000,246708), 76, CUL
flat300_20_0.col.b (300,21375), 20, CUL
flat300_26_0.col.b (300, 21633), 26, CUL
flat300_28_0.col.b (300, 21695), 28, CUL
fpsol2.i.1.col (496,11654), 65, REG
fpsol2.i.2.col (451,8691), 30, REG
fpsol2.i.3.col (425,8688), 30, REG
inithx.i.1.col (864,18707), 54, REG
inithx.i.2.col (645, 13979), 31, REG
inithx.i.3.col (621,13969), 31, REG
latin_square_10.col (900,307350), ?,
e450_15a.col (450,8168), 15, LEI
le450_15b.col (450,8169), 15, LEI
le450_15c.col (450,16680), 15, LEI
le450_15d.col (450,16750), 15, LEI
le450_25a.col (450,8260), 25, LEI
le450_25b.col (450,8263), 25, LEI
le450_25c.col (450,17343), 25, LEI
le450_25d.col (450,17425), 25, LEI
le450_5a.col (450,5714), 5, LEI
le450_5b.col (450,5734), 5, LEI
le450_5c.col (450,9803), 5, LEI
le450_5d.col (450,9757), 5, LEI
mulsol.i.1.col (197,3925), 49, REG
mulsol.i.2.col (188,3885), 31, REG
mulsol.i.3.col (184,3916), 31, REG
mulsol.i.4.col (185,3946), 31, REG
mulsol.i.5.col (186,3973), 31, REG
school1.col (385,19095), ?, SCH
school1_nsh.col (352,14612), ?, SCH
zeroin.i.1.col (211,4100), 49, REG
zeroin.i.2.col (211, 3541), 30, REG
zeroin.i.3.col (206, 3540), 30, REG
anna.col (138,493), 11, SGB
david.col (87,406), 11, SGB
homer.col (561,1629), 13, SGB
huck.col (74,301), 11, SGB
jean.col (80,254), 10, SGB
games120.col (120,638), 9, SGB
miles1000.col (128,3216), 42, SGB
miles1500.col (128,5198), 73, SGB
miles250.col (128,387), 8, SGB
miles500.col (128,1170), 20, SGB
miles750.col (128,2113), 31, SGB
queen10_10.col (100,2940), ?, SGB
queen11_11.col (121,3960), 11, SGB
queen12_12.col (144,5192), ?, SGB
queen13_13.col (169,6656), 13, SGB
queen14_14.col (196,8372), ?, SGB
queen15_15.col (225,10360), ?, SGB
queen16_16.col (256,12640), ?, SGB
queen5_5.col (25,160), 5, SGB
queen6_6.col (36,290), 7, SGB
queen7_7.col (49,476), 7, SGB
queen8_12.col (96,1368), 12, SGB
queen8_8.col (64, 728), 9, SGB
queen9_9.col (81, 2112), 10, SGB
myciel3.col (11,20), 4, MYC
myciel4.col (23,71), 5, MYC
myciel5.col (47,236), 6, MYC
myciel6.col (95,755), 7, MYC
myciel7.col (191,2360), 8, MYC

Notes:

DSJ: (From David Johnson (dsj@research.att.com)) Random graphs used in his paper with Aragon, McGeoch, and Schevon, ``Optimization by Simulated Annelaing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning", Operations Research, 31, 378--406 (1991). DSJC are standard (n,p) random graphs. DSJR are geometric graphs, with DSJR..c being complements of geometric graphs.

CUL: (From Joe Culberson (joe@cs.ualberta.ca)) Quasi-random coloring problem.

REG: (From Gary Lewandowski (gary@cs.wisc.edu)) Problem based on register allocation for variables in real codes.

LEI: (From Craig Morgenstern (morgenst@riogrande.cs.tcu.edu)) Leighton graphs with guaranteed coloring size. A reference is F.T. Leighton, Journal of Research of the National Bureau of Standards, 84: 489—505 (1979).

SCH: (From Gary Lewandowski (lewandow@cs.wisc.edu))Class scheduling graphs, with and without study halls.

LAT: (From Gary Lewandowski (lewandow@cs.wisc.edu)) Latin square problem.

SGB: (From Michael Trick (trick@cmu.edu)) Graphs from Donald Knuth's Stanford GraphBase. These can be divided into:

Book Graphs. Given a work of literature, a graph is created where each node represents a character. Two nodes are connected by an edge if the corresponding characters encounter each other in the book. Knuth creates the graphs for five classic works: Tolstoy's Anna Karenina (anna), Dicken's David Copperfield (david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck), and Hugo's Les Misérables (jean).

Game Graphs. A graph representing the games played in a college football season can be represented by a graph where the nodes represent each college team. Two teams are connected by an edge if they played each other during the season. Knuth gives the graph for the 1990 college football season.

Miles Graphs. These graphs are similar to geometric graphs in that nodes are placed in space with two nodes connected if they are close enough. These graphs, however, are not random. The nodes represent a set of United States cities and the distance between them is given by by road mileage from 1947. These graphs are also due to Kuth.

Queen Graphs. Given an n by n chessboard, a queen graph is a graph on n^2 nodes, each corresponding to a square of the board. Two nodes are connected by an edge if the corresponding squares are in the same row, column, or diagonal. Unlike some of the other graphs, the coloring problem on this graph has a natural interpretation: Given such a chessboard, is it possible to place n sets of n queens on the board so that no two queens of the same set are in the same row, column, or

diagonal? The answer is yes if and only if the graph has coloring number n. Martin Gardner states without proof that this is the case if and only if n is not divisible by either 2 or 3. In all cases, the maximum clique in the graph is no more than n, and the coloring value is no less than n.

MYC: (From Michael Trick (trick@cmu.edu)) Graphs based on the Mycielski transformation. These graphs are difficult to solve because they are triangle free (clique number 2) but the coloring number increases in problem size.