CAPÍTULO 5

ALGORITMO EXATO

Um problema de otimização combinatória pode ser definido matematicamente a partir do par (Ω,C), onde Ω é o conjunto (finito) de soluções possíveis e C é uma função $C : \Omega \rightarrow \Re$, sendo $\Re$ o conjunto dos reais. O objetivo, no problema de otimização de minimização, é encontrar uma solução ótima $S \in \Omega$, tal que $C(S) \leq C(S')$, $\forall S' \in \Omega$; ou seja, tal que a função C, conhecida como função custo do problema, seja minimizada. Como Ω é finito, intuitivamente poderíamos, selecionar a solução ótima por enumeração. Essa proposta de enumeração exaustiva funciona teoricamente, mas torna-se inviável para os problemas de interesse, devido ao tamanho extremamente grande – apesar de finito – do conjunto de soluções. Cada pequeno incremento na dimensão do problema introduz uma gama muito grande de possibilidades, configurando uma explosão combinatória de soluções possíveis, e uma conseqüente explosão no tempo necessário para encontrar a solução ótima (Bjorndal et al. 1995).

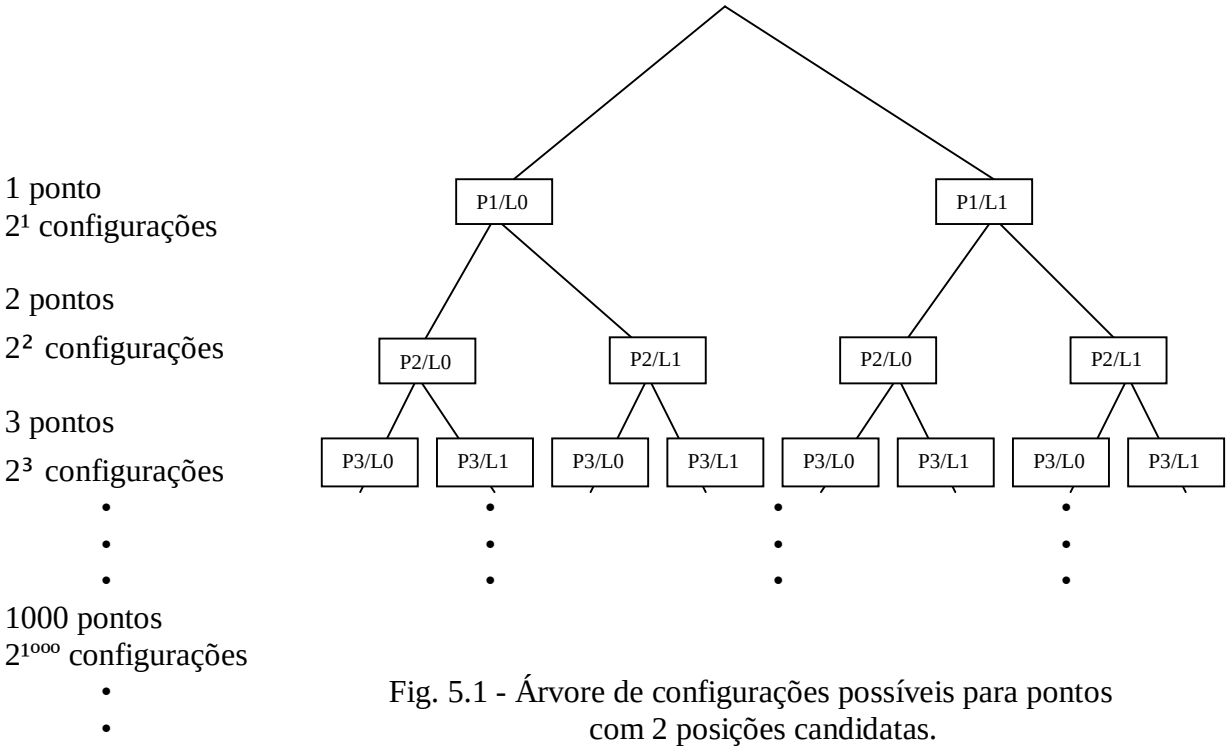

Fig. 5.1 - Árvore de configurações possíveis para pontos com 2 posições candidatas.

Em rotulação cartográfica se considerarmos 2 posições candidatas – L0 e L1 – para cada ponto pi, onde i varia entre 1 e o número de pontos, e desenvolvermos uma árvore de configurações veremos que o número de configurações possíveis aumenta exponencialmente dependendo do número de pontos do leiaute (Figura 5.1) .

Esta explosão combinatória mostra que se trata de um problema de otimização combinatória e neste caso a enumeração completa de todas as soluções viáveis de um problema conduzirá a solução ótima em um número finito de passos. Infelizmente mostra-se que, mesmo para problemas de porte pequeno, o número de passos será tão grande que o computador mais avançado levaria séculos no processo de cálculo. Campello R. E. e Maculan N., 1994 apresentam um exemplo de um problema cujo número de soluções viáveis é (n!). A Tabela 5.1 mostra a explosão exponencial, ou seja, a grande variação observada no tempo de execução, quando o valor de n aumenta de apenas uma unidade, levando em consideração que cada solução possa ser examinada em 10**(-9) segundos (1 nanosegundo).

TABELA 5.1 – EXPLOSÃO EXPONENCIAL

n	Tempo de Execução
20	80 anos
21	1680 anos

Neste caso muitos pesquisadores não acreditam na existência de um algoritmo exato eficiente e, mesmo que esse algoritmo fosse encontrado, ele seria bastante lento para problemas grandes, assim sendo existe a necessidade de introduzir heurísticas tais como o CGA e TS, que não buscam a solução exata, mas uma solução suficientemente boa em termos de custo.

Neste Capítulo está sendo proposto um algoritmo exato, com o objetivo de obtermos soluções ótimas para verificar a qualidade das soluções geradas pelo CGA e TS.

Basicamente o algoritmo exato trabalha com busca em profundidade em uma árvore, mas com possibilidades de corte de ramificações da árvore em algumas situações, pois como foi visto anteriormente a enumeração total causa uma explosão no tempo de execução, uma vez que o número de passos necessários, para garantir a obtenção da solução ótima cresce exponencialmente com o número de variáveis, digamos 2^n , onde no caso da rotulação de pontos, n é o número de variáveis ou pontos e 2 é o número de posições candidatas do ponto. Na seção 5.2 será descrito com mais detalhes o algoritmo exato proposto neste trabalho. A próxima seção faz uma breve descrição do algoritmo exato para conjunto independente de vértices.

5.1 ALGORITMO EXATO PARA CONJUNTO INDEPENDENTE DE VÉRTICES

Dowsland (1987) aplica o algoritmo exato, para o máximo conjunto estável de Loukakis e Tsouros (1982), ao problema de empacotamento bi-dimensional. O objetivo é encontrar um leiaute ótimo, para retângulos menores do mesmo tamanho, dispostos em um retângulo maior. O problema possui o grafo G(V, E), onde $V = \{v_1, v_2, ..., v_n\}$ representa o conjunto de posições possíveis dos retângulos menores, dentro do retângulo maior, e $E = \{e_1, e_2, ..., e_m\}$ é o conjunto de arestas formadas por dois retângulos que se sobrepõem.

Trata-se de um algoritmo de busca em árvore, onde os nós pertencem a um dos três conjuntos: S — lista de vértices da solução, S_+ — lista de vértices disponíveis que podem fazer parte de S, e S- — lista de vértices descartados que não podem fazer parte de S. Com o objetivo de diminuir o tamanho da busca, as seguintes restrições são impostas:

Se $|S| + |S_+| < N_{ub}$ **então** "backtracking"

Se $|S| = N_{ub}$ **então** parar

Onde N_{ub} é o limite superior, ou seja, é o maior número de retângulos menores que podem ser colocados no retângulo maior, sem causar sobreposição. Este valor foi

estimado pelo autor. O “backtracking” refere-se a voltar um nível da árvore e escolher outras opções.

5.2 DESCRIÇÃO DO ALGORITMO EXATO PARA ROTULAÇÃO DE PONTOS

O algoritmo exato descrito a seguir, aproveitou a idéia principal de Dowsland (1987), que aplica cortes nas ramificações da árvore, em um algoritmo de busca em árvore.

O algoritmo exato proposto coloca então, o rótulo do ponto selecionado na primeira posição candidata, até alcançar o número permitido de rótulos em conflito. Um corte de ramificação da árvore é executado, quando alcançar o limite permitido de rótulos em conflito. Neste caso, as sucessivas posições candidatas do ponto em questão são testadas para se executar um novo corte ou a continuação da busca em profundidade. Se um determinado ponto não puder ser rotulado porque não existem posições sem conflito, o algoritmo volta para o ponto rotulado anteriormente e muda a posição do rótulo. E assim o algoritmo continua até que todos os pontos tenham sido rotulados ou até que todas as soluções tenham sido verificadas, retendo neste caso a melhor solução.

A seguir será descrito o algoritmo exato para um problema de n pontos que possui o grafo G(V,E), onde V = {L1, L2, ..., L_n} representa o conjunto de rótulos e E = {A1, A2, ..., A_m} representa o conjunto de arestas formados por 2 rótulos que não podem estar ativos simultaneamente por causar conflito ou porque os 2 rótulos pertencem ao mesmo ponto. O grafo G(V,E), o número de pontos, a previsão do número de rótulos sem conflitos (M) e grau dos vértices são dados de entrada. O valor de M usado por nós para o corte das ramificações da árvore é dado levando em consideração o valor obtido da aplicação do CGA ou TS ao problema em questão, mas este valor pode ser obtido usando um algoritmo simples de busca exaustiva, algoritmo guloso ou qualquer outro algoritmo que consiga resolver o problema de rotulação de pontos, ou ainda estabelecer um valor inicial pequeno de acordo com a sua intuição como um limitante inferior, pois o valor M é atualizado toda vez que um valor maior de número de rótulos sem conflitos for encontrado.

Algoritmo exato

Início:

S := ∅ conjunto de rótulos ativos (solução)

V lista de todos os rótulos

M número de rótulos sem conflito (resultado CGA ou TS)

P pontos em ordem crescente por grau dos vértices

Enquanto $|S| \neq n$

- **Escolher** de V, rótulo v_i do ponto Pi e **acrescentar** em S
- **Calcular** nc (número de rótulos com conflito de S)
- **Calcular** nsc (número de rótulos sem conflito de S)
- **Se** nsc > M **então** M := nsc
- **Verificar** futuro (número de pontos ainda não rotulados, que possuem posições candidatas que não conflitam com S)
- **Se** (nsc + futuro ≤ M) ou (nc > n - M) **executar** corte

FIM_Enquanto

O algoritmo exato descrito acima envolve alguns conceitos: ordenação dos pontos, escolha de rótulos, verifica futuro e corte de ramificações da árvore. Eles serão descritos a seguir.

Ordenação de pontos: os pontos são ordenados de acordo com o grau dos vértices que o compõe, ou seja, a seqüência de escolha dos pontos depende do número de conflitos que as posições candidatas do ponto apresentam. Assim sendo, o ponto escolhido será aquele que possui a posição candidata, que apresenta o menor número de conflitos, entre todas as posições candidatas, de todos os pontos do cenário.

Usando o exemplo de 6 pontos, temos que a ordem dos pontos é P6, P1, P4, P3, P5 e P2 como mostrado na tabela 5.2. Como pode ser visto, uma vez escolhido o ponto, o restante das posições candidatas do ponto em questão, não são consideradas, mesmo

apresentando o segundo menor número de conflitos entre todos os rótulos do cenário, como por exemplo o L24 do ponto P6 e L01 do ponto P1.

TABELA 5.2 – ORDENAÇÃO DE PONTOS

Pontos	vértices (rótulos)	Grau dos vértices	Ordem crescente
P1	L01	5	
P1	L02	**4**	2
P1	L03	8	
P1	L04	10	
P2	L05	**8**	6
P2	L06	10	
P2	L07	13	
P2	L08	11	
P3	L09	**7**	4
P3	L10	11	
P3	L11	11	
P3	L12	7	
P4	L13	9	
P4	L14	9	
P4	L15	7	
P4	L16	**5**	3
P5	L17	15	
P5	L18	9	
P5	L19	**7**	5
P5	L20	11	
P6	L21	5	
P6	L22	4	
P6	L23	**3**	1
P6	L24	3	

Outro detalhe a ser considerado é o fato de ficar com o primeiro rótulo que apresenta o menor número de conflitos, quando existir mais de um, como pode ser visto com L09 do ponto P3 e L19 do ponto P5, ambos rótulos apresentam 7 conflitos, mas o ponto P3 teve a preferência.

O objetivo de se escolher vértices de menor grau ou posições candidatas que apresentam menor número de conflitos, vem do fato de que quanto menos conflitos apresentarem, a probabilidade de conflitos com os rótulos ativos de outros pontos é menor.

Optou-se por ordenar a seqüência dos pontos a serem rotulados, com o objetivo de se encontrar a solução boa ou a solução ótima logo no início, pois isto aumenta o valor de M (no início é um valor obtido via leitura, que representa um limitante inferior do número de rótulos sem conflito) e o corte das ramificações da árvore acontecem em um nível não muito profundo, diminuindo assim o tempo de processamento, que em algoritmos exatos é bastante grande.

Escolha de rótulos: os pontos são rotulados na seqüência em que foram ordenados, e o rótulo de cada ponto, é escolhido seqüencialmente por ordem crescente de custo, dentre as posições candidatas de cada ponto. Se considerarmos o exemplo de 4 posições candidatas da Figura 2.5, a posição candidata a se tornar ativa será na seguinte ordem: L0, L1, L2 e L3.

O algoritmo faz primeiro a busca em profundidade e depois a busca por largura. Tenta-se sempre rotular cada ponto na ordem estabelecida com a posição candidata L0, mas quando o número de conflitos ultrapassa o limite estabelecido, ou o número de rótulos sem conflito atual, somado ao número de rótulos sem conflito que pode existir no futuro, ultrapassa o limite estabelecido, a próxima posição candidata, no caso L1 é escolhida. Novamente o teste é aplicado e caso seja verdadeiro em um dos casos a próxima posição candidata L2 é escolhida, mas se as duas questões forem falsas, o próximo ponto é rotulado na posição L0. O algoritmo continua neste processo, até que um determinado ponto não pode ser rotulado porque não existem posições sem conflito. Neste caso ele executa o “backtracking”, ou seja, o algoritmo volta para o ponto rotulado anteriormente e muda para a próxima posição candidata. E assim o algoritmo continua até que todas as soluções possíveis tenham sido verificadas.

Verifica futuro: Para verificar se o número de rótulos sem conflito atual, somado ao número de rótulos sem conflito que pode existir no futuro, ultrapassa o limite estabelecido, é necessário fazer uma previsão do futuro quanto ao número de rótulos sem conflito que teremos, levando em consideração os pontos que ainda não foram rotulados. Esta previsão é feita, verificando se pelo menos uma posição candidata dos pontos que ainda não foram rotulados, não tem conflito com os rótulos ativos dos pontos já rotulados.

Corte de ramificações da árvore: este termo diz respeito ao corte das ramificações de um determinado nível de uma árvore, ou seja, todas as ramificações que vierem abaixo do nível que sofreu o corte, não precisarão ser analisados. O problema de rotulação de pontos pode ser visualizado em forma de uma árvore através da Figura 5.1, que mostra uma árvore de configurações possíveis para pontos com duas posições candidatas. Se a árvore do exemplo sofrer corte em uma das ramificações do nível dos 2**900 ramificações, estamos deixando de analisar 2**100 configurações, o que equivale a ganho em tempo de processamento. Este tipo de técnica se torna bastante atraente nos algoritmos exatos, se conseguir fazer corte em níveis bem acima dos níveis terminais, pois reduz bastante o número de configurações que precisa ser analisado, e por conseguinte diminui o tempo de processamento.

No algoritmo exato implementado por nós, o corte acontece quando existe a necessidade de troca da posição do rótulo por causa dos conflitos. Neste caso, todas as configurações possíveis de posições de rótulo, que possam ter os demais pontos que o sucedem, perdem a atratividade de análise.

5.3 EXEMPLO DE APLICAÇÃO DO ALGORITMO EXATO PARA 6 PONTOS

Será descrito em detalhes a aplicação do algoritmo exato implementado por nós, utilizando um exemplo de 6 pontos com 4 posições candidatas, cuja matriz de adjacência inicial e o grau dos vértices se encontram descritos na Figura3.6. Para o exemplo, a estimativa inicial do número de rótulos sem conflito foi estabelecido como

sendo M = 5 e a seqüência em que os pontos foram rotulados foi: P6, P1, P4, P3, P5 e P2.

O exemplo do algoritmo exato para 6 pontos está mostrado na Tabela 5.3, onde as linhas se referem à seqüência de passos seguidos para alcançar a solução, e as colunas se referem a:

- Ponto: pontos sendo rotulados
- Solução: rótulo que está sendo atribuído ao ponto
- nc: número de rótulos com conflito de S
- nsc: número de rótulos sem conflito de S
- futuro: número de pontos ainda não rotulados, que possuem posições candidatas que não conflitam com S
- previsão: nsc + futuro (mostra a tendência, de quanto será o número de rótulos sem conflito que pode se obter, se continuarmos analisando todas as ramificações da posição corrente da árvore até o fim)

TABELA 5.3 – EXEMPLO DO ALGORITMO EXATO PARA 6 PONTOS

ponto	solução	nc	nsc	futuro	previsão
P6	L21	0	1	5	6
P1	L01	0	2	4	6
P4	L13	0	3	2	5
P4	L14	0	3	2	5
P4	L15	2	1	3	4
P4	L16	2	1	3	4
P1	L02	0	2	4	6
P4	L13	0	3	3	6
P3	L09	0	4	1	5
P3	L10	0	4	1	5
P3	L11	2	2	1	3
P3	L12	2	2	2	4

P4	L14	0	3	2	5
P4	L15	2	1	3	4
P4	L16	2	1	3	4
P1	L03	0	2	4	6
P4	L13	0	3	3	6
P3	L09	0	4	1	5
P3	L10	2	2	1	3
P3	L11	2	2	1	3
P3	L12	2	2	2	4
P4	L14	0	3	2	5
P4	L15	2	1	2	3
P4	L16	2	1	3	4
P1	L04	0	2	3	5
P6	L22	0	1	5	6
P1	L01	0	2	4	6
P4	L13	0	3	2	5
P4	L14	0	3	2	5
P4	L15	2	1	3	4
P4	L16	0	3	3	6
P3	L09	0	4	2	6
P5	L17	2	3	0	3
P5	L18	0	5	0	5
P5	L19	0	5	0	5
P5	L20	2	3	0	3
P3	L10	0	4	1	5
P3	L11	0	4	0	4
P3	L12	0	4	2	6
P5	L17	2	3	0	3
P5	L18	0	5	0	5
P5	L19	0	5	0	5
P5	L20	3	2	0	2
P1	L02	0	2	4	6
P4	L13	0	3	3	6
P3	L09	0	4	1	5
P3	L10	0	4	1	5
P3	L11	2	2	1	3
P3	L12	2	2	2	4
P4	L14	0	3	2	5
P4	L15	2	1	3	4
P4	L16	0	3	3	6

P3	L09	0	4	2	6
P5	L17	2	3	0	3
P5	L18	0	5	0	5
P5	L19	0	5	0	5
P5	L20	2	3	0	3
P3	L10	0	4	1	5
P3	L11	0	4	1	5
P3	L12	0	4	2	6
P5	L17	2	3	0	3
P5	L18	0	5	1	6
P2	L05	0	6	0	6

A solução ótima: P1 = L02, P2 = L05, P3 = L12, P4 = L16, P5 = L18 e P6 = L22 foi encontrada, analisando apenas 63 configurações até um determinado nível da árvore, de um total de 4096 configurações. A maioria dos cortes aconteceu no nível 3 e 4 da árvore de configurações do exemplo dado, como pode ser visto na Tabela 5.4, onde as linhas estão mostrando os pontos na ordem em que foram rotulados, ou seja, cada ponto esta representando um determinado nível da árvore, e as colunas referem se a:

- ponto: pontos rotulados
- nv: número de vezes que sofreu o corte
- %: valor percentual do número de vezes que sofreu o corte

TABELA 5.4 – LOCALIZAÇÃO DOS CORTES

ponto	nv	%
P6	1	2
P1	5	9
P4	19	33
P3	19	33
P5	13	23
P2	0	0

No exemplo acima a estimativa inicial do número de rótulos sem conflito foi estabelecida como sendo M = 5 e conseguimos chegar à solução ótima examinando 63 configurações até um determinado nível da árvore, que é aproximadamente 1.6% de todas as 4096 configurações possíveis do exemplo. Para verificar a influência do valor inicial de M, foi processado novamente o mesmo exemplo de 6 pontos nas mesmas condições, mas com o valor de M = 4. Neste caso a solução ótima foi encontrada analisando-se 241 configurações até um determinado nível da árvore que é aproximadamente 6% de 4096, ou seja, a quantidade de análises feitas é pequena se considerarmos todas as configurações, mas é importante notar que quanto mais próximo o valor de M, mais rápido o algoritmo encontra a solução ótima. Não foi testado com o valor de M = 6, pois o M deve ser o limitante inferior do número de rótulos sem conflito.

5.4 RESULTADOS OBTIDOS

O algoritmo exato, implementado em linguagem C++, foi aplicado a 8 configurações diferentes de distribuição de pontos, onde cada configuração é composta de 25 pontos gerados randomicamente (Apêndice A3). Tentou-se criar as mesmas dificuldades impostas pelos autores da literatura Christensen et al. (1995) e Verner et. al. (1997) quando geraram o conjunto padrão de dados randomicamente para 1000 pontos. Para tanto manteve-se o tamanho da região, tamanho da folha de papel e aumentou-se o tamanho do rótulo de 8.5%. As demais condições foram mantidas: os rótulos podem ultrapassar o limite da região, foram consideradas 4 posições candidatas, não foi considerada a preferência cartográfica e não houve seleção de pontos. Os resultados da aplicação do algoritmo exato a 8 configurações estão mostrados na Tabela 5.5, onde a ultima linha refere-se a média das 8 configurações e as colunas se referem a:

- № : número da configuração;
- Tempo: tempo de processamento do algoritmo exato em segundos para encontrar a solução ótima em uma estação de trabalho SUN – SPARC20;

- Com conflito: número de rótulos em estado de conflito;
- Sem conflito: número de rótulos em estado de não conflito;
- Sem conflito (%): valor percentual do número de rótulos sem conflito;

TABELA 5.5 – RESULTADOS OBTIDOS POR ALGORITMO EXATO

№	Tempo	Com conflito	Sem conflito	Sem conflito (%)
1	681	2	23	92
2	876	3	22	88
3	864	3	22	88
4	1495	7	18	72
5	749	3	22	88
6	572	4	21	84
7	1726	7	18	72
8	2061	4	21	84
média	1128	4.125	20.875	83.5

5.5 ANÁLISE COMPARATIVA ENTRE OS ALGORITMOS EXATO, CGA E TS

Usando o mesmo conjunto de dados, das 8 configurações geradas na seção 5.4, foram feitos testes usando o TS e CGA, para tornar possível a análise comparativa dos resultados obtidos pelas duas heurísticas, com a solução ótima gerada pelo método exato proposto neste trabalho.

Apesar do exemplo, ser de apenas 25 pontos, o grau de dificuldade do problema é igual ou até um pouco maior do que de 1000 pontos, pois os rótulos são maiores, apesar da área ser o mesmo. Assim sendo os parâmetros da TS continuaram os mesmos e precisou-se calibrar novamente os parâmetros do CGA.

Os parâmetros utilizados pela Busca Tabu foram:

- Tamanho da lista tabu: 7 + INT(0.25 * num. de rótulos em conflito);

- Tamanho da vizinhança: 1 + INT (0.05 * num. de rótulos em conflito);
- Número de iterações para recálculo: 50;

Os parâmetros utilizados pelo CGA foram:

- Número de indivíduos na população inicial: 500;
- Constante de proporção: 0.00005;
- Incremento mínimo permitido: 0.0001;
- Quantidade de indivíduos novos criados em uma determinada geração: 500;
- Número de gerações para parada do processo: 100;
- Um número real $0<d\leq1$: 0.9;
- Número de símbolos #, nas estruturas que compõe a população inicial: 13;
- Tamanho da sub-população base (% da população): 15%;

Os resultados da aplicação do algoritmo exato, TS e CGA, a 8 configurações estão mostrados na Tabela 5.6, onde a ultima linha refere-se a média das 8 configurações e as colunas se referem a:

- № : número da configuração;
- Sem conflito (exato): número de rótulos em estado de não conflito;
- % (exato): valor percentual do número de rótulos sem conflito;
- Sem conflito (TS): número de rótulos em estado de não conflito;
- % (TS): valor percentual do número de rótulos sem conflito;
- * Sem conflito (CGA-média): número de rótulos em estado de não conflito. Refere-se a média das 6 amostras;

- * % (CGA-média): valor percentual do número de rótulos sem conflito. Refere-se a média das 6 amostras;
- * Sem conflito (CGA-melhor): número de rótulos em estado de não conflito. Refere-se a melhor das 6 amostras;
- * % (CGA-melhor): valor percentual do número de rótulos sem conflito. Refere-se a melhor das 6 amostras;

* Para cada distribuição de pontos o CGA foi aplicado 6 vezes gerando assim seis amostras para cada configuração, como na seção 3.7.

TABELA 5.6 – RESULTADOS OBTIDOS POR ALGORITMO EXATO, TS E CGA USANDO O MESMO CONJUNTO DE DADOS

	exato		TS		*CGA-média		*CGA-melhor	
№	Sem conflito	%	Sem conflito	%	Sem conflito	%	Sem conflito	%
1	23	92	22	88	23	92	23	92
2	22	88	22	88	22	88	22	88
3	22	88	21	84	21	84	21	84
4	18	72	18	72	18	72	18	72
5	22	88	19	76	20.5	82	22	88
6	21	84	21	84	20	80	20	80
7	18	72	17	68	16	64	16	64
8	21	84	21	84	21	84	21	84
média	20.875	83.5	20.125	80.5	20.1875	80.75	20.375	81.5

Como mostrado na Tabela 5.6, o CGA trouxe resultados um pouco melhores do que o TS, mas ambos algoritmos chegaram próximo ao valor ótimo gerado pelo algoritmo exato. CGA e TS conseguiram encontrar solução ótima em 50% das configurações e no restante das configurações chegou bastante próximo da solução ótima. Isto mostra que os dois algoritmos heurísticos, na média estão encontrando soluções muito próximas ao

ótimo. Os leiautes com configuração de rótulos após aplicação do método exato, TS e CGA para 25 pontos, encontram-se no Apêndice A2.

Como o objetivo aqui foi a de verificar a qualidade da solução alcançada por CGA e TS com relação à solução ótima, não foi mostrado o tempo de processamento dos mesmos, uma vez que o processamento foi feito usando um processador pentium II 350 Mhz. O processamento do algoritmo exato foi feito, usando uma estação de trabalho SUN – SPARC20, pois o tempo de processamento foi demasiadamente grande, quando foi utilizado um processador pentium II 350 Mhz.