CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Neste trabalho, a rotulação de pontos é tratada como um problema de otimização combinatória, portanto serão descritos a seguir conceitos sobre posições candidatas, preferência cartográfica e função objetivo, que serão necessários para os algoritmos desenvolvidos.

Entende-se por posições candidatas o conjunto de todas as possíveis posições que o rótulo de um determinado ponto pode ocupar. Estas possíveis posições são escolhidas de acordo com uma padronização cartográfica (Christensen et al. 1995). A Figura 2.1 mostra um conjunto de 8 posições candidatas para o rótulo de um ponto. Os números mostrados indicam a preferência cartográfica; quanto menor for o número, maior é a preferência.

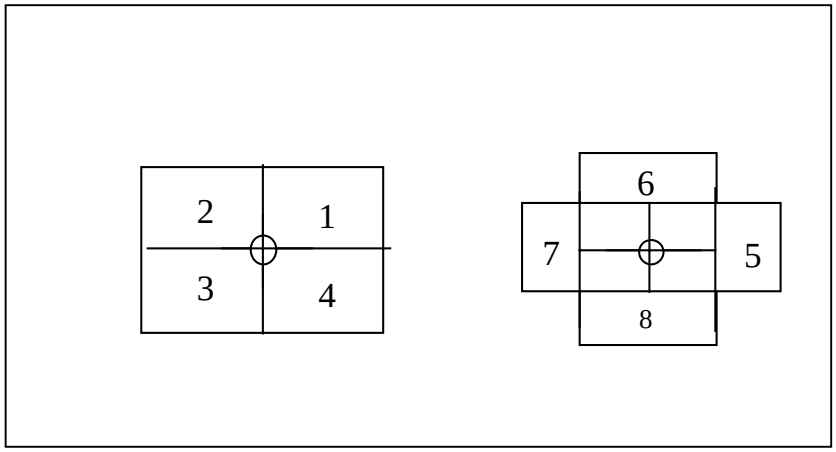

Fig. 2.1 - Conjunto de 8 posições candidatas para o rótulo de um ponto.
FONTE: Christensen et al. (1995, p. 205).

A Figura 2.2 mostra um conjunto de 17 posições candidatas para o rótulo de um ponto com suas respectivas penalidades. Trata-se de uma outra técnica de definição de posições candidatas, proposta por Shawn et al.(1996).

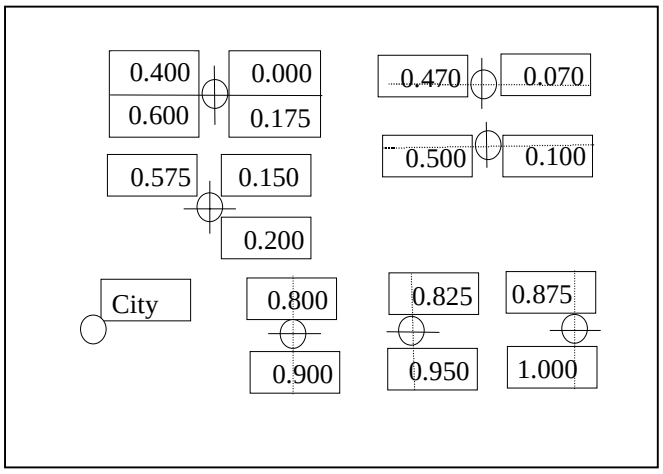

Fig. 2.2 - Conjunto de 17 posições candidatas para o rótulo de um ponto com suas respectivas penalidades.
FONTE:Shawn et al(1996, p. 6).

Função objetivo é uma função a ser otimizada que tem por meta medir a qualidade da rotulação, diferenciando elementos bons dentre as posições candidatas existentes. A qualidade da rotulação depende principalmente dos conflitos existentes entre os rótulos e opcionalmente da preferência cartográfica e seleção de pontos.

Seleção de pontos é o ato de excluir os pontos e os respectivos rótulos se os conflitos persistirem após um determinado número de tentativas. Quando a seleção de pontos é considerada, geralmente os pontos também recebem um custo ou penalidade para evitar a deleção de pontos mais importantes tais como uma capital. Neste trabalho não foi considerada a seleção de pontos.

O restante do capítulo está organizado da seguinte forma. A seção 2.1 faz uma breve descrição de alguns algoritmos da literatura que se encontram descritos em Yamamoto (1998). A seção 2.2 faz uma descrição um pouco mais detalhada do algoritmo Busca Tabu (Yamamoto 1998), que mostrou resultados muito bons comparados aos resultados dos algoritmos encontrados na literatura. Na seção 2.3 e 2.4 serão descritos outros dois algoritmos disponíveis na literatura, o algoritmo genético loGA (“local optimiser

Genetic Algorithm") apresentado em Dijk et. al. (1998) e conjunto independente de vértices de Strijk et. al. (2000).

2.1 BREVE DESCRIÇÃO DE ALGUNS ALGORITMOS DA LITERATURA

Nesta seção será feita uma breve descrição de alguns algoritmos da literatura que se encontram descritos em Yamamoto (1998).

- BUSCA EXAUSTIVA (Christensen et. al., 1993; 1995): O algoritmo utilizado é o "backtracking", onde o rótulo do ponto é colocado em uma das posições não obstruídas. Se um determinado ponto não pode 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. Este algoritmo não é recomendado para problemas de tamanho grande ou moderado por causa da sua natureza exponencial.

- ALGORITMO GULOSO (Christensen et. al., 1995): O algoritmo faz sucessivamente uma otimização local em todos os pontos e então termina. O resultado final é rápido mas a qualidade não é tão boa.

- "DISCRETE GRADIENTE DESCENT" (Christensen et. al., 1995): Este algoritmo repete um procedimento até que uma melhora adicional não seja possível. Este procedimento implica em considerar as posições alternativas do rótulo de cada ponto, calcular o valor de uma função objetivo que resultará se o rótulo for movido e implementar o movimento de rótulo que resultar na maior melhora. Um problema deste algoritmo é a incapacidade de escapar de um mínimo local.

- HIRSCH (Hirsch, 1982): O algoritmo de Hirsch é visto como um sistema dinâmico de repulsão dos rótulos, ou seja, todos os rótulos que se encontram no estado de conflito são movidos na tentativa de eliminar o conflito. Em cada iteração todos os conflitos entre os rótulos são computados, e o movimento de reposicionamento de cada rótulo em conflito é calculado de acordo com a quantidade de sobreposição

existente com os demais rótulos envolvidos. Isto resulta em movimento de repulsão de todos os rótulos em conflito de uma determinada iteração.

- "SIMULATED ANNEALING" (Christensen et. al., 1993 e 1995): Em cada iteração move-se um rótulo para uma nova posição e se calcula ΔE, a mudança na função objetivo causado por reposicionamento do rótulo. Se a nova posição do rótulo piorou o estado geral, aceitar o reposicionamento com probabilidade $P = 1.0 - \exp(-\Delta E/T)$. No início a temperatura T é alta e a probabilidade de aceitar soluções piores também é alta, mas a temperatura é decrementada lentamente com o tempo e a probabilidade de aceitar soluções piores converge para zero. Este tipo de algoritmo pode teoricamente assegurar uma otimização global, mas o seu tempo de processamento pode ser alto.

- ALGORITMO GENÉTICO COM MÁSCARA (Verner et. al., 1997): Neste algoritmo cada cromossomo representa um candidato a solução do problema ou seja o cromossomo representa uma configuração de distribuição de rótulos de um dado conjunto de pontos, onde cada posição no cromossomo corresponde a um ponto e pode assumir uma das posições candidatas que lhe é permitido. A máscara utilizada é do mesmo tamanho do cromossomo, e cada componente da máscara possui o valor 0 (com conflito) ou 1 (sem conflito). Desta maneira, aos pontos cujo rótulo se encontram em conflito será permitido a troca por outra posição candidata através do cruzamento e talvez corrigir a situação de conflito.

2.2 BUSCA TABU (Yamamoto, 1998)

Nesta seção será apresentado o algoritmo Busca Tabu, que mostrou resultados muito bons comparados aos resultados dos algoritmos encontrados na literatura.

Busca Tabu ou "Tabu Search" (TS) é um procedimento heurístico proposto por Fred Glover para resolver problemas de otimização combinatória. A idéia básica é evitar que a busca por soluções ótimas termine ao encontrar um mínimo local ainda distante de um

ótimo global. (Glover. 1989a, 1989b, 1990; Laguna. 1994; Glover et al. 1995; Glover e Laguna, 1997).

Em Yamamoto, 1998 o algoritmo Busca Tabu foi aplicado ao problema de rotulação de pontos da seguinte forma: Dada uma solução inicial, os pontos que mais degradam a solução são selecionados. Escolhe-se para cada ponto em questão a posição candidata com menor custo, gerando assim várias soluções que farão parte da vizinhança. A melhor solução, ou seja, a solução que apresentar o menor custo, será escolhida para ser a próxima solução e será aceita mesmo que o custo seja maior que a solução anterior para escapar de ótimos locais. Para evitar que o procedimento forme um ciclo, os últimos k pontos visitados são armazenados na lista tabu T e a solução será rejeitada se o ponto que sofreu a modificação na solução escolhida estiver nesta lista T. A lista T tem tamanho finito, portanto quando um ponto é introduzido nesta lista, o ponto mais antigo da lista é removido, podendo então participar novamente da formação das próximas soluções. Algumas vezes é permitido aceitar que o ponto que faz parte da lista T participe na formação da próxima solução, se a solução gerada for “suficientemente” boa. Este procedimento é conhecido como critério de aspiração. No problema de rotulação foi usado também o conceito de freqüência normalizada para diversificar a busca e penalizar os pontos que não causam melhora.

2.2.1 DESCRIÇÃO DO PROCESSO DE BUSCA DO TS

O algoritmo apresentado a seguir descreve em linhas gerais o processo de busca do TS para resolver o problema de rotulação de pontos.

1. **Gerar uma configuração inicial**, rotulando cada ponto com o seu rótulo na posição candidata de menor penalidade;

2. **Repetir** os seguintes passos até alcançar uma solução sem conflitos ou por um número pré-definido de iterações;

    - **Criar uma lista de candidatos** para iteração corrente;

- **Recalcular a lista de candidatos** para encontrar posição de rótulo de menor custo para cada ponto referenciado na lista de candidatos;

- **Escolher o melhor candidato da lista**, baseado no custo, levando em consideração a lista tabu e o critério de aspiração;

- **Realizar a mudança de configuração**, designando a solução obtida como sendo a nova solução corrente. Cada mudança de configuração consiste na modificação da posição de rótulo de um ponto;

- **Atualizar a lista tabu**.

O algoritmo TS descrito acima envolve seis componentes: função objetivo, lista tabu, lista de candidatos, mudança de configuração, critério de aspiração e memória de longo prazo. Eles serão descritos a seguir.

Função objetivo (F): O algoritmo TS usado aqui é inteiramente determinístico e seleciona os melhores candidatos. Portanto é necessário examinar e comparar os candidatos, o que acarreta um grande número de cálculos, especialmente quando o número de pontos é grande. Então uma boa função objetivo é aquela que o custo pode ser calculado facilmente, tornando a busca eficiente e obtendo ao mesmo tempo soluções de qualidade. A função objetivo de minimização usada é

$$\sum_{i=1}^{np} C(i)$$

Onde, np = número de pontos; C(i) = custo de cada ponto i, definido onde, overlap(i) = número de conflitos do rótulo associado ao ponto i (Figura 2.3); preference(i) = preferência cartográfica do rótulo ativo no ponto i e rótulos em conflito com ele; α_1 nível de consideração a ser dada aos rótulos em conflito; α_2 = nível de consideração a ser dada a preferência cartográfica.

Para o cálculo de overlap(i), foi utilizado a informação referente ao número de conflitos do rótulo associado ao ponto i (Figura 2.3), pois existe a necessidade de se saber a informação sobre o número de conflitos do ponto i em questão.

Os parâmetros α_1 e α_2 são manipulados pelo usuário que pode escolher o que é mais importante, sem conflitos ou qualidade cartográfica. Se $\alpha_2 = 0$, a preferência cartográfica não é considerada.

Lista tabu: é um componente essencial do algoritmo, e armazena pontos que ultimamente sofreram mudança de posição de rótulo.

Em Yamamoto, 1998 foi usada uma lista de tamanho dinâmico, porque o problema de rotulação de pontos necessita de uma lista tabu grande no início para evitar solucionar conflitos somente em certas regiões do mapa. Entretanto, quando o número de conflitos diminui, o tamanho da lista tabu pode ser reduzido, para que a busca seja conduzida em pequenas regiões do mapa, apenas para se fazer um ajuste final.

O tamanho da lista tabu usado foi 7 + INT(0.25 * número de rótulos em conflito). Após algumas iterações o número de conflitos diminui e consequentemente o tamanho da lista tabu. O coeficiente 0.25 assim como o recalculo do tamanho da lista tabu a cada 50 iterações foram estabelecidas depois de testes e experimentos feitos em configurações com 100, 250, 500, 750 e 1000 pontos (detalhes em Yamamoto, 1998).

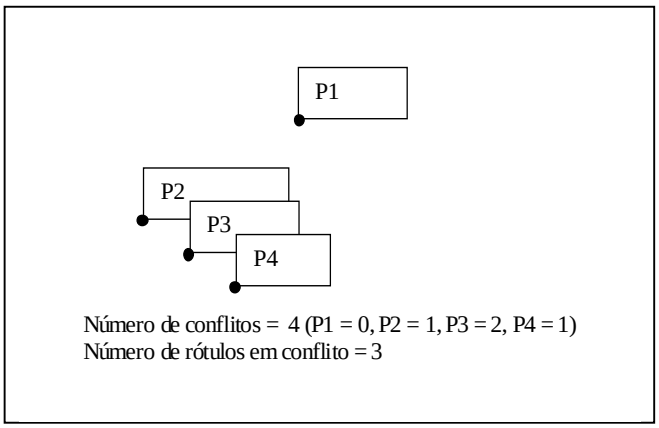

Fig. 2.3 - Calculo de conflitos

Para o cálculo do tamanho da lista tabu e tamanho da lista de candidatos descrito no próximo item, foram utilizados a informação referente ao número de rótulos em conflito (Figura 2.3), pois alem de mostrar o número total de rótulos que se encontra em conflito, é o tipo de contagem de conflitos que se é adotado na maioria dos artigos encontrados na literatura.

Lista de candidatos: Esta lista é composta por um conjunto de triplas (ponto, rótulo, custo) que apresentam o custo alto na configuração corrente. Os custos associados a cada ponto são calculados como descrito na memória de longo prazo. Em geral (dependendo do nível de consideração α_1, α_2) soluções de custo alto têm um grande número de conflitos e seus rótulos não estão nas melhores posições cartográficas.

O tamanho da lista é recalculado apos 50 iterações consecutivas usando a expressão: 1 + INT (0.05 * número de rótulos em conflito). O coeficiente 0.05 foi escolhido apos testes em nove diferentes configurações de 1000 pontos. A média de rótulos sem sobreposição das nove diferentes configurações para coeficientes 0.03, 0.04, 0.05, 0.06 e 0.07, mostraram que o coeficiente 0.05 produziu resultados melhores (detalhes em Yamamoto, 1998).

Mudança de configuração: Todas as triplas da lista de candidatos procuram a melhor posição de rótulo. Apos mudanças nas posições de rótulos, o candidato com o menor custo é escolhido. Soluções geradas por um ponto que faz parte da lista tabu são descartados e a próxima melhor alternativa é selecionada.

Critério de aspiração: Em algumas situações, é necessário considerar alternativas que são parte da lista tabu. Em tais casos, um critério de aspiração é usado para ignorar a restrição tabu em dois casos:

- A solução é selecionada se o seu custo for menor do que a melhor solução encontrada.
- Se todas as soluções candidatas são parte da lista tabu e não satisfaz o critério acima, então o candidato com o maior tempo de permanência na lista tabu é escolhido.

Memória de longo prazo: freqüentemente, o custo de diferentes soluções são iguais, resultando na entrada dos mesmos pontos na lista de candidatos. Portanto existe a necessidade de diversificar a busca. Foi usado então a estratégia de memória baseada em freqüência que conta o número de vezes que o ponto muda a posição do seu rótulo e após 50 iterações consecutivas divide-se o valor acumulado de cada ponto pelo valor máximo, obtendo a freqüência normalizada. Esta informação é usada para aplicar penalidades aos pontos que não trazem melhora, fazendo com que perca a sua atratividade pois, em rotulação a escolha dos pontos que compõe a lista de candidatos é em função do maior custo. O custo C(i) de cada ponto "i" foi portanto modificado para:

CN(i) = C(i) – freqüência normalizada(i)

Onde a freqüência normalizada(i) é um instrumento de diversificação da busca.

2.2.2 EXEMPLO DE APLICAÇÃO DE TS PARA 6 PONTOS

Para tornar claro o uso do algoritmo TS, será descrito em detalhes o seu uso. Foi considerada uma configuração de seis pontos, como ilustrado na Figura 2.4. Cada ponto tem quatro posições potenciais de rótulo (L0, L1, L2 e L3) e cada posição de rótulo tem um custo associado (0.0, 0.4, 0.6 e 0.9), onde 0.0 indica a melhor posição e 0.9 a pior (Figura 2.5).

Fig. 2.4 - Configuração Inicial

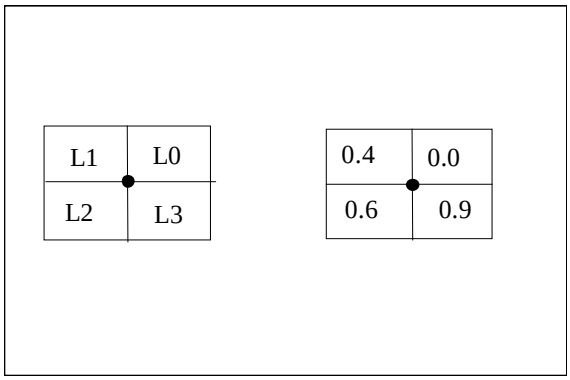

Fig. 2.5 - Posições candidatas e preferência cartográfica.

Na configuração inicial existem 5 sobreposições de rótulos. Os pontos e seus rótulos são: P0 = Youngstown (10 caracteres), P1 = Yankton (7 caracteres), P2 = Yakima (6 caracteres), P3 = Worcester (9 caracteres), P4 = Wisconsin Dells (15 caracteres) e P5 = Winston-Salem (13 caracteres). Winston-Salem é o único sem sobreposição.

Foi considerado para o exemplo, importâncias iguais para a sobreposição e preferência cartográfica, portanto foram usados $\alpha_1 = 1$ e $\alpha_2 = 1$ em C(i) = α1 overlap(i) + α2 preference(i), para se calcular o custo do ponto i. A expressão 2 + INT(0.25 * número de rótulos em conflito) foi usada para calcular o tamanho da lista tabu, mas se o tamanho da lista for maior que 4, ele é forçado a ficar igual a 4. O tamanho da lista de candidatos adotado foi 2 + INT(0.05 * número de rótulos em conflito) e eles são recalculados a cada 5 iterações. A seguir o algoritmo para resolver os conflitos entre os rótulos é apresentado passo a passo.

Estado inicial: tamanho da lista tabu = 4; tamanho da lista de candidatos = 2; solução = {(P0, L0, 1.0), (P1,L0, 3.0), (P2, L0, 3.0), (P3, L0, 2.0), (P4, L0, 3.0), (P5, L0, 0.0)}; lista tabu = {}; F = 12.0; melhor solução = 12.0 e rótulos em conflito = 5

Iteração 1: lista de candidatos = {(P1, L0, 3.0), (P2, L0, 3.0)}; lista de candidatos recalculado = {(P1, L1, 2.4), (P2, L1, 3.4)}; candidato escolhido = (P1, L1, 2.4); solução = {(P0, L0, 1.4), (P1, L1, 2.4), (P2, L0, 2.0), (P3, L0, 2.0), (P4, L0, 3.4), (P5, L0, 0.0)}; lista tabu = {P1}; F = 11.2; melhor solução = 11.2 e rótulos em conflito = 5.

A lista de candidatos é composta no início por triplas (P1, L0, 3.0) e (P2, L0, 3.0), os dois primeiros casos de custo alto. Em seguida, as quatro posições de rótulos potenciais do ponto P1 são examinados e a posição de rótulo com o menor custo é escolhida. O mesmo procedimento é repetido para o ponto P2. No exemplo, a posição de rótulo L1 com custo 2.4 é escolhida para o ponto P1 e a posição de rótulo L1 com custo 3.4 é selecionada para o ponto P2.

O candidato (P1, L1, 2.4) foi escolhido porque seu custo é menor do que o candidato (P2, L1, 3.4) e P1 não está na lista tabu. Aplicando o mesmos passos iterativamente, serão obtidos os seguintes resultados:

Iteração 2: lista de candidatos = {(P4, L0, 3.4), (P1, L1, 2.4)}; lista de candidatos recalculado = {(P4, L2, 1.6), (P1, L2, 2.6)}; candidato escolhido = (P4, L2, 1.6); solução = {(P0, L0, 1.4), (P1, L1, 1.4), (P2, L0, 1.0), (P3, L0, 1.0), (P4, L2, 1.6), (P5, L0, 1.6)}; lista tabu = {P4, P1}; F = 8.0; melhor solução = 8.0 e rótulos em conflito = 6

Iteração 3: lista de candidatos = {(P5, L0, 1.6), (P4, L2, 1.6)}; lista de candidatos recalculado = {(P5, L2, 0.6), (P4, L1, 1.8)}; candidato escolhido = (P5, L2, 0.6); solução = {(P0, L0, 1.4), (P1, L1, 1.4), (P2, L0, 1.0), (P3, L0, 1.0), (P4, L2, 0.6), (P5, L2, 0.6)}; lista tabu = {P5, P4, P1}; F = 6.0; melhor solução = 6.0 e rótulos em conflito = 4.

Iteração 4: lista de candidatos = {(P0, L0, 1.4), (P1, L1, 1.4)}; lista de candidatos recalculado = {(P0, L1, 1.8), (P1, L0, 2.0)}; candidato escolhido = (P0, L1, 1.8); solução = {(P0, L1, 1.8), (P1, L1, 1.8), (P2, L0, 1.0), (P3, L0, 1.0), (P4, L2, 0.6), (P5, L2, 0.6)}; lista tabu = {P0, P5, P4, P1}; F = 6.8; melhor solução = 6.0 e rótulos em conflito = 4.

Iteração 5: lista de candidatos = {(P1, L1, 1.8), (P0, L1, 1.8)}; lista de candidatos recalculado = {(P1, L0, 1.0), (P0, L0, 1.4)}; candidato escolhido = (P1, L0, 1.0); solução = {(P0, L1, 0.4), (P1, L0, 1.0), (P2, L0, 2.0), (P3, L0, 1.0), (P4, L2, 0.6),

(P5, L2, 0.6)}; lista tabu = {P1, P0, P5, P4}; F = 5.6; melhor solução = 5.6 e rótulos em conflito = 3.

Os pontos P0 e P1 estão na lista tabu, mas o candidato (P1, L0, 1.0) foi escolhido, porque o valor F= 5.6 da nova configuração é menor do que a melhor solução = 6.0 alcançada.

Iteração 6: lista de candidatos = {(P1, L0, 1.0), (P2, L0, 2.0)}; lista de candidatos recalculado = {(P1, L1, 1.8), (P2, L3, 1.9)}; candidato escolhido = (P2, L3, 1.9); solução = {(P0, L1, 0.4), (P1, L0, 0.0), (P2, L3, 1.9), (P3, L0, 1.9), (P4, L2, 0.6), (P5, L2, 0.6)}; lista tabu = {P2, P1, P0, P5}; F = 5.4; melhor solução = 5.4 e rótulos em conflito = 2.

Nesta iteração é recalculado o tamanho da lista tabu, tamanho da lista de candidatos e o custo da solução para cada ponto usando a freqüência normalizada: tamanho da lista tabu = 2; tamanho da lista de candidatos = 2; solução = {(P0, L1, 0.01), (P1, L0, 0.0), (P2, L3, 1.44), (P3, L0, 1.9), (P4, L2, 0.14), (P5, L2, 0.14)}; lista tabu = {P2, P1}.

O mesmo procedimento é executado até alcançar uma solução sem sobreposição ou até um pre-especificado limite de iterações. A Figura 2.6 mostra o resultado da aplicação de TS ao exemplo de seis pontos.

Fig. 2.6 - Após aplicação do TS ao exemplo da Figura 2.4

2.2.3 RESULTADOS OBTIDOS

Christensen et al. (1995) e Verner et al. (1997) compararam vários algoritmos usando um conjunto padrão de dados gerados randomicamente:

- Região de tamanho 792 x 612 unidades
- Rótulo de tamanho fixo 30 x 7 unidades
- Folha de papel de tamanho 11 x 8.5 polegada
- Número de pontos: n = 100, 250, 500, 750, 1000
- Para cada n, gerar 25 configurações diferentes de distribuição aleatória de pontos através do uso de diferentes sementes.
- Para cada n calcular a média percentual do número de rótulos sem conflito das 25 configurações.
- Não foi determinada nenhuma penalidade para os rótulos que se situam além do limite da região.
- Foram consideradas 4 posições candidatas.
- Não foi considerada a preferência cartográfica (todas as posições candidatas são igualmente desejáveis).
- Não houve seleção de pontos (não deleta ponto ou rótulo que estiver em conflito na configuração final).

O algoritmo TS, implementado em linguagem C++, foi aplicado a 25 configurações diferentes de distribuição de pontos para 100, 250, 500, 750 e 1000 pontos (Apêndice A3), usando o mesmo conjunto padrão de dados gerados aleatoriamente e as

mesmas condições impostas pelos autores da literatura Christensen et al.(1995) e Verner et al.(1997).

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 recalculo: 50

As médias obtidas da aplicação de TS nas 25 configurações estão mostradas na Tabela 2.1, onde as linhas se referem a:

- Tempo: tempo médio de processamento do algoritmo TS em segundos para alcançar o estado de não conflito, ou de menor número de conflitos dentre as 30000 iterações, em um processador Pentium II 350 Mhz. O tempo mostrado se refere apenas ao tempo de processamento do algoritmo TS, ou seja, as informações de conflitos são dados de entrada.

- Tempo total: tempo médio de processamento do algoritmo TS em segundos para processar 30000 iterações em um processador Pentium II 350 Mhz.

- Com conflito: número médio de rótulos em estado de conflito.

- Desvio padrão: desvio padrão dos rótulos colocados sem conflito.

- Sem conflito: número médio de rótulos em estado de não conflito.

- Sem conflito (%): trata-se da média percentual do número de rótulos sem conflito das 25 configurações.

TABELA 2.1 – RESULTADOS OBTIDOS POR TS USANDO O CONJUNTO PADRÃO DE DADOS

Num. de pontos	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos
Tempo	0	0	1.36	76	352.9
Tempo total	0	0	9	202	507
Com conflito	0	0	4	24	100
Desvio padrão	0.0	0.0	2.2	6.1	12.8
Sem conflito	100	250	496	726	900
Sem conflito (%)	100.00	100.00	99.3	96.8	90.0

TABELA 2.2 – RESULTADOS OBTIDOS POR VÁRIOS ALGORITMOS USANDO O CONJUNTO PADRÃO DE DADOS

Algoritmo	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos
**Busca Tabu	100.00	100.00	99.28	96.76	90.00
GA com máscara	100.00	99.98	98.79	95.99	88.96
*GA sem máscara	100.00	98.40	92.59	82.38	65.70
“Simulated Annealing”	100.00	99.90	98.30	92.30	82.09
Zoraster	100.00	99.79	96.21	79.78	53.06
Hirsch	100.00	99.58	95.70	82.04	60.24
* “3-Opt Gradient Descent”	100.00	99.76	97.34	89.44	77.83
* “2-Opt Gradient Descent”	100.00	99.36	95.62	85.60	73.37
“Gradient Descent”	98.64	95.47	86.46	72.40	58.29
Algoritmo guloso	95.12	88.82	75.15	58.57	43.41
Busca exaustiva	84.56	65.63	44.06	29.06	19.53

Adaptada de Verner et al. (1997, p.273).

* Algoritmos não estudados neste trabalho. Verner et al. (1997); Christensen et al. (1995);

** A Tabela 2.2 é uma cópia fiel da tabela encontrada em Verner et al. (1997) com exceção do algoritmo Busca Tabu que foi inserido por nós.

A tabela 2.2 mostra os resultados obtidos por vários algoritmos da literatura. As colunas se referem ao percentual médio de rótulos sem conflitos para 100, 250, 500, 750 e 1000 pontos, por diferentes algoritmos da literatura e as linhas mostram o percentual médio de rótulos sem conflito alcançados pelos algoritmos de otimização testados na literatura por Christensen et al. (1995) (Busca Exaustiva, Algoritmo Guloso, "Gradient Descent", "2-Opt Gradient Descent', "3-Opt Gradient Descent", Algoritmo de Hirsch, Zoraster (seção 2.5) e "Simulated Annealing"), por Verner et al. (1997) (GA sem máscara e GA com máscara) e por Yamamoto et. al. (1999) (Busca Tabu).

Não foi possível fazer as comparações dos tempos computacionais entre os algoritmos, pois o processamento do Busca Tabu foi feito usando um processador pentium II 350 Mhz, o Christensen utilizou uma estação de trabalho DEC 3000/400 AXP e Verner usou uma estação de trabalho Sun-Sparc 10.

2.3 ALGORITMO GENÉTICO loGA (Dijk et. al. , 1998)

Nesta seção, será descrito o algoritmo genético loGA ("local optimiser Genetic Algorithm") de Dijk et. al., que se encontra disponível na literatura. De acordo com a sua formulação, cada cromossomo representa uma configuração de distribuição de rótulos de um dado conjunto de pontos, onde cada posição de cromossomo corresponde a um ponto e pode assumir uma das posições candidatas que lhe é permitido.

O algoritmo apresentado a seguir, descreve em linhas gerais o processo de busca do loGA para resolver o problema de rotulação de pontos.

Algoritmo loGA:

1. **Gerar** população inicial aleatoriamente ou usar qualquer método de inicialização.
2. **Repetir** até alcançar a condição de término

- **Selecionar** 2 cromossomos aleatórios da população para reprodução
- **Fazer o cruzamento**
- **Aplicar otimizador local** nos filhos
- **Selecionar** os 2 melhores cromossomos para substituir os pais, dentre os 4 cromossomos (2 filhos e 2 pais), levando em consideração o número de rótulos sem conflito.

Para o cruzamento são usados as máscaras dos cromossomos que são pre-computados. Cada máscara é do mesmo tamanho do cromossomo e é formada por 0 (conflito) e 1 (não conflito).

As máscaras dos pais selecionados são usadas para formar uma única máscara através da operação "OR". Esta máscara conhecida por multi-máscara será usada para realizar o cruzamento.

Na operação de cruzamento o filho C1 receberá os genes do pai P1 se a multi-máscara possuir valor 1 na posição equivalente ao gen, caso contrário receberá os gens do pai P2. O filho C2 receberá por sua vez os gens do pai P1, quando a multi-máscara apresentar valor 0 e gens do pai P2 quando a multi-máscara apresentar o valor 1.

Após o cruzamento usa-se um otimizador local nos cromossomos filhos, que verifica as posições candidatas dos gens que apresentam conflito e troca, se possível, com uma posição candidata que não apresenta conflito.

Os autores implementaram os algoritmos SA de Christensen et.al. (1995) e GA de Verner et al. (1997) e compararam os resultados usando o conjunto padrão de dados gerados aleatoriamente. A Tabela 2.3 mostra os resultados obtidos pelos algoritmos. As colunas se referem ao percentual médio de rótulos sem conflitos para 100, 250, 500, 750, 1000 e 1500 pontos, por diferentes algoritmos e as linhas mostram o percentual médio de rótulos sem conflito alcançados pelos algoritmos LoGA, SA e GA.

Infelizmente, não é possível incluir o loGA, na Tabela 2.2 dos resultados obtidos por vários algoritmos, pois SA e GA alcançaram resultados diferentes dos trabalhos originais de Christensen et.al. (1995) e Verner et al. (1997).

TABELA 2.3 – RESULTADOS OBTIDOS (Dijk et. al., 1998)

Algoritmo	100 pontos	250 pontos	500 pontos	750 pontos	1000 pontos	1500 pontos
LoGA	100.00	99.36	98.00	93.333	84.94	63.346
SA	100.00	99.36	97.92	92.933	84.70	63.466
GA	99.40	99.20	96.80	87.146	69.76	17.466

2.4 CONJUNTO INDEPENDENTE DE VÉRTICES (Strijk et. al., 2000)

A versão do problema de rotulação de pontos com seleção de pontos pode ser vista como o de encontrar o maior conjunto independente de vértices em um grafo.

2.4.1 CONSTRUÇÃO DO GRAFO DE RELACIONAMENTO PARA ROTULAÇÃO DE PONTOS

O objetivo aqui é organizar as informações referentes a conflitos de rótulos de pontos, usando uma representação em grafo. Cada vértice representa um rótulo e vértices adjacentes são compostos por posições candidatas que não podem estar ativas simultaneamente sem que ocorra conflito e estão interligados por uma aresta. O conjunto independente de vértices diz respeito a par de vértices não adjacentes, ou seja, é composto por rótulos que podem estar ativos simultaneamente sem causar conflitos. Rótulos ativos neste contexto diz respeito a posições candidatas do ponto que são mostrados no mapa.

A Figura 2.7 mostra 2 pontos com 4 posições candidatas e os conflitos existentes entre os rótulos, e a Figura 2.8 mostra o grafo associado.

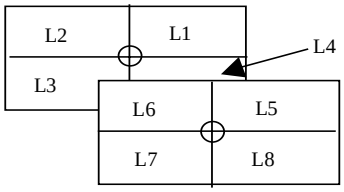

Figura 2.7 posições candidatas e os conflitos

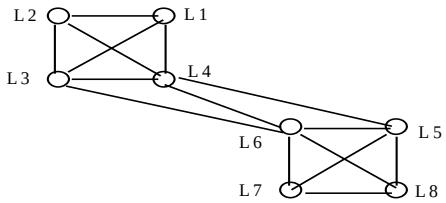

Figura 2.8 grafo de relacionamento das posições candidatas

Portanto dado uma instância do problema de n pontos, onde cada ponto possui 4 posições candidatas, podemos representar o relacionamento das posições candidatas em um grafo G (V,E), onde V = {L1, L2, ..., Ln} representa o conjunto de rótulos, ou seja, cada rótulo está representado por um vértice e E = { A1, A2, ..., Am} 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.

Se usarmos a matriz de adjacências para representarmos o grafo de relacionamento das posições candidatas, teremos uma matriz simétrica onde a soma por linha ou coluna fornece o grau de cada vértice, que representa o número de posições candidatas que não podem estar ativas se o vértice em questão estiver ativo. Assim sendo, quanto maior o número de conflitos com outras posições candidatas, maior é o grau do vértice como pode ser visto na matriz de adjacências da Figura 2.9, onde as linhas e as colunas são as

posições candidatas e o “1” da matriz representa que as posições candidatas i e j não podem estar ativas ao mesmo tempo.

	L1	L2	L3	L4	L5	L6	L7	L8
L1	0	1	1	1	0	0	0	0
L2	1	0	1	1	0	0	0	0
L3	1	1	0	1	0	1	0	0
L4	1	1	1	0	1	1	0	0
L5	0	0	0	1	0	1	1	1
L6	0	0	1	1	1	0	1	1
L7	0	0	0	0	1	1	0	1
L8	0	0	0	0	1	1	1	0

Figura 2.9 matriz de adjacências do grafo da Figura 2.8

Grau dos vértces: L1 = 3; L2 = 3; L3 = 4; L4 = 5; L5 = 4; L6 = 5; L7 = 3; L8 = 3.

2.4.2 ALGORITMO

Nesta seção será apresentado em linhas gerais o algoritmo de busca local que usa Busca Tabu. O algoritmo apresentado em Strijk et. al. (2000) está composto de algoritmo principal e algoritmo Busca Tabu.

Algoritmo principal

Início:

S := ∅ conjunto independente (solução)

L := V lista de vértices ativos

Enquanto $L \neq \emptyset$

- **Extrair** vértice v de L
- **Tentar encontrar** S’ de tamanho |S| + 1, usando **algoritmo Busca Tabu** com $S0 = S \cup \{v\}$
- **Se** não encontrar S’

- **Tornar** vértice v inativo

- **Senão**
  - **S' = (S – U) ∪ W**, onde U é o vetor com vértices removidos de S e W é o vetor com vértices adicionados ao S
  - **Acrescentar** em L todo vértice u∉ L que seja adjacente ao W ativo

FIM_Enquanto

Algoritmo Busca Tabu descrito em Strijk et. al. (2000)

Início:

S = S0	solução inicial
T = ∅	lista tabu
Tmin = 2	tamanho mínimo de T
Tmax = 7	tamanho máximo de T
C1 = 100	número de iterações para se conseguir escapar do mínimo local
C2 = 10	número de tentativas para sair do mínimo local
C3 = 3	número de iterações realizados com a função objetivo perturbada

Iteração:

- S' = (S – {u}) ∪ {w}

  u ∈ S e w ∈ (V – S) são escolhidos visando minimizar a f(S). Os vértices u e w não devem pertencer a lista tabu
- S ← S'
- Se f(S) não diminuir, os vértices envolvidos vão para lista tabu T
- Quando f(S) = 0, S é o conjunto independente e a busca termina.

Dado uma solução inicial S0 com |S0| = α0, tenta-se encontrar uma solução S que satisfaz |S| = α0 e minimize a função objetivo f(S) = |E(S)| onde E(S) = {{u,v} ∈ E | u,v ∈ S} que denota o conjunto de arestas com pontos terminais em S. Para tanto mantêm-se 2 partições, uma com vértices S e outra com vértices V-S.

O algoritmo encontra mínimo local se em C1 iterações não encontrar uma solução melhor do que a melhor solução até então encontrada S*. Neste caso, faz-se (C2 – 1) tentativas para escapar do mínimo local, onde cada tentativa consiste de C1 iterações com as C3 primeiras iterações realizados com a função objetivo perturbada:

$$ff(S)=|E(S)| + \frac{1}{2}\sum_{v \in V} P^T X^S$$

onde cada elemento do vetor X recebe valor 1(um) se $v \in S$ e recebe valor 0(zero) se $v \notin S$. P é um vetor que possui a informação de quantas vezes cada vértice $v \in V$ esteve envolvido em conflito. Neste caso o tamanho da lista tabu é modificado para T(i) = Tmin + i (Tmax – Tmin) / C2 onde i é o número da tentativa.

Se encontrar a solução S, onde | E(S) | < | E(S*) |, volta-se a usar a função objetivo normal, o tamanho da lista tabu é Tmin e o processamento continua. Caso as tentativas não tenham sucesso a busca termina.

2.4.3 RESULTADOS OBTIDOS

Os autores compararam com o algoritmo SA de Christensen et.al. (1995) usando o conjunto padrão de dados gerados aleatoriamente. A Tabela 2.4 mostra os resultados obtidos, onde as linhas referem-se a número médio de rótulos em estado de não conflito, alcançados por TS (Strijk et. al., 2000) e SA (Christensen et. al., 1995). As colunas referem-se a número médio de rótulos em estado de não conflito das 25 configurações, para 100, 250, 500, 750 e 1000 pontos. É possível verificar que, quando o número de pontos é maior, o algoritmo TS (Strijk et. al., 2000) traz resultados melhores do que SA (Christensen et. al., 1995).

TABELA 2.4 – RESULTADOS OBTIDOS (Strijk et. al., 2000)

Algoritmo	100 pontos	250 pontos	500 pontos	750 pontos	950 pontos
TS (Strijk et. al., 2000)	100	249.5	490.9	704.6	844.7
SA (Christensen et.al., 1995)	100	249.6	491.7	703.1	834.4

Estes resultados não são comparáveis com os da Tabela 2.2, pois estes últimos foram obtidos para um problema que não admite seleção de pontos.

2.5 PROGRAMAÇÃO MATEMÁTICA

Trata-se de uma programação inteira 0-1, introduzido por Zoraster (1986, 1990, 1991) para resolver o problema de sobreposição de rótulos. Dado:

k número de pontos

N_j $1 \leq j \leq k$ número de posições candidatas do ponto j

X_{ij} $1 \leq i \leq N_j$ $1 \leq j \leq k$ posição candidata i do ponto j

$X_{ij} = 0$ ausência do rótulo

$X_{ij} = 1$ presença do rótulo

$\sum_{i=1}^{N_j} X_{ij} = 1$ $1 \leq j \leq k$ obriga cada ponto a ter somente um rótulo ativo

W_{ij} $1 \leq i \leq N_j$ $1 \leq j \leq k$ representa a preferência cartográfica

Q número de sobreposição entre rótulos candidatos. Cada par de sobreposição é contabilizado como sendo 1.

$X_{rq,\,sq} + X_{r'q,\,s'q} \leq 1$ $1 \leq q \leq Q$ expressa a não sobreposição de dois rótulos candidatos.

$X_{rq,\,sq}$ $1 \leq q \leq Q$ posição candidata r do ponto s

$X_{r'q,\,s'q}$ $1 \leq q \leq Q$ posição candidata r' do ponto s'

Temos então :

$$\text{Min} \quad \sum_{j=1}^{k} \sum_{i=1}^{N_j} W_{ij} X_{ij}$$

$$\text{Suj. a :} \quad \sum_{i=1}^{N_j} X_{ij} = 1 \qquad 1 \leq j \leq k$$

$$X_{rq,\,sq} + X_{r'q,\,s'q} \leq 1 \qquad 1 \leq q \leq Q$$

$$X_{ij} \in \{0,1\}$$

Seleção de pontos é obtida, especificando uma posição candidata especial que indica que o ponto não está sendo considerado.

O autor combina relaxação lagrangeana, “subgradient optimization” e várias heurísticas específicos ao problema em questão para alcançar a solução. Primeiramente, ele fez uma relaxação com relação a sobreposição de dois rótulos, incluindo um termo adicional de penalidade na função objetivo.

Temos então:

$$\text{Min} \quad \sum_{j=1}^{k}\sum_{i=1}^{N_j} w_{ij}\, x_{ij} + \sum_{q=1}^{Q}\left(x_{rq,\,sq} + x_{r'q,\,s'q} - 1\right)dq$$

$$\text{Suj. a :} \quad \sum_{i=1}^{N_j} x_{ij} = 1 \qquad 1 \le j \le k$$

$$x_{ij} \in \{0,1\}$$

$$dq \ge 0 \qquad 1 \le q \le Q \qquad \text{multiplicador lagrangeano}$$

Algoritmo:

- **Rotular** cada ponto na melhor posição candidata e **gerar** rotulação corrente (CL)
- **Inicializar** ACS (conjunto de conflitos ativo) como vazio
- **Repetir** por 40 iterações ou até alcançar uma configuração sem conflito
  - **Identificar** os conflitos entre os rótulos e **adicionar** em ACS se não existir
  - **Fazer** uma copia CL’ de CL
  - **Chamar** heurística lagrangeano
- **Retornar** CL

Com o objetivo de produzir novas soluções, a heurística lagrangeano usa “subgradient optimization”. O multiplicador lagrangiano é incrementado, decrementado ou não e levado em consideração no calculo da função objetivo dos rótulos em conflito.

Sabe-se que $x_{ij} \in \{0,1\}$, logo as configurações possíveis são :

$(x_{rq,sq} + x_{r'q,s'q} - 1)dq$

0		0	-1	=	-1 dq (decrementa o multiplicador lagrangeano)
0		1	-1	=	0 dq (não leva em consideração o multiplicador lagrangeano)
1		0	-1	=	0 dq (não leva em consideração o multiplicador lagrangeano)
1		1	-1	=	1 dq (incrementa o multiplicador lagrangeano)

Algoritmo: heurística lagrangeano para rotulação

Repetir até 400 iterações ou até alcançar uma solução aceitável.

- **Atualizar** CL' , escolhendo posições candidatas com menor função objetivo para cada ponto
- **Se** CL' é melhor que CL, **copiar** CL' para CL
- **Se** os dois rótulos em conflito estiverem ativos, **incrementar** o multiplicador lagrangeano a função objetivo dos dois rótulos em questão
- **Se** os dois rótulos em conflito não estiverem ativos, **decrementar** o multiplicador lagrangeano da função objetivo dos dois rótulos em questão

O algoritmo possui duas fraquezas:

- Mínimo local
- Comportamento cíclico inútil

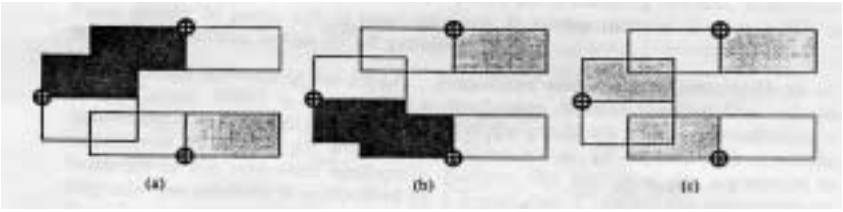

Figura 2.10 – configuração estável e instável da abordagem do Zoraster

A Figura 2.10 mostra a convergência do algoritmo para o mínimo local, como pode ser visto na Figura 2.10(a), a função objetivo dos rótulos ativos em conflito são

incrementados com multiplicador lagrangeano tornando-se menos atraente, enquanto que a função objetivo dos rótulos inativos em conflito são decrementados de multiplicador lagrangeano tornando-se assim mais atraente. Isto conduz a configuração modificada da Figura 2.10(b) que por sua vez volta a configuração da Figura 2.10(a) por mesmos motivos descritos acima e nunca consegue alcançar a configuração da Figura 2.10(c).

O autor tenta resolver o problema aplicando o multiplicador lagrangeano a função objetivo de apenas um dos rótulos em conflito. Mas a qual das duas funções objetivos o multiplicador lagrangeano será então aplicado? Será de acordo com a iteração corrente, ou seja se a iteração corrente for impar, o multiplicador lagrangeano será aplicado a função objetivo da primeira variável e quando a iteração corrente for par, o multiplicador lagrangeano será aplicado a função objetivo da segunda variável.

Outra modificação sugerido por ele é a de reduzir o valor do multiplicador lagrangeano se um especificado número de iterações tem passado sem apresentar uma melhora, pois neste caso o algoritmo se encontra na região próximo ao mínimo local e redução do valor do multiplicador lagrangeano de tempos em tempos, torna o algoritmo capaz de identificar o melhor mínimo.

A Figura 2.11 ilustra o caso de sobreposição de 3 rótulos, onde os rótulos dos dois pontos abaixo são encorajados a mover para a área em conflito. O autor tenta sanar esta deficiência fixando os X_{ij} dos rótulos com mais de 3 conflitos para 0 (zero), quando após 400 iterações ainda não conseguiu alcançar uma solução aceitável.

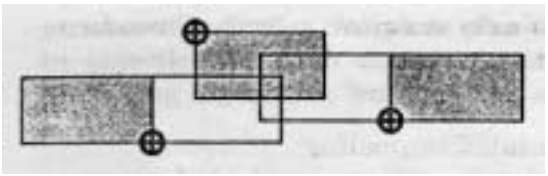

Figura 2.11 – Uma configuração instável do algoritmo do Zoraster

A outra modificação sugerida por ele foi com relação a escolha do valor inicial do multiplicador lagrangeano. Ele sugere 1/8 para rótulos ativos em conflito e -1/16 para

rótulos inativos em conflito. O autor forneceu estes valores baseados em seus experimentos com uma variedade de diferentes mapas, mas valores ótimos são provavelmente dependentes da densidade de rótulos.