ROTULAÇÃO CARTOGRÁFICA DE PONTOS: NOVAS SOLUÇÕES COM UM ALGORITMO GULOSO E UM GRASP

Gildásio Lecchi Cravo[1]
Glaydston Mattos Ribeiro[1,2] **(Orientador)**
Luiz Antonio Nogueira Lorena[2] **(Orientador)**

[1]Faculdade de Aracruz – UNIARACRUZ
Departamento de Ciência da Computação e Informática
R. Prof. Berilo Basílio dos Santos, 180, Vila Rica, Aracruz – ES. CEP 29.190-000.
{lecchi, glaydston}@fsjb.edu.br

[2]Instituto Nacional de Pesquisas Espaciais – INPE
Laboratório Associado de Matemática e Computação Aplicada
Av. dos Astronautas, 1.758, Jardim da Granja, São José dos Campos – SP. CEP 12.227-010.
{glaydston, lorena}@lac.inpe.br

RESUMO

O Problema da Rotulação Cartográfica de Pontos (PRCP) é uma tarefa importante na criação de mapas e diagramas, principalmente em sistemas de informações geográficas. Esta tarefa pode ser visualizada como um problema de otimização combinatória que tem como objetivo, produzir uma rotulação livre de sobreposições. Entretanto, esse problema é de difícil solução e com isso, várias heurísticas/metaheurísticas foram propostas na literatura. Seguindo esta idéia, este trabalho reporta os resultados encontrados por uma heurística gulosa e um GRASP, ambas aqui propostas para o PRCP. Essas duas heurísticas, quando aplicadas às instancias propostas na literatura, apresentaram melhores resultados que os melhores conhecidos, em um tempo computacional baixo.

PALAVRA CHAVES: Heurísticas, Algoritmo Guloso, GRASP. Otimização Combinatória.

ABSTRACT

The Point-Feature Cartographic Label Placement Problem (PFCLP) is an important task in maps and diagrams creation process, mainly in geographic information systems. This problem can be modeled as a combinatorial optimization problem aiming a labeling without overlaps. However, this problem is NP-Hard, thus several heuristics/metaheuritics have been proposed in the literature. Following this idea, this paper reports our results using a greedy heuristic and a GRASP, both proposed for the PFCLP. These heuristics applied over instances proposed in the literature, have presented better solutions than all those reported, in reasonable computational times.

KEY WORDS: Heuristics, Greedy Heuristic, GRASP. Combinatorial Optimization.

1. Introdução

O Problema da Rotulação Cartográfica de Pontos (PRCP) é uma importante etapa na elaboração de mapas e diagramas, principalmente em sistemas de informações geográficas. O problema consiste em colocar os rótulos (textos) dos pontos de tal maneira, que não ocorram sobreposições (Ribeiro e Lorena, 2005). A Figura 1 mostra um exemplo de uma rotulação. As setas indicam as sobreposições (conflitos).

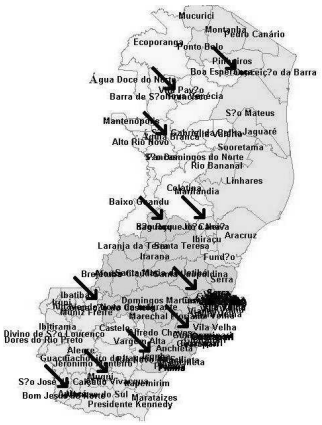

Figura 1. Mapa do Espírito Santo. Exemplo de uma rotulação com conflitos.

Na literatura, o PRCP é classificado como NP–Difícil (Marks e Scheiber, 1991), conseqüentemente, várias heurísticas/metaheurísticas foram propostas. Seguindo esta idéia e considerando que o PRCP pode ser representado por um grafo de conflitos, este artigo reporta os resultados de duas heurísticas propostas para o PRCP: uma heurística gulosa e um GRASP. Para um dado conjunto de instancias, ambas heurísticas apresentaram resultados interessantes quando comparados com os reportados na literatura. O GRASP produziu soluções melhores que as melhores conhecidas, fornecidas pela relaxação Lagrangeana (LagClus) de Ribeiro e Lorena (2006). Os algoritmos são comparados a partir do número obtido de rótulos sem conflitos (rótulos livres).

Sendo assim, esse trabalho está dividido da seguinte maneira. Na Seção 2 é feita uma breve revisão bibliográfica sobre o PRCP. Na Seção 3 é apresentada a heurística gulosa e a sua implementação. A Seção 4 apresenta o GRASP e sua implementação e na Seção 5 são apresentados os resultados computacionais, seguida das conclusões e referências bibliográficas.

2. PRCP e seu Grafo de Conflitos

Na literatura existem três diferentes tipos de rotulação: rotulação de pontos (Ex. cidades), linhas (Ex. rodovias), e área (Ex. estados). No entanto, o problema da rotulação de pontos é considerado o mais difícil de ser resolvido (Wolff, 1999) e o presente trabalho está direcionado para este tipo de problema.

No PRCP, cada ponto tem um conjunto de locais nos quais podem ser colocados os seus rótulos. Esses locais são conhecidos como posições candidatas e são definidos a partir de uma padronização cartográfica. Cada posição recebe um peso indicando a sua preferência, quanto menor for esse peso, maior a preferência pela posição indicada. A Figura 2(a) apresenta uma padronização com quatro posições candidatas proposta por Christensen et al. (1995).

Considerando a padronização mostrada na Figura 2(a) e o problema da Figura 2(b), o PRCP pode ser representado através de um grafo de conflitos, como mostra a Figura 2(c). Sendo assim, seja N o número de pontos a serem rotulados e P o número de posições candidatas para cada ponto. $G= \{V, A\}$ é o grafo de conflitos correspondente, sendo $V= \{v_1, v_2,\ldots, v_{N*P}\}$ o conjunto de posições candidatas e $A=\{(v_i, v_j): i, j \in V, i \neq j\}$ os conflitos potenciais entre posições candidatas (vértices) (Ribeiro e Lorena, 2005).

Para o PRCP existem três abordagens diferentes na literatura. A primeira busca rotular o maior número de pontos sem sobreposições, mesmo que alguns pontos não sejam rotulados, desta

forma, o PRCP pode ser visto como um Problema de Máximo Conjunto Independente (PMCI). Seguindo esta abordagem, Zoraster (1990) formulou matematicamente o PRCP, trabalhando com restrições de conflitos e posições candidatas extras de custo elevado, que são utilizadas quando os pontos não podem ser rotulados sem conflitos. Strijk et al (2000) propuseram uma modelagem de programação inteira binária que usa restrições de corte para reduzir o número de restrições.

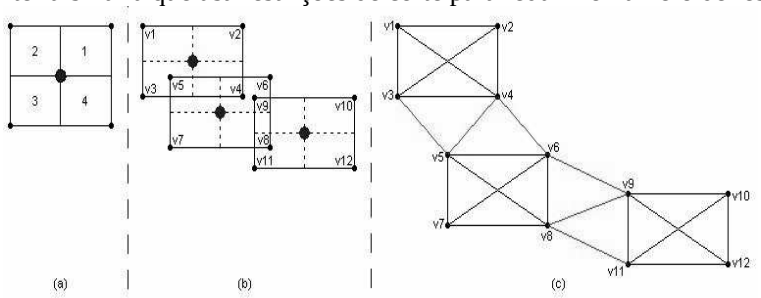

Figura 2. (a) - Padronização cartográfica segundo Christensen et al. (1995). (b) Exemplo de um problema e (c) o grafo de conflitos correspondente.

A segunda abordagem também procura o máximo número de rótulos livres, porém neste caso, todos os pontos devem ser rotulados. Essa abordagem é conhecida como o Problema do Máximo Número de Rótulos Sem Conflitos (PMNRSC) (Ribeiro e Lorena, 2005). Essa abordagem apresenta diversos trabalhos na literatura. Christensen et al. (1995) propuseram um algoritmo guloso com sucessivas otimizações locais e um algoritmo denominado *Discrete Gradient Descent* que considera posições alternativas dos rótulos, porém, esse algoritmo tem dificuldades de escapar de máximos locais. Verner et al. (1997) aplicaram um Algoritmo Genético com e sem máscara. Yamamoto et al. (2002) propuseram um algoritmo de Busca Tabu que obteve bons resultados quando comparados com a literatura. Yamamoto e Lorena (2005) desenvolveram um algoritmo exato para instâncias pequenas do PRCP e aplicaram um Algoritmo Genético Construtivo (CGA), para instâncias de grande porte. Yamamoto et al. (2005) também propuseram um algoritmo denominado FALP que mostrou resultados melhores que o CGA usando a mesmas instâncias propostas em Yamamoto et al. (2002).

A última abordagem considera o Problema da Minimização do Número de Conflitos (PMNC) e foi introduzida primeiramente por Ribeiro e Lorena (2005; 2006). O presente trabalho segue esta abordagem que também rotula todos os pontos, mas minimiza o número de conflitos na solução, se uma solução sem conflitos não é possível. Ribeiro e Lorena (2005; 2006) propuseram dois modelos matemáticos baseados em programação linear inteira binária, e também uma Relaxação Lagrangeana com divisão em *clusters* (LagClus) que apresentou os melhores resultados na literatura para as instâncias propostas por Yamamoto et al. (2002). A segunda formulação proposta pelos autores é apresentada abaixo.

$$v(PMNC) = Min \sum_{i=1}^{N} \sum_{j=1}^{P_i} \left(w_{i,j} x_{i,j} + \sum_{c \in C_{i,j}} y_{i,j,c} \right) \tag{1}$$

Sujeito a:

$$\sum_{j=1}^{P_i} x_{i,j} = 1 \quad \forall i = 1...N \tag{2}$$

$$\left|C_{i,j}\right| x_{i,j} + \sum_{(k,t) \in S_{i,j}} x_{k,t} - \sum_{c \in C_{i,j}} y_{i,j,c} \leq \left|C_{i,j}\right| \quad \forall i = 1...N, \ \forall j = 1...P_i \tag{3}$$

$$x_{i,j}, x_{k,t} \text{ e } y_{i,j,c} \in \{0,1\} \quad \forall\, i = 1...N \quad \forall\, j = 1...P_i \quad c \in C_{i,j} \tag{4}$$

Sendo:

- N o número de pontos a serem rotulados e P_i o conjunto de posições candidatas do ponto i;
- $x_{i,j}$ uma variável binária com $i \in N$ e $j \in P_i$;
- $w_{i,j}$ a preferência cartográfica associada a cada posição candidata. Permite priorizar algumas posições candidatas como descrito na Figura 2(a);
- $S_{i,j}$ um conjunto de par de índices (k,t): $k>i$ de posições candidatas tal que $x_{k,t}$ tem conflito (aresta) com $x_{i,j}$;
- $C_{i,j}$ um conjunto com todos os pontos que contêm posições candidatas em conflito com a posição candidata $x_{i,j}$; e
- $y_{i,j,c}$ uma variável de conflito entre a posição candidata $x_{i,j}$ e o ponto $c \in C_{i,j}$: $c>i$.

A Restrição (2) garante que cada ponto deve ser rotulado com uma posição candidata. A Restrição (3) garante que, se vértices com conflitos forem escolhidos para compor a solução, a função objetivo descrita na Equação (1) será penalizada. E a Restrição (4) indica que todas as variáveis no modelo são binárias.

O grafo produzido pelo PRCP pode ser grande, inviabilizando o processo de solução com a modelagem (1)-(4). Com isso, Wagner et al. (2001) apresentaram um método para reduzir o grafo de conflitos que é composto de três regras que não alteram a solução ótima do problema, essas regras são descritas abaixo:

1. Se um ponto p tem uma posição candidata livre de conflitos, deve-se usar essa posição candidata na solução do problema e todas as outras posições candidatas do ponto p devem ser eliminadas;
2. Se um ponto p tem uma posição candidata p_i que está em conflito somente com uma posição candidata q_k, e o ponto q têm uma posição candidata q_j ($j \neq k$) que está sobreposta somente por uma posição candidata p_l ($l \neq i$), então adicione p_i e q_j na solução do problema e elimine as demais posições candidatas de p e q.
3. Se um ponto p tem somente uma posição candidata p_i restante, e todas as posições candidatas que sobrepõem p_i formam uma clique, então adicione p_i na solução do problema e elimine todas as posições candidatas que sobrepõem p_i.

Essas regras são aplicadas exaustivamente. Para mais detalhes, ver Wagner et al. (2001).

3. Uma Heurística Gulosa para o PRCP

A heurística gulosa proposta é baseada na heurística RSF (*Recursive-Smallest-First*) de Yamamoto et al. (2005) e utiliza a técnica de redução proposta por Wagner et al. (2001). Dado o grafo de conflitos, a heurística RSF inicia escolhendo o vértice de grau mínimo para entrar na solução do problema, em seguida, esse vértice e seus adjacentes são colocados como inativos. A partir da lista de vértices ativos, é recalculado o grau de cada vértice. Isso é necessário porque os graus dos vértices ativos podem ter sido alterados quando a etapa de remover e adicionar na lista de vértices inativos é executada. Em seguida, um novo vértice de grau mínimo é selecionado e o ciclo é, então, repetido. O algoritmo termina quando não houver mais vértices ativos. Todos os vértices escolhidos formam um conjunto independente de vértices. Consequentemente, a RSF pode ser utilizada no PRCP quando este for modelado como um PMCI.

Considerando a abordagem desse trabalho, ou seja, do PRCP como um PMNC, esse algoritmo precisa ser modificado para permitir, se necessário, que vértices em conflitos sejam selecionados. Isso porque a abordagem PMNC rotula todos os pontos mesmo que conflitos sejam inevitáveis. Portanto, o algoritmo guloso proposto considera como critério de ordenação não só o grau de cada vértice, mas também a existência de possíveis conflitos com a solução que está sendo construída.

Uma vez definido o vértice $x_{i,j}$ de menor grau, deve-se tornar inativo todos os outros vértices

do ponto *i*, inclusive $x_{i,j}$, porém, os demais vértices adjacentes do ponto *i* não são removidos como na heurística RSF, mas ao calcular o grau dos vértices ativos, no caso dos vértices não removidos adjacentes ao vértice $x_{i,j}$ selecionado, acrescenta-se uma constante dada por $N*P_i$ ao valor obtido para os respectivos graus. Isso penaliza os vértices adjacentes a $x_{i,j}$, prevenindo que logo na próxima iteração sejam selecionados. No entanto, se no final do processo a lista de vértices ativos apresentar somente vértices penalizados, aquele com o menor número de conflitos com a solução corrente será selecionado.

O algoritmo guloso utiliza duas listas dinâmicas *L1* e *L2*, que representam, respectivamente, a lista de vértices ativos e a lista de vértices inativos. Na definição do vértice de menor grau, uma vez definido o peso de cada vértice (o grau acrescido da constante), faz-se uma ordenação crescente baseada nesses valores e escolhe-se o primeiro vértice. Em seguida, esse vértice e os demais que deverão ser removidos do grafo de conflitos, são inseridos no final da lista de vértices inativos. Assim, dado que os vértices são inseridos na lista de vértices inativos em uma ordem diferente da lista inicial de vértices ativos, ao término da heurística gulosa, a lista de vértices inativos passa a possuir uma ordenação diferente da original.

Experimentos computacionais mostraram que uma reaplicação da heurística gulosa sobre essa lista de vértices inativos transformada em vértices ativos, permite melhorar os resultados. Sendo assim, a heurística gulosa proposta é então mostrada na Figura 3. A constante *N_Iterações* representa o número máximo de repetições do algoritmo guloso.

```
1  i ← 1
2  S* ← {}
3  Sol* ← N*(Maior P_i ∀ i = 1...N)
4  L1 ← {x_1,1, x_1,2, ..., x_N,PN}
5  Aplicar técnica de redução proposta por Wagner et al. (2001)
6  Enquanto (i<=N_Iteracoes) Faça
7    L2 ←{}
8    S ← {}
9    Enquanto (L1 ≠ {}) Faça
10     Ordene crescentemente L1
11     Selecione o primeiro vértice v de L1
12     S ← S + {v}
13     Executar processo de remoção em L1 e atualizar L2
14   Fim Enquanto
15   Solução_S ← Função_Objetivo_Equação_1 (S)
16   Se (Solução_S<Sol*) Então
17     Sol* ← Solução_de_S
18     S* ← S
19   Fim Se
20   L1 ← L2
21   i ← i + 1
22 Fim Enquanto
23 Retornar S* e Sol*
```

Figura 3. Heurística Gulosa proposta

Como a heurística gulosa proposta possui uma forte ligação com a metaheurística GRASP, foi também implementado um GRASP para o PRCP para tentar melhorar os resultados obtidos pela heurística gulosa.

4. Um GRASP para o PRCP

O GRASP, proposto por Feo e Resende (1995), é um método iterativo constituído de duas fases: a de construção e a de busca local. Na fase de construção, é gerada uma lista de candidatos, ordenados de acordo com sua contribuição na função objetivo. Uma solução é então construída elemento a elemento. Essa construção é probabilística, pois a escolha do novo elemento que deverá compor a solução, é feita aleatoriamente a partir de uma lista, denominada lista de candidatos restritos (LCR) que é formada pelo melhores elementos da lista de candidatos. A heurística também é adaptativa, pois a cada iteração da fase de construção os custos (pesos utilizados pela função de ordenação) são atualizados para refletir as mudanças ocasionadas pela seleção do elemento na iteração anterior.

Tendo em vista que essa construção é probabilística, as soluções geradas nesta fase provavelmente não são localmente ótimas. Daí a importância da segunda fase do GRASP, a busca local, na qual se tenta melhorar a solução construída na fase anterior, trabalhando na

vizinhança dessa solução.

O parâmetro de aleatoriedade que determina o tamanho da LCR, ou seja, quantos melhores elementos farão parte da LCR, é um dado importante a ser ajustado em um procedimento GRASP. Prais e Ribeiro (2000) propuseram um GRASP Reativo no qual não se usa um valor fixo para o tamanho da LCR durante a fase construtiva. Para uma revisão completa sobre o GRASP e suas aplicações, veja Festa e Resende (2002) e Resende e Ribeiro (2003). A Figura 4 apresenta o pseudocódigo do GRASP para o PRCP.

```
Dados     : Número de iteracões i_max
Resultado : Solucão x* ∈ X
1 f* ← ∞
2 Rótulos_Livres* ← ∞
3 Para i=1,...,i_max Faça
4  x ← Contrucao_Aleatoria_e_Gulosa()
5  [f(x), Rótulos_Livres] ← BuscaLocal(x)
6  Se f(x) < f* OU ( f(x) = f* E Rótulos_Livres > Rótulos_Livres*) Então
7    f* ← f(x)
8    x* ← x
9    Rotulos_Livres* ← Rótulos_Livres
10  Fim Se
11 Fim Para
```

Figura 4. Algoritmo principal do GRASP proposto para o PRCP.

4.1 Fase Construtiva do GRASP para o PRCP

Para a fase de construção da solução inicial do GRASP, linha 4 da Figura 4, foi utilizada a heurística gulosa apresentada na Seção 3, com duas adaptações. A primeira diz respeito à parte do algoritmo que seleciona o primeiro vértice de *L1* (veja Figura 3, linha 11) que foi modificada de modo que um vértice passasse a ser selecionado aleatoriamente dentre os vértices pertencentes à lista de candidatos restrita. O tamanho dessa lista é definido por uma variável denominada *LCR_tamanho*. A segunda adaptação diz respeito ao número de iterações, nesse caso o algoritmo guloso é executado uma única vez. Deste modo, quando uma solução gulosa é encontrada, procede-se a fase de busca local, conforme linha 5 da Figura 4.

```
Procedimento BuscaLocal(x)
// Seja:
// - FO(Sol) a função objetiva descrita na Equação (1)
// - NumeroRotulosSemConflitos (Sol) uma função que conta o número de rótulos sem conflitos
//   presentes na solução Sol
1 Sol ← x
2 EncontrarNovaSolução ← verdadeiro
3 fCorrente ← FO (Sol)
4 RotulosLivres ← NumeroRotulosSemConflitos (Sol)
5 Enquanto EncontrarNovaSolução Faça
6  EncontrarNovaSolução ← falso
7  Para i=1,…, N Faça
8    PosiçãoCandidataCorrente = Sol[i]
9    Para ∀j∈P_i Faça
10      Se j==Sol[i] Então Continue
11      Sol[i] ← j
12      Se (FO(Sol)<fCorrente) Então
13        fCorrente ← FO(Sol)
14        EncontrarNovaSolução ← verdadeiro
15        MelhorVizinho ← j
16        PontoAlterado ← i
17        RotulosLivres ← NumeroRotulosSemConflitos (Sol)
18        Fim Se
19      Fim Para
20    Sol[i] ← PosiçãoCandidataCorrente
21    Fim Para
22  Se EncontrarNovaSolução Então
23    Sol[PontoAlterado] ← MelhorVizinho
24  Fim Se
25  Fim Enquanto
21 x ← Sol
22 Retorne (FO(Sol) e Rótulos_Livres)
```

Figura 5. Heurística de Busca Local.

4.2. Fase de Busca Local do GRASP para o PRCP

Na segunda fase do GRASP usa-se uma heurística de busca local que trabalha alterando a posição candidata obtida para cada ponto *i* na heurística construtiva, por uma outra posição válida

$j \in P_i$ procurando obter uma solução que reduz a função objetiva descrita na Equação (1). A Figura 5 mostra o pseudocódigo da Busca Local.

Como pode ser visto, o GRASP é dependente do número de iterações e do tamanho da lista de candidatos restrita. Assim, para definir os melhores valores para estes parâmetros, esse GRASP foi aplicado em um conjunto padrão de instâncias gerado e proposto por Yamamoto et al (2002), disponível em *http://www.lac.inpe.br/~lorena/instancias.html*.

5. Resultados Computacionais

A heurística gulosa e o GRASP foram implementados em C++, e os testes computacionais foram executados em um PC com processador Pentium IV (2.80GHz) e 512 MB de memória.

Como realizado em Christensen et al. (1995), Verner et al. (1997), Yamamoto e Lorena (2005) e Ribeiro e Lorena (2005), para todos os problemas não foram consideradas preferências cartográficas, ou seja, custo igual a 1 para todas as posições candidatas, sendo o número dessas posições igual a 4 para todos os pontos. O conjunto de teste possui problemas com 25, 100, 250, 500, 750 e 1000 pontos. Sendo que o problema de 25 pontos possui 8 instâncias, enquanto os demais apresentam 25 instâncias cada um.

A Tabela 1 e 2 apresenta os resultados computacionais médios para cada problema obtidos pela Heurística Gulosa, pela LagClus de Ribeiro e Lorena (2006) e pelo GRASP. O número máximo de iterações utilizado no algoritmo guloso foi igual a 35. Experimentos realizados com valores maiores que 35 não apresentaram melhoria significativa. Os parâmetros utilizados pelo GRASP foram 5 e 6 para *LCR_tamanho* e i_{max}=100. Esses valores foram os que apresentaram os melhores resultados.

N.º Pontos	Heurística Gulosa		LagClus (Ribeiro e Lorena, 2006)		GRASP			
					Média de 10 Execuções		Melhor Média das 10 Execuções	
	Rótulos Livres (%)	Tempo (s)	Rótulos Livres (%)	Tempo* (s)	Rótulos Livres (%)	Tempo (s)	Rótulos Livres (%)	Tempo (s)
100	100,00	0,001	100,00	0,12	100,00	0,000	100,00	0,000
250	100,00	0,013	100,00	0,12	100,00	0,002	100,00	0,000
500	99,62	0,196	99,67	0,40	99,67	0,362	99,67	0,346
750	96,99	1,738	97,65	53,84	97,70	5,214	97,74	5,216
1000	90,74	10,441	91,42	3445,40	92,02	26,820	92,07	26,718

Tabela 1. Comparação dos resultados da heurística gulosa, LagClus e do GRASP com *LCR_tamanho*=5. *Máquina utilizada: Pentium IV (2.66GHz) com 512 MB de memória.

N.º Pontos	Heurística Gulosa		LagClus (Ribeiro e Lorena, 2006)		GRASP			
					Média de 10 Execuções		Melhor Média das 10 Execuções	
	Rótulos Livres (%)	Tempo (s)	Rótulos Livres (%)	Tempo* (s)	Rótulos Livres (%)	Tempo (s)	Rótulos Livres (%)	Tempo (s)
100	100,00	0,001	100,00	0,12	100,00	0,000	100,00	0,000
250	100,00	0,013	100,00	0,12	100,00	0,000	100,00	0,000
500	99,62	0,196	99,67	0,40	99,67	0,370	99,67	0,350
750	96,99	1,738	97,65	53,84	97,69	5,220	97,72	5,220
1000	90,74	10,441	91,42	3445,40	92,06	26,790	92,20	26,690

Tabela 1. Comparação dos resultados da heurística gulosa, LagClus e do GRASP com *LCR_tamanho*=6. *Máquina utilizada: Pentium IV (2.66GHz) com 512 MB de memória.

Note que o GRASP proposto melhora os resultados do guloso e da LagClus. Os tempos computacionais do GRASP permanecem reduzidos em relação ao algoritmo guloso, e bem inferiores aos da LagClus. A Tabela 3 faz uma comparação dos resultados obtidos neste trabalho com os apresentados na literatura. Observe que o GRASP apresenta resultados melhores que várias metaheurísticas bem conhecidas.

6 Conclusões

Este trabalho apresentou uma heurística gulosa e um GRASP para o problema da rotulação cartográfica de pontos. Considerando o tempo reduzido e a qualidade das rotulações geradas, tanto o GRASP quanto a heurística gulosa, podem ser utilizados embutidos em aplicações comerciais de mapas, ou dependendo talvez do propósito, em aplicações *Web*. Sob o ponto de vista de otimização, o GRASP apresentou resultados melhores que todos os informados na

literatura, superando metaheurísticas conhecidas como *Simulated Annealing*, Busca Tabu e Algoritmos Genéticos.

Algoritmos	Rótulos Livres (%)				
	Problemas				
	100	250	500	750	1000
$GRASP_{Melhor,\ RCL\_Tamanho\ =\ 6}$	100,00	100,00	99,67	97,72	92,20
$GRASP_{Médio,\ RCL\_Tamanho\ =\ 6}$	100,00	100,00	99,67	97,69	92,06
$GRASP_{Melhor,\ RCL\_Tamanho\ =\ 5}$	100,00	100,00	99,67	97,70	92,02
$GRASP_{Médio,\ RCL\_Tamanho\ =\ 5}$	100,00	100,00	99,67	97,74	92,07
LagClus (Ribeiro e Lorena, 2006)	100,00	100,00	99,67	97,65	91,42
Algoritmo Guloso Prosposto	100,00	100,00	99,62	96,99	90,74
AGC_{Melhor} (Yamamoto e Lorena, 2005)	100,00	100,00	99,60	97,10	90,70
FALP (Yamamoto et al., 2005)	100,00	100,00	99,50	96,70	90,12
$AGC_{Médio}$ (Yamamoto e Lorena, 2005)	100,00	100,00	99,60	96,80	90,40
Busca Tabu (Yamamoto et al, 2002)	100,00	100,00	99,30	96,80	90,00
AG com máscara (Verner et al, 1997)	100,00	99,98	98,79	95,99	88,96
AG (Verner et al, 1997)	100,00	98,40	92,59	82,38	65,70
Simulated Annealing (Christensen et al, 1995)	100,00	99,90	98,30	92,30	82,09
Zoraster (Zoraster, 1990)	100,00	99,79	96,21	79,78	53,06
3-opt Gradient Descent (Christensen et al, 1995)	100,00	99,76	97,34	89,44	77,83
2-opt Gradient Descent (Christensen et al, 1995)	100,00	99,36	95,62	85,60	73,37
Gradient Descent (Christensen et al, 1995)	98,64	95,47	86,46	72,40	58,29
Algoritmo Guloso (Christensen et al, 1995)	95,12	88,82	75,15	58,57	43,41

Tabela 3. Comparação dos resultados com a literatura.

Referências

Christensen J., Marks J., Shieber S. (1995), An Empirical Study of Algorithms for Point-Feature Label Placement, *ACM Transactions on Graphics*, 14(3), 203-232.

Feo, T.A., Resende, M. G. C. (1995), Greedy Randomized Adaptive Search Procedures, *Journal of Global Optimization*, 6, 109-133.

Festa, P. Resende, M. G. C., GRASP: An Annotated bibliography, em Ribeiro, C. C. e Hansen, P. (Eds.). *Essays and Surveys on Metaheuristics*, Kluwer Academic Publishers, 325-367, 2002

Marks, J. e Scheiber, S., The computational complexity of cartographic label placement, *Center for Research Computing Technology*, Harvard University, 1991.

Prais M., Ribeiro, C. C. (2000), Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment, *INFORMS Journal on Computing*, 12, 164-176.

Resende, M. G. C., Ribeiro, C. C., Greedy randomized adaptive search procedures, em Glover, F. and Kochenberger, G. (Eds.), *Handbook of Metaheuristics*, Kluwer Academic Publishers, 219-249, 2003.

Ribeiro G. M., Lorena, L. A. N. (2005), Heuristics for cartographic label placement problems, *Computers and GeoSciences*, In Press.

Ribeiro G. M., Lorena, L. A. N. (2006), Lagrangean relaxation with clusters for point-feature cartographic label placement problems, *Computers and Operations Research*. To Appear.

Strijk T., Verweij B., Aardal K. (2000)**,** Algorithms for maximum independent set applied to map labeling. Disponível em ftp://ftp.cs.uu.nl/pub/RUU/CStechreps/CS-2000/2000-22.ps.gz.

Verner O. V., Wainwright R. L., Schoenefeld D. A. (1997), Placing text labels on maps and diagrams Using Genetic Algorithms with Masking, *INFORMS Journal on Computing*, 9, 266-275.

Yamamoto M., Camara G., Lorena L. A. N. (2002), Tabu search heuristic for point-feature cartographic label placement, *GeoInformatica and International Journal on Advances of Computer Science for Geographic Information Systems*, 6(1), 77-90.

Yamamoto M., Lorena L. A. N., A constructive genetic approach to point-feature cartographic label placement, em Ibaraki, T., Nonobe, K. and Yagiura, M. (Eds.), *Metaheuristics: Progress as Real Problem Solvers*, Kluwer Academic Publishers, 285-300, 2005.

Yamamoto, M., Camara, G., Lorena, L. A. N., (2005) Fast point-feature label placement for real time screen maps. *GEOINFO 2005 - VII Brazilian Symposium on GeoInformatics* - Campos do Jordão – São Paulo, Brazil. Disponível em http://www.lac.inpe.br/~lorena/missae/sbc_Missae.pdf.

Wagner, F., Wolff, A., Kapoor, V., Strijk, T. (2001), Three rules suffice for good label placement, *Algorithmica*, 30, 334-349.

Wolff, A., Automated label placement in theory and practice, Tese de Doutorado, Fachbereich Mathematik und Informatik, Freie Universität, Berlin, pp. 153, 1999.

Zoraster S. (1990), The solution of large 0-1 integer programming problems encountered in automated cartography, *Operations Research*, 38(5), 752-759.