IMPLEMENTAÇÃO E TESTE DE CRITÉRIOS DE PARADA PARA HEURÍSTICAS GRASP

Layni A. Neves
Universidade Federal do Estado do Rio de Janeiro - UNIRIO
Avenida Pasteur, 458, Urca, Rio de Janeiro, RJ, Brasil
layni.neves@uniriotec.br

Adriana C. F. Alvim
Universidade Federal do Estado do Rio de Janeiro - UNIRIO
Avenida Pasteur, 458, Urca, Rio de Janeiro, RJ, Brasil
adriana@uniriotec.br

Mauricio G. C. Resende
AT&T Labs Research
180 Park Avenue, Florham Park, NJ, USA
mgcr@research.att.com

RESUMO

O uso de algoritmos baseados em metaheurísticas é de relevante importância para a resolução de diversos problemas de otimização combinatória. Estes algoritmos procuram uma solução viável de boa qualidade quando é difícil encontrar a solução exata devido à complexidade computacional do problema. Uma das questões fundamentais em heurísticas baseadas em metaheurísticas é equilibrar tempo de processamento e qualidade da solução. Este trabalho apresenta a implementação e teste de critérios de parada baseados em estatística bayesiana em heurísticas baseadas na metaheurística GRASP. O objetivo é escolher um critério de parada que resulte em um tempo de processamento curto da heurística e ao mesmo tempo alcance uma solução de boa qualidade. No experimento computacional, foram comparados, em cinco heurísticas GRASP previamente publicadas, os resultados obtidos com a versão original usando como critério de parada um número fixo de iterações com os resultados obtidos com a versão que usa o critério de parada bayesiano.
PALAVRAS CHAVE. Metaheurísticas. Estatística Bayesiana. GRASP.
Área Principal: MH - Metaheurísticas

ABSTRACT

The use of algorithms based on the GRASP metaheuristic is relevant for the solution of many combinatorial optimization problems. These algorithms find a feasible solution of good quality when is difficult to find an exact solution because of the problem's computational complexity. One of the key issues in designing heuristics is the balance between processing time and solution quality. This thesis presents the implementation and testing of Bayesian stopping rules in GRASP heuristics. The objective is to choose a stopping rule that results in a short running time of the heuristic while at the same time achieving good solution quality. In the computational experiment, we compared, on five previously published GRASP heuristics, the results of original version using a stopping rule based on a fixed number of iterations with the results of versions that use Bayesian stopping rules.
KEYWORDS. Metaheuristics. Bayesian Statistic. GRASP.
Main Area: MH - Metaheuristics

1 Introdução

Um importante desafio no projeto de buscas heurísticas é determinar quando parar a busca e declarar que a melhor solução encontrada é próxima da solução ótima. Não conhecer o valor da solução ótima torna o critério de parada mais desafiante. Uma abordagem possível seria executar a heurística infinitamente – quanto maior o número de execuções da heurística, maior a probabilidade da melhor solução encontrada ser a solução ótima. Entretanto, esta abordagem não é desejável pois não contempla outro objetivo importante das heurísticas, a eficiência.

Neste trabalho é explorado o *trade-off* entre o tempo de processamento e a qualidade da solução na determinação da parada das heurísticas baseadas na metaheurística GRASP. Foi incorporado um critério de parada baseado em estatística bayesiana em cinco heurísticas baseadas na metaheurística GRASP e comparado estas novas heurísticas com as suas versões originais com parada determinada por um número fixo de iterações.

Uma metaheurística GRASP (Feo e Resende (1989,1995), Resende e Ribeiro (2003,2009)), do inglês, *greedy randomized adaptive search procedure*, aplica repetidamente buscas locais iniciando em soluções construídas através de um algoritmo guloso aleatório. O melhor ótimo local encontrado em todas as buscas locais é retornado como a solução da heurística. A Figura 1 mostra o pseudo-código para um GRASP genérico. A linha 2 no pseudo-código faz referência ao critério de parada, o foco deste trabalho.

```
begin GRASP
1  x* ← ∞;
2  enquanto critério de parada não é satisfeito faça
3      x ← Construcao_Solucao(·);
4      x ← Busca_Local(x);
5      se f(x) < f(x*) então
6          x* ← x;
7      fim-se
8  fim-enquanto;
end
```

Figura 1: GRASP

2 Critério de parada proposto por Boender e Rinnooy Kan

Foi demonstrado em (Aiex, Resende e Ribeiro (2002), Bartkuté, Felinskas e Sakalauskas (2006), Bartkuté e Sakalauskas (2007,2009), Boender e Rinnooy Kan (1987,1991), Grigaitis, Bartkuté e Sakalauskas (2007), Hart (1998), Orsenigo e Vercellis (2006)) que o uso da teoria estatística pode determinar um critério de parada mais inteligente do que o critério que simplesmente limita um número fixo de iterações para um algoritmo. Existem abordagens que utilizam conceitos teóricos da estatística clássica, outras que se fundamentam na análise empírica dos dados observados, e uma terceira classe que baseia-se na teoria da probabilidade bayesiana, a qual, de certa forma, agrega a teoria e a análise dos dados.

A abordagem bayesiana é fundamentada na probabilidade condicional; ou seja, dado que a distribuição *a priori* $P(A)$ do evento A é conhecida, pode-se derivar a probabilidade *posteriori* do evento B dado A, $P(B|A)$, usando o Teorema de Bayes.

Na abordagem bayesiana é possível expressar algum conhecimento prévio de A através da distribuição *a priori*. Dados resultados experimentais da aplicação de uma determinada heurística a

um determinado problema, pode-se derivar a distribuição *posteriori*, que reflete o caminho no qual o conhecimento prévio (*a priori*) afeta os resultados do experimento. Este trabalho propõe o critério de parada com abordagem bayesiana para heurísticas baseadas na metaheurística GRASP.

O método GRASP (Feo e Resende (1989,1995), Resende e Ribeiro (2003,2009)) executa uma busca local a partir de vários pontos iniciais em um subconjunto de uma amostra aleatória com distribuição uniforme. Então, pode-se contar o número de diferentes ótimos locais W encontrados em n repetições da busca local e determinar a distribuição da variável W. Segundo Boender e Rinnooy Kan (1987) um conjunto de diferentes ótimos locais é uma amostra de distribuição multinomial, porque caracteriza o número de vezes que um ótimo local distinto foi encontrado. Logo, tendo-se a distribuição multinomial *a priori* do conjunto de ótimos locais, condicionada a um conjunto de ótimos locais observados, pode-se estimar o número de ótimos locais ideal, considerando-se uma função de perda que agrega a perda de tempo de execução e a perda de qualidade da solução.

Boender e Rinnooy Kan (1987) propuseram algumas regras de parada bayesiana para métodos com múltiplas inicializações para otimização combinatória. Boender e Rinnooy Kan (1987) introduzem quatro funções de perda, denominadas L_1, L_2, L_3 e L_4. A perda de terminação L_1 é igual a uma constante fixa se a amostragem é parada antes de todos os ótimos locais serem descobertos. A perda da terminação L_3 é proporcional a fração de ótimos locais não observados. A perda da terminação L_4 é proporcional ao volume total relativo das regiões de atração não observadas. Não será abordado neste trabalho a função L_2 em que a perda de terminação é proporcional ao número de ótimos locais não observados. Segundo Boender e Rinnooy Kan (1987), para as regras de parada baseadas nesta função de perda, não é possível determinar que a partir de uma determinada iteração a execução do programa vai parar. Dada uma amostra de tamanho n com w diferentes ótimos locais observados (denotado por (n,w)), estas três funções consideram o custo da amostragem. Isto é, o custo quando a execução é parada antes de todos os ótimos locais terem sido encontrados (*perda de terminação*) e o custo de uma nova execução de busca local (*perda de execução*). A perda depois de $n' > n$ observações é uma variável aleatória $E(L_j|(n',W))$. Dado (n,w), é possível calcular o valor esperado condicional da perda *posteriori* de $n+1$ observações de acordo com $E(E(L_j|(n+1,W))|(n,w))$. A estratégia compara a perda *posteriori* corrente com a perda *posteriori* esperada de outra observação assumindo-se que *a melhor estratégia é adotada daí em diante*. Para esta abordagem ser possível, deve-se encontrar um valor n_j^* tal que para todos os pares (n,w) com $n \geq n_j^*$ a seqüencia de perdas *posteriori* nunca decresce se uma nova observação é realizada. Sabe-se então que para todos os pares (n,w) com $n = n_j^*$ a decisão ótima é parar, e é possível calcular a perda *posteriori* para $E(L_j|(n,w))$.

A seguir apresenta-se o *framework* `GRASP_BSR`, do inglês *GRASP with Bayesian Stopping Rule*, que implementa o critério de parada proposto por Boender e Rinnooy Kan (1987) aplicado a metaheurísticas GRASP. O *framework* `GRASP_BSR` pode ser visto na Figura 2.

Os parâmetros de entrada são: função de perda $L_j, j \in \{1,3,4\}$, controle c e um conjunto P de parâmetros GRASP. O parâmetro c é usado para calcular a perda posteriori $E(L_j|(n,w))$ e calcular o limite superior de iterações n_j^* (Boender e Rinnooy Kan (1987)).

A seguir, relata-se o significado das variáveis usadas. A variável n representa o número corrente de iterações GRASP; a variável w indica o número de diferentes ótimos locais encontrados; o conjunto O armazena os w diferentes ótimos locais encontrados durante a busca; a variável n_j^* é o limite superior de iterações definido pelo critério de parada j; a variável *melhor_sol* armazena a melhor solução encontrada até o momento; a variável *parada* indica se a busca deve parar ou continuar; a variável s armazena a solução corrente e as variáveis x_1 e x_2 são usadas para armazenar os valores das perdas posteriori.

As variáveis são inicializadas nas linhas 1-6. Na linha 4 é chamada a função que calcula o limite superior de iterações n_j^* (Figura 3). A variável que guarda a melhor solução até o momento é atualizada na linha 5. O número de iterações corrente n é atualizado na linha 8. O procedimento `Uma_Iteração_GRASP` chama uma iteração GRASP completa (fases construtiva e de busca local)

```
Framework GRASP_BSR(L_j, j ∈ {1,3,4}; c, P);
1   w ← 0;
2   n ← 0;
3   O ← ∅
4   n*_j ← Calcula_limite_sup_iteracao(c, j);
5   melhor_sol ← Inicializa_Melhor_Solucao;
6   parada ← FALSO;
7   enquanto (NÃO parada) faca
8       n ← n + 1;
9       s ← Uma_Iteração_GRASP(P);
10      Atualiza_Melhor_Solucao(s, melhor_sol);
11      se (s ∉ O) então
12          w ← w + 1;
13          O ← O ∪ {s};
14      fim-se
15      se (n ≥ w + 2) então
16          x1 ← E(E(L_j|(n + 1, w))|(n, w));
17          x2 ← E(L_j|(n, w));
18      fim-se
19      se ((x1 ≥ x2 e n ≥ w + 2) ou n ≥ n*_j) então
20          parada ← VERDADEIRO;
21      fim-se
22  enquanto
23  retorna melhor_sol
24  fim-GRASP_BSR.
```

Figura 2: O *framework* do procedimento GRASP_BSR usado para implementar os critérios de parada proposta por Boender e Rinnooy Kan (1987), aplicado a metaheurísticas GRASP.

de uma específica implementação GRASP para o problema tratado com o conjunto P de parâmetros, e retorna a solução ótima local s. A melhor solução encontrada até o momento é atualizada na linha 10. Na linha 11, verifica-se se a solução ótima local s não existe em O, o conjunto de ótimos locais. Caso verdadeiro, o número w de diferentes ótimos locais w é atualizado na linha 12 e a solução s é incluída no conjunto O na linha 13. Na linha 15 é testado se a iteração corrente n é maior ou igual à condição necessária para o teste do critério de parada ($w+2$). Se verdadeiro, a perda posteriori esperada de $n+1$ iterações (observações) x_1 é calculada na linha 16 e a perda posteriori corrente x_2 é calculada na linha 17. A busca pára se $x_1 \geq x_2$ (para iteração corrente n maior ou igual à condição necessária para o teste do critério de parada ($w+2$)) ou se o número de iterações n é maior ou igual ao limite superior de iterações n_j^* (linhas 19-21). A melhor solução encontrada na busca é retornada na linha 23.

Para finalizar, deve-se considerar o custo adicional de se contabilizar os diferentes ótimos locais w (linhas 11 a 14 do pseudo código). Considerando-se uma solução representada como um conjunto de m elementos (o tamanho do problema), verificar a igualdade de duas soluções custa $O(m)$. Como, a cada iteração, esta verificação é realizada para w ótimos locais e, no pior caso, $w = n_j^*$, tem-se que a complexidade de uma iteração da busca é adicionada em $O(n_j^* m)$, sendo o valor de n_j^* calculado em função da constante c (Figura 3). É importante salientar que o *framework* proposto é genérico e duas dadas soluções são consideradas dois ótimos locais distintos quando suas respectivas configurações

```
Função Calcula_limite_sup_iteracao(c, j);
1  caso j igual a
2      1: n* = c1 + 1 − √(4c + 1)
3      3: n* = c/4
4      4: n* = c/3
5  fim-caso
6  retorna n*
7  fim-Calcula_limite_sup_iteracao.
```

Figura 3: Pseudo código da função que calcula o limite superior de iterações n_j^*.

são distintas. Casos especiais como, por exemplo, o de soluções simétricas que acontece, por exemplo, em coloração de vértices, necessita um tratamento diferenciado.

3 Experimentos Computacionais

Este experimento preliminar investiga se o uso do *framework* `GRASP_BSR` é uma boa estratégia comparada a heurísticas GRASP com parada determinada por um número fixo de iterações. Para dado valor c_j uma simples tabela resume a estratégia de parada ótima para as funções de perda L_1, L_3 e L_4, onde o limite superior de iterações de cada função de perda L_j é dado pelo valor da variável n_j^*. A execução de um programa pára ou porque a perda posteriori esperada de $n+1$ iterações é maior ou igual a perda posteriori corrente ou porque o número de iterações alcançou o limite superior n_j^*. Desta forma, considerando que se pretende executar, no máximo, um dado número de n iterações, deseja-se investigar o comportamento do *framework* `GRASP_BSR` para cada um dos critérios de parada fazendo $n_j^* = n$. Para tal, foi idealizado o experimento que compara a qualidade das soluções obtidas pelo *framework* `GRASP_BSR` com aquelas obtidas pelo GRASP original com n iterações para $n \in \{1.000, 10.000, 100.000\}$. O objetivo é perceber se dado um limite máximo de iterações que o usuário possa esperar, o que é mais vantajoso: executar este limite como número fixo de iterações ou utilizar um critério de parada inteligente que demore até no máximo este limite. A escolha favorecerá o critério de parada inteligente se a qualidade da sua solução for igual ao do número fixo de iterações com um número de iterações menor. A Tabela 1 relaciona, para cada função de perda L_j, os valores do parâmetro c_j com os valores do limite superior de iterações n_j^* usados no experimento. A métrica usada para avaliar a qualidade da solução q será calculada como a diferença relativa entre o valor da solução obtida em `GRASP_BSR` (s_{GRASP_BSR}), e o valor da melhor solução conhecida (s_{melhor}). A qualidade q é um índice de escala de 100 pontos, onde quanto mais próximo de 100, melhor a qualidade da solução:

$$q = 100 - \left(\frac{|s_{GRASP\_BSR} - s_{melhor}|}{s_{melhor}} \times 100\right). \quad (1)$$

As cinco heurísticas GRASP usadas nos experimentos são GRASP para o problema de conjunto independente máximo (`gmis` de Resende, Feo e Smith (1998)); GRASP para o problema quadrático de atribuição (`gqapd` de Resende, Pardalos e Li (1996)); GRASP para o problema de satisfabilidade máxima ponderada (`maxsat` de Resende, Pitsoulis e Pardalos (1999)); GRASP para o problema de planarização de grafos (`gmpsg` de Ribeiro e Resende (1999)) e GRASP para o problema de recobrimento (`sc` de Feo e Resende(1989)). O projeto experimental totaliza 15 abordagens do tipo "heurística versus critério de parada", daqui em diante chamada de "heurística-parada". A heurística para o problema quadrático de atribuição foi testada com 18 instâncias e as demais heurísticas foram

Tabela 1: Relação entre c_j e n_j^*.

	$n_j^* \cong$	1.000	10.000	100.000
			c_j	
L_1	$c_j + 1 - \sqrt{4c+1}$	1.000	10.000	100.000
L_3	$c_j/4$	4.000	40.000	400.000
L_4	$c_j/3$	3.000	30.000	300.000

testadas com 5 instâncias. Para cada uma das 114 combinações "heurística-parada-instância" foram feitas 10 execuções independentes.

A Tabela 2 sumariza os resultados. Para cada uma das cinco heurísticas e para cada um dos três valores de n_j^* existe uma linha na tabela com as médias dos resultados, a quarta linha do grupo mostra a média dos três valores de n_j^* para a mesma combinação heurística e critério de parada. O último grupo de quatro linhas mostra as médias das cinco heurísticas. A primeira coluna indica o limite superior de iterações n_j^*. As próximas duas colunas apresentam os resultados para heurística GRASP original com um número fixo de n iterações. Para cada critério de parada L_1, L_3 e L_4 existe um grupo com quatro colunas que apresentam os resultados para `GRASP_BSR`. Para todos os grupos, a coluna *w* mostra o número de diferentes ótimos locais, e a coluna *mi* reporta o número da iteração onde a melhor solução foi encontrada. Para os últimos três grupos, a coluna *n* representa o número de iterações executadas e a coluna *q* representa a qualidade da solução.

Considere o gráfico comparativo da Figura 4 dividido em 12 regiões. O eixo da abscissa (horizontal) indica o número *n* de iterações executadas e o eixo da ordenada (vertical) indica a qualidade *q* da solução encontrada até esta iteração. Em relação à qualidade da solução, soluções na área "superior" do gráfico são aquelas cuja qualidade encontra-se na faixa entre 99 e 100. Soluções cuja qualidade encontra-se abaixo de 97 estão localizadas na área "inferior" e soluções cuja qualidade está na faixa entre 97 e 99 estão localizadas na área "média". Em relação ao número de iterações executadas (tempo de processamento), soluções muito rápidas (com até 20% do tempo estipulado pelo limite superior de iteração) estão localizadas na área "extremo esquerda", soluções muito lentas (acima de 80% do tempo estipulado pelo limite superior de iteração) estão localizadas na área "extremo direita" e soluções intermediárias podem estar localizadas nas áreas "esquerda" (entre 20% e 50% do tempo estipulado pelo limite superior de iteração) ou "direita" (entre 50% e 80% do tempo estipulado pelo limite superior de iteração). O cenário ideal é aquele onde as soluções estão localizadas na área "superior extremo esquerda", em oposição ao cenário menos favorável que é aquele onde as soluções estão localizadas na área "inferior extremo direita". Considera-se que as "melhores soluções" encontram-se nas quatro áreas mais sombreadas do gráfico (região "sombreada"), a saber: "superior extremo esquerda", "superior esquerda", "média extremo esquerda" e "média esquerda".

Os gráficos das Figuras 5 a 7 mostram um panorama comparativo para todas os critérios de parada aplicados às cinco heurísticas GRASP. Em cada gráfico existem 18 pontos, 15 deles representam a média dos resultados de cada combinação "heurística-parada" correspondente às primeiras 15 linhas da Tabela 2 (colunas $n_j^* \in \{1.000, 10.000, 100.000\}$) e os demais três pontos são relativos à média geral de cada critério de parada relativos às últimas três linhas da Tabela 2. Considerando-se o critério número de iterações executadas, percebe-se que a heurística indentificada por `maxsat` possui nos três gráficos (Figuras 5 a 7) seis pontos concentrados na região "superior extremo direita" e três pontos na região "superior direita". Isto significa que a heurística executou um número alto de iterações. Isto indica que se o programa continuasse a execução, ele provavelmente continuaria encontrando novos ótimos locais diferentes. O mesmo acontece com a heurística mais lenta, a `gmpsg`, com oito pontos na área "superior extremo direita" e um ponto na "superior direita". A heurística mais rápida é a identificada por `gmis` (seis pontos na região "extremo esquerda"). A

Tabela 2: Resumo com as médias dos resultados.

	$n = n^*$		L_1				L_3				L_4			
n^*	w	mi	n	w	mi	q	n	w	mi	q	n	w	mi	q
GRASP para o problema de conjunto independente máximo (programa `gmis`)														
1.000	250,1	25,2	34,7	13,7	3,3	97,2	677,0	242,6	25,2	100,0	457,2	198,6	25,2	100,0
10.000	822,1	221,9	82,3	13,9	3,3	97,1	3864,0	662,2	25,2	99,9	1870,7	469,8	25,2	99,9
100.000	2091,9	1509,2	231,7	13,9	3,3	97,0	17689,4	1378,3	1509,2	100,0	6286,1	894,5	221,9	99,8
médias	1054,7	585,4	116,2	13,8	3,3	97,1	7410,1	761,0	519,9	100,0	2871,3	521,0	90,8	99,9
GRASP para o problema de satisfabilidade máxima ponderada (programa `maxsat`)														
1.000	1000,0	449,8	938,0	938,0	444,8	100,0	1000,0	1000,0	449,8	100,0	1000,0	1000,0	449,8	100,0
10.000	9998,8	5008,7	8976,1	8975,7	4199,0	100,0	9135,3	9134,9	4199,0	100,0	9135,3	9134,9	4199,0	100,0
100.000	99945,4	51298,4	78376,8	78376,2	40142,2	99,9	78805,9	78805,2	40142,2	99,9	78805,9	78805,2	40142,2	99,9
médias	36981,4	18919,0	29430,3	29430,0	14928,7	100,0	29647,1	29646,7	14930,3	100,0	29647,1	29646,7	14930,3	100,0
GRASP para problema de planarização de grafos (programa `gmpsg`)														
1.000	1000,0	409,5	938,0	938,0	365,9	99,9	1000,0	1000,0	409,5	100,0	1000,0	1000,0	409,5	100,0
10.000	9995,2	3543,3	8360,2	8359,8	3347,3	100,0	8519,4	8519,0	3491,1	100,0	8519,4	8519,0	3491,1	100,0
100.000	99533,2	27472,7	79784,5	79784,0	30204,5	99,9	80276,7	80276,2	30204,5	99,9	80276,7	80276,2	30204,5	99,9
médias	36842,8	10475,2	29694,2	29693,9	11305,9	99,9	29932,0	29931,7	11368,4	100,0	29932,0	29931,7	11368,4	100,0
GRASP para o problema de recobrimento (programa `sc`)														
1.000	810,6	240,6	719,3	718,7	203,6	99,9	847,7	771,3	220,9	99,9	808,7	771,3	220,9	99,9
10.000	7987,0	1455,8	6102,7	6101,9	1260,3	99,7	6507,2	6229,2	1260,3	99,7	6327,6	6229,2	1260,3	99,7
100.000	78246,5	6500,8	59843,5	59842,7	6292,1	99,7	61151,9	60238,5	6305,2	99,7	60478,0	60249,6	6305,2	99,7
médias	29014,7	2732,4	22221,8	22221,1	2585,3	99,8	22835,6	22413,0	2595,5	99,8	22538,1	22416,7	2595,5	99,8
GRASP para o problema quadrático de atribuição (programa `gqapd`)														
1.000	939,6	434,6	757,4	756,9	361,2	97,3	899,4	848,9	410,2	97,8	883,8	844,7	411,9	97,8
10.000	8808,6	3541,8	6055,3	6054,3	2843,5	98,5	6980,1	6387,3	2885,7	99,0	6899,3	6539,1	2936,7	99,1
100.000	82035,3	21528,5	30590,9	30589,3	12319,5	98,7	43670,0	35313,8	12428,8	99,4	39540,1	34698,8	12428,8	99,4
médias	30594,5	8501,3	12467,9	12466,8	5174,7	98,2	17183,2	14183,3	5241,6	98,7	15774,6	14027,6	5258,8	98,8
Resumo das cinco heurísticas														
1.000	800,0	312,0	677,5	673,1	275,8	98,9	884,8	772,5	303,1	99,5	829,9	762,9	303,5	99,5
10.000	7522,3	2754,3	5915,3	5901,1	2330,7	99,1	7001,2	6186,5	2372,2	99,7	6550,5	6178,4	2382,5	99,7
100.000	72370,5	21661,9	49765,5	49721,2	17792,3	99,0	56318,8	51202,4	18118,0	99,8	53077,4	50984,8	17860,5	99,7
médias	26897,6	8242,7	18786,1	18765,1	6799,6	99,0	21401,6	19387,1	6931,1	99,7	20152,6	19308,7	6848,8	99,6

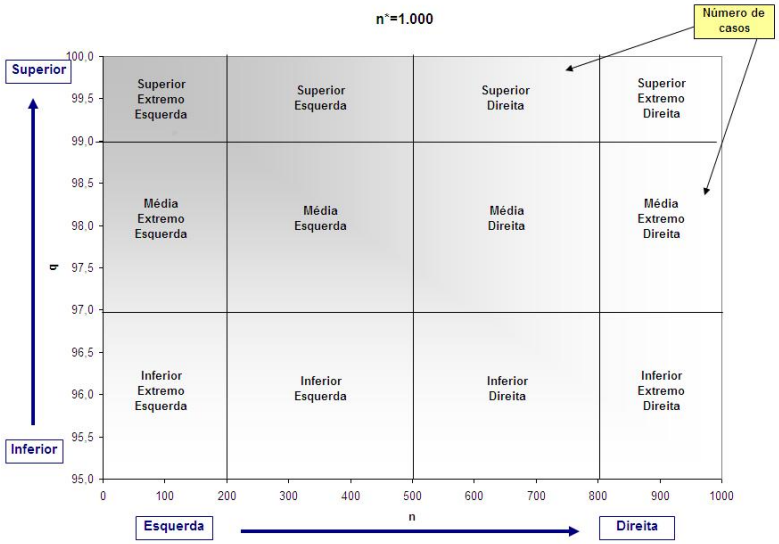

Figura 4: Gráfico comparativo dividido em 12 regiões.

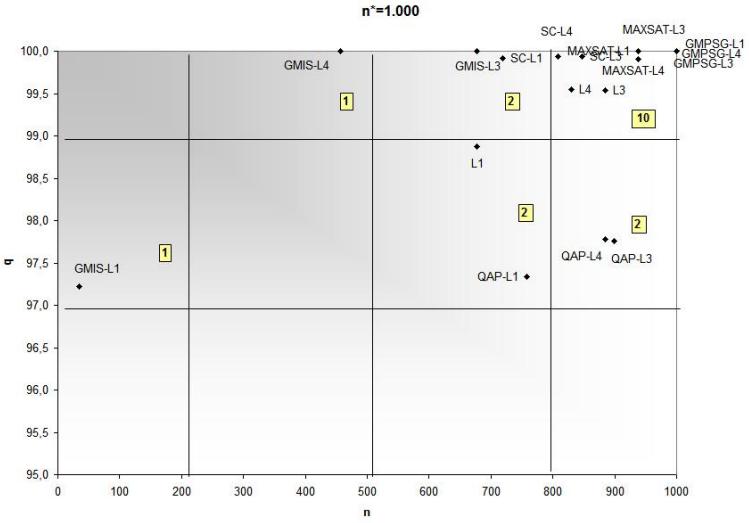

Figura 5: Comparação com $n_j^* \cong 1.000$.

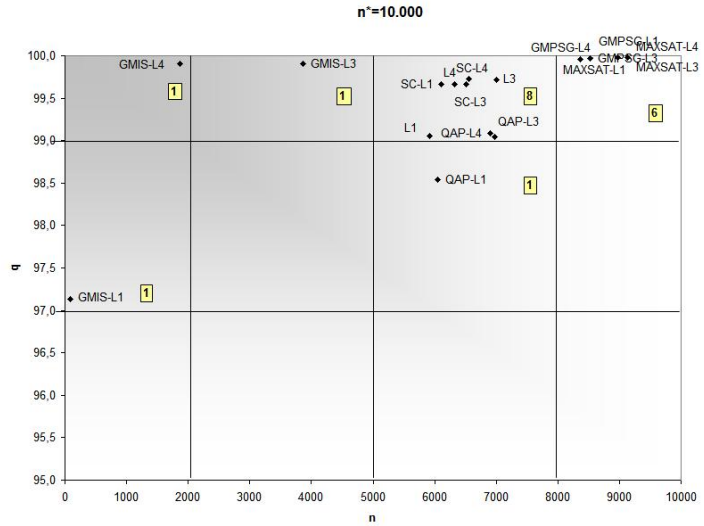

Figura 6: Comparação com $n_j^* \cong 10.000$.

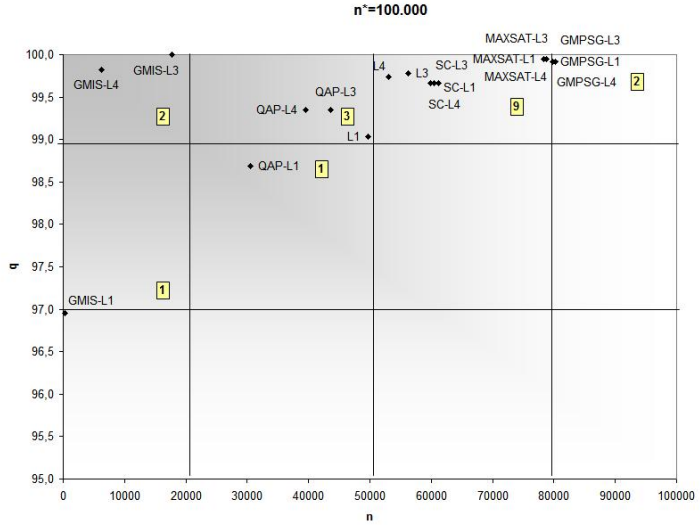

Figura 7: Comparação com $n_j^* \cong 100.000$.

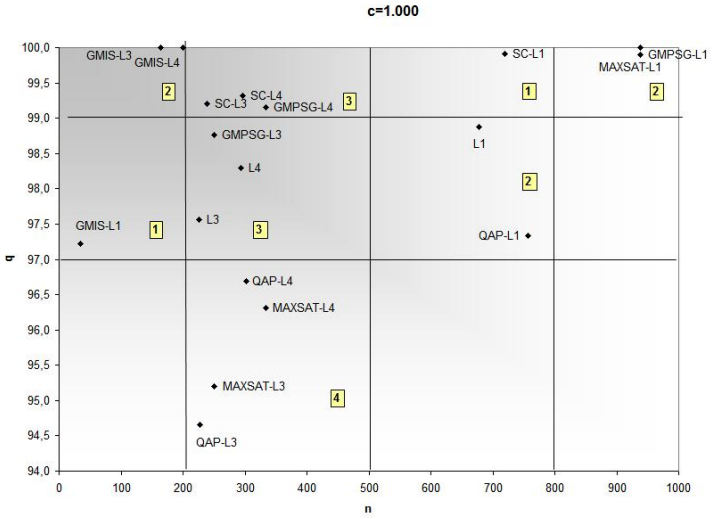

Figura 8: Comparação com $c = 1.000$.

heurística gqapd executou em tempo intermediário, conforme n_j^* cresce ela fica relativamente mais rápida: três pontos na região "esquerda" para $n_j^* = 100.000$, três pontos na região "direita" para $n_j^* = 10.000$ e três pontos nas regiões "direita" e "extremo direita" para $n_j^* = 1.000$. Em seguida vem a heurística sc, sete pontos na região "direita" e dois na "extremo direita".

Considerando-se a relação tempo de processamento e qualidade da solução, a heurística que apresentou o melhor comportamento é a identificada por gmis. Dos nove pontos, oito deles encontram-se na região "sombreada", sendo que três deles encontram-se na melhor região a "superior extremo esquerda". De fato, pode-se observar na Tabela 2 que, para o problema do conjunto independente máximo, a média da qualidade das soluções encontra-se acima de 97 e a média do número máximo de iterações em apenas um caso em nove (L_3 com $n_j^* = 1.000$) ultrapassou 50% do limite superior de iterações n_j^*. Em segundo lugar, encontra-se a heurística para o problema quadrático de atribuição (gqapd): quatro casos na região "superior" e cinco casos na região "média", sendo três deles na região "sombreada". Todos os noves pontos de cada um dos problemas de satisfabilidade máxima ponderada (maxsat), de recobrimento (sc) e de planarização de grafos (gmpsg), encontram-se na área "superior", o que indica que as soluções são de qualidade, entretanto, o tempo de processamento é alto.

Ainda na Tabela 2, pode-se analisar o comportamento considerando cada um dos problemas individualmente. Para o problema de conjunto independente máximo (gmis), o critério L_1 é o mais rápido e o que produz soluções de pior qualidade. O critério L_4 é o melhor, produz soluções de qualidade equivalentes aos do critério L_3 e em menos tempo. Tanto o critério L_3 quanto o L_4 produzem soluções de qualidade equivalentes as produzidas pelo GRASP original com $n = n_j^*$ e com um número de iterações bem menor. Para este problema o uso dos critérios de parada mostrou-se eficiente. Os problemas de satisfabilidade máxima ponderada (maxsat) e de planarização de grafos (gmpsg) têm comportamento similar, em termos de qualidade de solução e de tempo de processamento: os três critérios são equivalentes. Em ambos os casos o programa mostrou potencial para melhorar a melhor solução encontrada caso executasse mais iterações (em muitos casos valor de n igual ao limite superior n_j^*). Por este motivo, o uso de critérios de parada também é válido pois ele reconhece que a busca não deve parar. O problema de recobrimento (sc) também é beneficiado pelo uso de critérios de parada. A qualidade média dos três critérios são equivalentes e altas ($99,8\%$) e o número médio de iterações é em torno de 60% do limite superior. De forma geral, para o problema quadrático de atribuição (gqapd) novamente os três critérios são equivalentes em termos de qualidade de solução. A ordem de eficiência em relação ao tempo é L_1, L_4 e L_3, com pequenas diferenças entre elas.

Observando-se o posicionamento da média geral dos critérios de parada L_1, L_3 e L_4 nos gráficos da Figuras 5 a 7, percebe-se que de maneira geral o critério de parada L_1 é mais rápido e produz soluções de qualidade inferior às demais (embora ainda seja muito boa). Enquanto que os critérios de parada L_3 e L_4 produzem soluções de qualidade similares, porém com uma pequena vantagem de tempo para o L_4 (um pouco mais rápido que L_3). Esta ordem de parada está ligada diretamente a região de continuidade, que é controlada pelo parâmetro c. Neste experimento, para manter-se o limite superior de iteração igual entre os critérios de parada, o parâmetro c teve que variar de critério para critério, conforme Tabela 1. Com isto, o critério de parada L_1, com o parâmetro c de menor valor, pára mais rápido; o critério de parada L_3, com o parâmetro c de maior valor, demora mais; e o critério de parada L_4, com o parâmetro c de valor intermediário, pára em tempo intermediário.

No primeiro experimento os resultados obtidos para os diferentes critérios foram comparados considerando-se um mesmo valor de n_j^*. No próximo experimento, os resultados serão comparados considerando-se o mesmo valor do parâmetro c para todos os critérios de parada. Trabalhou-se com $c = 1.000$ conforme o artigo de Boender e Rinnooy Kan (1987). Sabe-se que o limite superior de iterações dos critérios de parada L_1, L_3 e L_4, respectivamente n^* igual a 938, 250 e 333, são inferiores ao número fixo de iterações determinado igual a 1.000 e portanto a melhor solução conhecida no cálculo da qualidade será a solução obtida pela execução fixa de 1.000 iterações.

O gmis é o programa em que os critérios de parada obtiveram melhor desempenho, conforme o gráfico da Figura 8, onde aparece com dois de três critérios na região "superior extremo esquerda". Para a heurística gmis o critério de parada L_1 é o mais rápido (3,5% do número de iterações de um GRASP com o número de iterações fixas), produzindo soluções de com 97% de qualidade. Enquanto que o critério L_3 encontra as melhores soluções (100% de qualidade), assim como o L_4, porém com um tempo médio menor (16% do número de iterações fixas). Como avaliado no primeiro experimento as heurísticas maxsat, gmpsg e sc têm comportamento similar. Todas páram com $n = w = n_j^*$. Com isto, o tempo de processamento e a qualidade são definidos diretamente pelo limite superior de iteração n_j^*: quanto maior o limite, maior o tempo e a qualidade. Ainda no gráfico, observa-se que para a heurística gqapd, o critério L_3 é mais rápido, podendo executar cerca de 23% do tempo do GRASP original e obter uma solução em média aproximadamente 95% boa. Com os critérios L_1 e L_4 é possível obter soluções em média 97% boas utilizando-se 76% e 30% do tempo do GRASP original respectivamente. De forma geral, o critério de parada L_1 demora mais porém produz soluções de melhor qualidade. Caso reduzir tempo seja imprescindível, L_3 e L_4 produzem soluções cerca de 97% boas com 23% e 29% do tempo do GRASP original respectivamente.

4 Conclusões

Foi proposto um *framework* GRASP_RPB para incorporar critérios de parada baseados em estatística bayesiana em heurísticas baseadas na metaheurística GRASP.

Foram realizados estudos comparativos entre os programas sob o *framework* GRASP_RPB e os programas originais com parada determinada por um número fixo de iterações. Resultados computacionais mostram que, para um conjunto de 6.080, experimentos o uso de critérios de parada bayesianos é uma boa estratégia.

Como trabalho futuro, pretende-se estudar a extensão deste critério, proposta por Boender e Rinnooy Kan (1991), que também considera o valor da função objetivo dos ótimos locais como informação de entrada na definição do critério de parada, em heurísticas baseadas na metaheurística GRASP. Outro trabalho pode ser o estudo de critérios de parada bayesianos aplicados a outros métodos com múltiplas inicializações, como algoritmos genéticos.

Agradecimentos

Adriana C.F. Alvim agradece ao CNPq e à FAPERJ pelo apoio parcial deste trabalho. Os autores agradecem também os comentários e sugestões de avaliadores anônimos que contribuíram para melhoria da versão final do texto.

5 Referências

Aiex, R.M., Resende, M.G.C. e Ribeiro, C.C. (2002), Probability distribution of solution time in GRASP: An experimental investigation, *Journal of Heuristics*, 8, 343-373.

Bartkuté, V., Felinskas, G. e Sakalauskas, L. (2006), Optimality Testing in Stochastic and Heuristic Algorithms, *Technological And Economic Development Of Economy*, 12, 4-10.

Bartkuté, V. e Sakalauskas, L. (2007), Simultaneous perturbation stochastic approximation of nonsmooth functions, *European J. of Operational Research*, 181, 1174-1188.

Bartkuté, V. e Sakalauskas, L. (2009), Statistical inferences for termination of Markov type random search algorithms, *Journal of Optimization Theory and Applications*, Publicação online, http://dx.doi.org/10.1007/s10957-008-9502-3.

Boender, C. e Rinnooy Kan, A. (1987), Bayesian stopping rules for multistart global optimization methods, *Mathematical Programming*, 37, 59-80.

Boender, C. e Rinnooy Kan, A. (1991), On When to Stop Sampling for the Maximum, *Journal of Global Optimization*, 1, 331-340.
Burkard, R., Karisch, S. e Rendl, F. (1991), QAPLIB: A quadratic assignment problem library, *European Journal of Operational Research*, 55, 115-119.
Feo, T.A. e Resende, M.G.C. (1989), A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters*, 8, 67-71.
Feo, T.A. e Resende, M.G.C. (1995), Greedy Randomized Adaptive Search Procedures, *Journal of Global Optimization*, 6, 109-133.
Feo, T.A., Resende, M.G.C. e Smith, S.H. (1994), A greedy randomized adaptive search procedure for maximum independent set, *Operations Research*, 42, 860-878.
Grigaitis, D., Bartkuté, V. e Sakalauskas, L. (2007), An optimization of system for automatic recognition of ischemic stroke areas in computed tomography images, *Informatica*, 18, 603-614.
Hart, W. (1998), Sequential stopping rules for random optimization methods with applications to multistart local search, *SIAM Journal of Optimization*, 9, 270-290.
Li, Y., Pardalos, P.M. e Resende, M.G.C., A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem, em Pardalos, P.M. e Wolkowicz, H. (Eds.), *Quadratic assignment and related problems*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 16, 237-261, 1994.
Orsenigo, C. e Vercellis, C. (2006), Bayesian stopping rules for greedy randomized procedure, *Journal of Global Optimization*, 36, 365-377.
Resende, M.G.C., Feo, T.A. e Smith, S.H. (1998), Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP, *ACM Transactions on Mathematical Software*, 24, 386-394.
Resende, M.G.C., Pardalos, P.M. e Li, Y. (1996), Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP, *ACM Transactions on Mathematical Software*, 22, 104-118.
Resende, M.G.C., Pitsoulis, L.S. e Pardalos, P.M., Approximate solution of weighted MAX-SAT problems using GRASP, em Gu, J. e Pardalos, P.M. (Eds.), *Satisfiability problems*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 35, 393-405, 1997.
Resende, M.G.C., Pitsoulis, L.S. e Pardalos, P.M. (1999), Fortran subroutines for computing approximate solutions of weighted MAX-SAT problems using GRASP, *Discrete Applied Mathematics*, 100, 95-113.
Resende, M.G.C. e Ribeiro, C.C. (1997), A GRASP for graph planarization, *Networks*, 29, 173-189.
Resende, M.G.C. e Ribeiro, C.C., Greedy Randomized Adaptive Search Procedures, em Kochenberger, G. e Glover, F. (Eds.), *Handbook of Metaheuristics*, Kluwer, Boston, 219-249, 2003.
Resende, M.G.C. e Ribeiro, C.C., Greedy Randomized Adaptive Search Procedures - Advances and Applications, em Potvin, J.Y. e Gendreau, M. (Eds.), *Handbook of Metaheuristics*, Springer, Boston, 2009.
Ribeiro, C.C. e Resende, M.G.C. (1999), Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP, *ACM Transactions on Mathematical*, 25, 341-352.
Schrage, L. (1979), A More Portable Fortran Random Number Generator, *ACM Transactions on Mathematical Software*, 5, 132-138.