versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

PROPOSTA E AVALIAÇÃO DE HEURÍSTICAS GRASP PARA O PROBLEMA DA DIVERSIDADE MÁXIMA

Geiza Cristina da Silva
Luiz Satoru Ochi *
Simone Lima Martins
Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói – RJ
gsilva@ic.uff.br
satoru@ic.uff.br
simone@ic.uff.br

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 02/2005; aceito em 11/2005
Received February 2005; accepted November 2005

Resumo

O Problema da Diversidade Máxima (PDM) consiste em, dado um conjunto N composto de n elementos, selecionar um subconjunto $M \subset N$ de forma tal que os elementos de M possuam a maior diversidade possível entre eles. O PDM pertence à classe de problemas NP-Difícil limitando, com isso, o uso exclusivo de métodos exatos e tornando atrativo o desenvolvimento de novos métodos heurísticos na solução aproximada deste problema. Neste trabalho são propostos métodos heurísticos de construção e busca local que, combinados, são usados como base em diferentes versões do algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*). Incluímos como objetivos analisar o impacto destas heurísticas no desempenho da metaheurística GRASP. Resultados computacionais mostram que os algoritmos propostos sempre alcançam uma solução ótima quando esta é conhecida e, para instâncias maiores, apresentam um desempenho médio superior quando comparados com as melhores heurísticas GRASP da literatura.

Palavras-chave: GRASP; metaheurística; problema da diversidade máxima.

Abstract

The Maximum Diversity Problem (MDP) consists of, given a set N with n elements, selecting a subset $M \subset N$ such that the elements of M have the most possible diversity among them. The MDP belongs to the class of NP-Hard problems limiting the exclusive use of exact methods and turning attractive the development of heuristics to solve the problem. In this work we propose constructive and local search heuristics which are used in different versions of GRASP (*Greedy Randomized Adaptive Search Procedure*). We also analyze the impact of this heuristics in the GRASP performance. Computational results show that the proposed algorithms always find an optimal solution when this one is known and, for larger instances, produce an average performance better than well known versions of GRASP from the literature.

Keywords: GRASP; metaheuristic; maximum diversity problem.

1. Introdução

O Problema da Diversidade Máxima (PDM) consiste em, dado um conjunto N com n elementos, selecionar um subconjunto $M \subset N$ de forma tal que os elementos de M possuam a maior diversidade possível entre eles.

Este problema é classificado como NP-Difícil [KGD93] limitando o uso exclusivo de métodos exatos para a sua solução. Neste trabalho são propostos diferentes algoritmos heurísticos de construção e busca local, que aplicados aos conceitos da metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*), se mostram bastante promissores na resolução de instâncias do PDM. Nos testes computacionais realizados, é mostrado que, para instâncias de pequeno porte, os algoritmos propostos sempre alcançam uma solução ótima sempre que estes são conhecidos e, para instâncias maiores, apresentam um desempenho médio superior quando comparados com as melhores heurísticas da literatura.

O restante deste artigo está organizado conforme segue. Na seção 2 o Problema da Diversidade Máxima é apresentado. A seção 3 apresenta aplicações do problema e trabalhos relacionados. Na seção 4 são apresentadas características da metaheurística GRASP. Na seção 5, os algoritmos GRASP desenvolvidos em [Gh96] e [An03] são descritos e na seção 6 são discutidas as propostas de algoritmos deste trabalho. Na seção 7 os resultados e análises computacionais são apresentados. Finalmente, a seção 8 mostra as conclusões e sugestões de trabalhos futuros.

2. O Problema da Diversidade Máxima

O Problema da Diversidade Máxima (PDM) consiste em, a partir de um conjunto N contendo n elementos, selecionar um subconjunto $M \subset N$ com m elementos que possuam a maior diversidade entre si. Uma medida de diversidade entre dois elementos pode ser representada pela distância entre eles. A distância, neste caso, é calculada com base no conjunto de atributos (componentes) dos elementos de N.

Desta forma, para definir o problema, considere o conjunto de índices $N = \{1, 2, ..., n\}$, onde n é o número de elementos em N, e:

- t, o número de atributos que caracterizam os elementos de N;
- a_{ik}, o valor do atributo k, $k = \{1,..., t\}$ do elemento i, $i \in N$.
- $d(i,j)$, o índice de diversidade entre dois elementos distintos i e j com os respectivos valores de atributos ($a_{i1}, a_{i2}, ..., a_{it}$) e ($a_{j1}, a_{j2}, ..., a_{jt}$).

O índice de diversidade representa o grau de diferença existente entre os atributos dos elementos i e j, ou seja, a distância entre eles, que deve ser calculada usando-se uma dentre as métricas de distância existentes. De acordo com a área de aplicação, uma métrica pode refletir melhor o problema que outra [KGD93].

Seja D $[n \times n]$ uma matriz simétrica que armazena os índices de diversidade entre cada um dos elementos de N, preenchida da seguinte maneira: $d(i, j) = d(j, i)$ e $d(i, i) = 0$, $\forall (i, j) \in N$. O valor da diversidade de M é determinado pela fórmula $div(M) = \sum_{i,j \in M, i<j} d(i, j)$.

3. Trabalhos Relacionados

O Problema da Diversidade Máxima pode ser aplicado nas mais diversas áreas relacionadas ao campo da pesquisa operacional como na administração de recursos humanos [DGK94], [WL98], biologia [Ag97] e preservação ecológica [Ma99], [Un85].

Em [KGD93] foram apresentadas duas definições distintas para o PDM. Neste trabalho aborda-se a definição MAXSUM cujo objetivo é maximizar a soma das diversidades de um conjunto de elementos selecionados a partir de uma coleção maior. Os autores apresentaram ainda as duas formulações matemáticas para o PDM mostradas a seguir e provaram que o problema MAXSUM é NP-Difícil.

O PDM pode ser representado como um Problema de Programação Inteira como proposto em [KGD93]. Seja x_i, $\forall_i \in N$, variável binária igual a 1, se e somente se o elemento i estiver na solução e 0, caso contrário, e $d(i, j)$ representando a diversidade entre i e j, $i, j \in N$.

(P1) *Maximizar*

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(i,j) x_i x_j, \tag{1}$$

sujeito a:

$$\sum_{i=1}^{n} x_i = m, \tag{2}$$

$$x_i \in \{0,1\}, \forall i = 1,...,n\ . \tag{3}$$

A formulação P1, descrita pelas equações 1-3, é um modelo não linear devido a sua função objetivo quadrática (1) e tem como meta maximizar a soma das diversidades de m elementos de N. A igualdade (2) exige que exatamente m elementos façam parte de uma solução viável e (3) indica que as variáveis do problema devem ser binárias.

Uma versão linearizada de P1 é também proposta em [KGD93], descrevendo o PDM como um Problema de Programação Linear Inteira. Para isto, além das notações da formulação anterior considere y_{ij}, variáveis de decisão que se tornam iguais a 1 se i e j fizerem parte da solução e 0, caso contrário, $\forall_{ij} \in N$, $i < j$ e $Q = \{(i,j): i, j \in \boldsymbol{N}, i < j\}$.

(P2) *Maximizar*

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(i,j) y_{ij}, \tag{4}$$

sujeito a:

$$\sum_{i=1}^{n} x_i = m, \tag{5}$$

$$x_i + x_j - y_{ij} \leq 1, \forall (i,j) \in Q \tag{6}$$

$$-x_i + y_{ij} \leq 0, \forall (i,j) \in Q \tag{7}$$

$$-x_j + y_{ij} \leq 0, \forall (i,j) \in Q \tag{8}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in Q \tag{9}$$

$$x_i \in \{0,1\}, \forall i \in N. \tag{10}$$

Na formulação (P2), a função objetivo (4) deve maximizar a soma das diversidades de m elementos de N. A igualdade (5) exige que exatamente m elementos pertençam a uma solução viável. Nas restrições (6) pode-se observar que a variável binária y_{ij} assume valor 1 sempre que x_i e x_j receberem tal valor e y_{ij} assume valor 0 se x_i e x_j forem ambas iguais a 0 ou se x_i ou x_j (somente uma delas) for igual a 1. As restrições (7) exigem que no caso em que x_i pertencer à solução então deverá existir algum j tal que $y_{ij} = 1$. Da mesma forma que em (7), as restrições (8) exigem que se x_j pertencer à solução então deverá existir algum i tal que $y_{ij} = 1$. As restrições (9) e (10) representam as variáveis binárias do problema, sendo $y_{ij} = 1$ se x_i e x_j estiverem na solução e 0 caso contrário e cada x_i assumindo valor 1 se pertencer à solução e 0, caso contrário.

Outras formulações matemáticas para o PDM foram propostas em [GKD95] para aplicações específicas na área de meio ambiente.

Em [Gh96] foi proposto um algoritmo GRASP para o PDM e para comprovar a qualidade da heurística proposta foi implementado um algoritmo exato. Os resultados computacionais mostraram que o GRASP proposto alcança uma solução ótima ou uma solução muito próxima da ótima, mostrando desta forma sua eficiência ao menos para instâncias de pequeno porte ($n \leq 40$).

[Ag97] explorou o problema de selecionar subconjuntos de componentes químicos. No trabalho introduziu duas medidas para quantificar a diversidade e apresentou uma solução para o problema baseada em *simulated annealing*. O método foi testado usando bases de dados formadas a partir de métodos estatísticos. Os resultados foram visualizados a partir do algoritmo de mapeamento não-linear de Sammon [Sa69]. O artigo não fornece detalhes dos testes computacionais mas cita que o método encontra resultados satisfatórios.

Em [CK97] foram apresentadas duas heurísticas para o PDM, uma relaxação lagrangeana e uma heurística gulosa. Para avaliar os resultados obtidos foi implementado um algoritmo *branch and bound* e, desta forma, os autores concluíram que a heurística gulosa apresentou bons resultados para um conjunto de testes com número pequeno de elementos ($n \leq 40$).

Em 1998, [WL98] apresentou cinco heurísticas para o PDM. Este trabalho foi realizado para resolver uma aplicação prática que é criar grupos de projetos em uma classe de estudantes, de forma tal que esses estudantes sejam inseridos em ambientes heterogêneos, baseado em critérios como, por exemplo, nacionalidade, idade e formação. As heurísticas foram testadas somente para uma instância composta de dados reais e foi implementado também um algoritmo exato. Os testes apontaram a heurística LCW (Método *Lotfi-Cerveny-Weitz*) [WL98] como sendo a melhor para a solução do problema.

Em [GKD98] foram propostas quatro heurísticas sendo duas delas construtivas e as outras destrutivas onde, a partir de uma solução heurística com e elementos ocorre uma eliminação progressiva até que se tenha uma solução viável com m elementos. A eficiência dessas heurísticas foi testada através de comparação com os resultados de um método exato e para isso foram criadas diversas instâncias de populações com até 30 elementos. Foi citado pelos autores que as heurísticas propostas obtiveram em média resultados próximos aos do algoritmo exato (a no máximo 2% do ótimo), com um tempo de computação muito menor.

[HRT97] trata de um problema que considera um grafo simétrico $G=(V,E)$ completo, onde a cada aresta é associado um peso e $|V| = n$. Sendo $p = \{2, ...,n\}$ e $k = \{1, ..., n/p\}$ devem-se obter k conjuntos disjuntos, com p vértices cada um de tal forma que a soma dos pesos das

arestas de cada subconjunto seja máxima. Este problema possui diversas variações e uma delas é o PDM, no caso particular em que $k = 1$.

Em [BMD00] foi introduzido um novo modelo para tratar o problema de diversidade de grupos de trabalho que permitiu reduzi-lo ao problema de fluxo em redes e desenvolver um algoritmo exato para resolvê-lo. O modelo foi aplicado a uma base de dados real obtida em um curso de pós-graduação.

Em [KG99] foi desenvolvida uma heurística busca tabu para encontrar a máxima diversidade. Segundo os autores, a heurística foi testada gerando aleatoriamente instâncias de problemas com populações de tamanho que variam de n = 100 a n = 1000 elementos onde se desejava obter subconjuntos de m elementos mais diversos com tamanho entre 10% a 30% de n. Os autores citam que testes computacionais foram realizados, mas os resultados não foram comparados com qualquer outro método da literatura.

Além de [Gh96], outro trabalho que propõe a metaheurística GRASP para o PDM é [An03]. Nesta proposta, desenvolveu-se para a fase de construção uma nova heurística construtiva e incorporou-se a busca local encontrada em [Gh96]. O autor gerou uma extensa bateria de testes onde o tamanho da população variou de 10 a 250 elementos. Para avaliação dos resultados produzidos por seu algoritmo, foi implementado um algoritmo exaustivo para as instâncias pequenas (até 50 elementos) para produzir soluções exatas e implementou-se também o algoritmo de Ghosh para a comparação dos resultados.

De nosso conhecimento, os algoritmos GRASP apresentados em [Gh96] e [An03] são os que detêm os melhores resultados aproximados da literatura para o PDM e por isto, estes foram usados para comparações com os algoritmos aqui propostos. Os algoritmos de [Gh96] e [An03] serão detalhadamente descritos na seção 5.

4. A Metaheurística GRASP

Proposta por Feo e Resende, a metaheurística GRASP [FR95, Re01, RR02] é um algoritmo iterativo onde cada iteração é composta por duas fases: uma fase de construção onde uma solução viável para o problema é criada e uma fase de busca local, posterior à construção, que tem como objetivo tentar melhorar a solução inicial obtida na fase de construção. As iterações GRASP são independentes, ou seja, na iteração atual não se leva em conta nenhuma informação das iterações anteriores. O critério de parada normalmente usado é um número máximo de iterações. A melhor solução obtida ao final da execução do GRASP é a solução final. A Figura 1 mostra um pseudocódigo para o GRASP.

```
proc grasp (MAX_ITER)
    Carrega_Instancia_Entrada()
    para k=1 ate MAX_ITER faca
        Solucao = constroiSolucao ()
        Solucao = buscaLocal ()
        atualizaSolucao (Solucao, Melhor_Solucao)
    fim para
    retorna (Melhor_Solucao)
fim grasp
```

Figura 1 – Pseudocódigo do procedimento GRASP.

A fase de construção do GRASP é também iterativa, onde uma solução é construída elemento a elemento. Cada inserção de um novo elemento é feita através da escolha aleatória em uma lista restrita de candidatos (*LRC*). Considere para a construção da *LRC* que o problema aqui tratado é de maximização de uma função *f*. A *LRC* é composta de elementos candidatos à solução avaliados de acordo com o benefício associado a sua inclusão na solução parcial através de uma função gulosa *g*. Os elementos participantes da *LRC* podem ser escolhidos através de duas formas: pelo número de elementos ou pela qualidade dos elementos [RR02].

Na primeira forma, os elementos candidatos são ordenados em ordem decrescente de benefício segundo a função *g* e os *p* primeiros elementos são incluídos na *LRC*. O valor de *p* é definido como $p = 1 + \alpha\ (a - 1)$, onde *a* é número total de candidatos (α é um dado de entrada que tem valores definidos no intervalo [0,1]). Note que, se $\alpha = 0$ um algoritmo totalmente guloso é executado, já que só é possível a escolha de um único elemento a ser inserido na solução. Por outro lado, se $\alpha = 1$, a lista conterá todos os possíveis candidatos, o que resultará em um algoritmo aleatório.

Na segunda forma, considerando um problema de maximização, $e_{min} = \min\{g(e)\}$ o menor incremento a solução parcial de acordo com *g* e $e_{max} = \max\{g(e)\}$, o maior incremento, a LRC será constituída dos elementos candidatos à solução cujo valor retornado pela função *g* esteja no intervalo $[(1 - \alpha)(e_{min} - e_{max}) + e_{max}\ ,\ e_{max}]$.

Como já citado, α é um valor definido no intervalo [0,1]. Neste caso, se $\alpha = 1$, estará sendo executado um algoritmo absolutamente guloso, já que somente elementos de maior incremento e_{max} podem ser inseridos na solução e se $\alpha = 0$, a lista conterá todos os possíveis candidatos, o que resultará em um algoritmo aleatório.

Portanto é perceptível que a escolha do valor de α é um ponto importante no desempenho da metaheurística GRASP. A influência desta escolha na qualidade da solução construída é estudada em [RR02]. Na Figura 2 é mostrado o pseudocódigo da rotina de construção de uma solução.

```
proc constroiSolucao ()
    Solucao = {}
    enquanto Solucao não está completa faca
        Construa LRC
        Selecione aleatoriamente um elemento s da LRC
        Solucao = Solucao ∪ {s}
        Adapte a funcao gulosa
    fim enquanto
    retorna (Solucao)
fim constroiSolucao
```

Figura 2 – Pseudocódigo da fase de construção.

A busca local é uma tentativa de melhoria da solução obtida na fase de construção (`Solucao`). Nesta fase, percorre-se a vizinhança da solução corrente buscando uma solução de melhor qualidade. Por exemplo, em um problema onde o objetivo é a maximização de uma função *f*, entende-se que uma solução *t* é melhor que `Solucao` se $f(t) > f(\texttt{Solucao})$.

Se não existir uma melhor solução na vizinhança, a solução corrente é considerada um ótimo local. A busca termina quando um ótimo local for atingido. O pseudocódigo da busca local é apresentado na Figura 3.

Diversos conceitos, alguns já utilizados em outras metaheurísticas, têm sido incorporados ao GRASP como alternativa para tentar melhorar sua eficiência. Dentre estes podem ser citados reconexão por caminhos [LM99], memória de longo prazo [FG99], [GL98], GRASP Paralelo [PPMR95], [MRRP00] GRASP reativo [PR97] entre outros. A idéia de GRASP reativo é utilizada neste trabalho e será discutida a seguir.

```
proc buscaLocal (Solucao)
     enquanto Solucao s não é um ótimo local faca
          Encontre uma solução t em V(s)
          se custo(t) > custo(Solucao) entao
               Solucao = t
          fim se
     fim enquanto
     retorna (Solucao)
fim buscaLocal
```

Figura 3 – Pseudocódigo da fase de busca local.

Como já citado anteriormente, a composição da lista restrita de candidatos (*LRC*) é determinada a partir do dado de entrada α. Este dado pode ser responsável pela qualidade das soluções geradas na fase de construção.

Prais e Ribeiro propuseram em [PR97] um método chamado GRASP reativo cuja característica principal é o auto ajuste do valor de α de acordo com a qualidade das soluções previamente encontradas. Neste procedimento, um valor de α é selecionado aleatoriamente em um conjunto $A = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ contendo m possíveis valores de α. Desta forma, já que se tem vários valores de α ao invés de um único, é possível construir listas restritas de candidatos de diferentes cardinalidades. A cada elemento $\alpha_i \in A$ é associada uma probabilidade p_i, $i = 1, ..., m$. Inicialmente, tem-se $p_i = 1/m$, o que corresponde a uma distribuição uniforme.

Periodicamente, ou seja, a cada bloco de iterações, a distribuição de probabilidade é atualizada, utilizando as informações coletadas nas iterações realizadas anteriormente. Na aquisição dessas informações, várias estratégias podem ser usadas. Em uma delas, o valor médio das soluções obtidas com cada valor de α é utilizado. Desta forma, no próximo bloco de iterações GRASP, o valor de α com o qual se obteve melhores soluções receberá uma probabilidade maior e, conseqüentemente, possuirá mais chance de ser escolhido e poderá ser mais usado neste bloco de iterações.

5. Descrição dos Algoritmos GRASP da Literatura

Nesta seção os algoritmos GRASP propostos em [Gh96] e [An03] para o PDM são descritos. Inicialmente em ambos os trabalhos, os autores consideram que para construir uma solução completa do PDM, tem-se m soluções parciais, assim, cada solução parcial é aqui denotada como M_{c-1} , ou seja, uma solução parcial que contém $c-1$ elementos ($1 \leq c \leq m$).

5.1 Algoritmo GRASP de [Gh96]

Na fase de construção, para se gerar uma solução M_c a partir de $M_{c\text{-}1}$ são calculados dois valores associados a cada elemento $i \in N - M_{c\text{-}1}$ candidato a ingressar na solução parcial. O primeiro valor corresponde à soma dos índices de diversidade do elemento candidato i aos elementos já pertencentes à solução parcial acrescida da soma dos índices de diversidade de i aos $m - c$ elementos que apresentam as *menores* distâncias de i. Este valor é considerado um limite *inferior* de distâncias e é denominado $\Delta z_L(i)$. O outro valor a ser calculado é, equivalentemente, um limite superior de distâncias $\Delta z_U(i)$, e corresponde à soma dos índices de diversidade de i aos elementos já pertencentes à solução parcial acrescida da soma dos índices de diversidade de i aos $m - c$ elementos que apresentam as *maiores* distâncias de i. Os cálculos de $\Delta z_L(i)$ e $\Delta z_U(i)$ são mostrados nas equações 11 e 12.

$$\Delta z_L(i) = \sum_{j \in M_{c-1}} d(i,j) + \sum_{n-m+1 \le r \le n-c} d_i^r(Q_{ic}) \tag{11}$$

$$\Delta z_U(i) = \sum_{j \in M_{c-1}} d(i,j) + \sum_{1 \le r \le n-c} d_i^r(Q_{ic}) \tag{12}$$

O fator aleatório da heurística está na escolha randômica de um número u no intervalo (0,1). A partir de sua determinação, calcula-se $\Delta z'(i) = \Delta z_L(i) + u\Delta z_u(i)$. O elemento i que retorna o maior valor de $\Delta z'$ é incluído na solução parcial. Este procedimento é repetido até que seja obtida uma solução completa com m elementos.

Em [Gh96] é proposta uma busca local exaustiva para cada iteração GRASP onde, a partir de uma solução completa, a cada passo verifica-se a melhoria associada à troca de cada elemento $i \in M$ por cada elemento $j \in N - M$. Essa melhoria é rotulada como $\Delta z(i,j)$ e é calculada da seguinte forma:

$$\Delta z(i,j) = \sum_{u \in M - \{i\}} d(j,u) + d(i,u) \tag{13}$$

5.2 Algoritmo GRASP de [An03]

O GRASP proposto neste artigo, inicialmente, para cada $i \in N$ calcula dois valores que serão usados na fase de construção. O primeiro deles é chamado $SD(i)$ e corresponde à soma dos índices de diversidade de i a todos os demais elementos $j \in N - \{i\}$. O segundo valor, denominado $MD(i)$ corresponde à média dos índices de diversidade, calculada fazendo-se a soma da diversidade de i a todos os demais elementos $j \in N - \{i\}$, e dividindo-se esta soma pelo número de elementos em N. Os cálculos de $SD(i)$ e $MD(i)$ são detalhados nas equações 14 e 15.

$$SD(i) = \sum_{j \in N - \{i\}} d(i.j) \tag{14}$$

$$MD(i) = \frac{\sum_{j \in N - \{i\}} d(i.j)}{n} \tag{15}$$

Na fase de construção, a escolha dos elementos de M é feita como se segue. O primeiro elemento da solução é selecionado aleatoriamente dentre os m elementos de maior valor em SD. A partir daí, para construir cada solução parcial, $M_{c\text{-}1}$ com $2 \leq c \leq m$, é calculado, para cada $i \in N - M_{c\text{-}1}$ candidato à solução, o valor de $SDS(i)$ que corresponde à média dos índices de diversidade entre i e os elementos $j \in M_{c\text{-}1}$, que já foram incluídos na solução parcial como pode ser visto na equação 16.

$$SDS(i) = \frac{\sum_{j \in N - \{i\}} d(i,j)}{c-1} \tag{16}$$

Em seguida, é calculado para cada candidato i o valor de $\Delta z(i)$, baseado em $SDS(i)$ segundo o seguinte critério: se a solução está a poucos elementos de ser completada então elementos com $SDS(i)$ alto serão incluídos, caso contrário, deve ser avaliada a contribuição que a inserção do elemento trará também com base em $MD(i)$. O cálculo de $\Delta z(i)$ é detalhado na equação 17.

$$\Delta z(i) = \begin{cases} SDS(i) & \text{, se } SDS(i) > MD(i) \text{ e } c/2 \\ \dfrac{SDS(i) + MD(i)}{2} & \text{, caso contrário} \end{cases} \tag{17}$$

Após todos os $\Delta z(i)$ terem sido calculados, eles são ordenados de forma crescente e com os m elementos de maiores $\Delta z(i)$ é calculado D que resulta da diferença entre o maior e menor $\Delta z(i)$ dividido por $m - 1$. Com os Q maiores elementos cuja diferença entre seu Δz e o Δz do próximo elemento seja menor que o valor de D é criada uma LRC de onde um elemento é selecionado aleatoriamente para fazer parte da solução. O cálculo de D pode ser visto na equação 18.

$$D = \frac{\Delta z(i)_{\max} - \Delta z(i)_{\min}}{m-1} \tag{18}$$

O procedimento de construção se encerra quando uma solução completa é formada. Em [An03], a busca local utilizada é a mesma de [Gh96].

6. Algoritmos Propostos

Nesta seção são apresentadas novas propostas de algoritmos de construção e busca local para a metaheurística GRASP aplicada a solução aproximada do PDM. Os métodos de construção apresentados incorporam estratégias que visam melhorar a qualidade e a diversidade das soluções encontradas nesta etapa.

Outra proposta refere-se ao uso de estratégias de atualização dinâmica do parâmetro α no GRASP obtendo uma versão denominada GRASP Reativo [PR97] que, conforme citado na seção 3, estende o conceito do GRASP original com iterações independentes a fim de levar informações de uma iteração para outra. A idéia é de, inicialmente, utilizar diferentes valores do parâmetro α para cada instância do PDM durante certo número de iterações e num segundo bloco de iterações privilegiar, com maior número de iterações, os valores de α para os quais foram encontradas as melhores soluções até o momento.

Outra estratégia implementada neste trabalho é o uso de um mecanismo de *filtro* de soluções na etapa de construção do GRASP. Este procedimento é motivado pelo fato que, na maioria das metaheurísticas, a etapa de busca local consome a maior parte do tempo de computação do algoritmo. Desta forma, o objetivo é inicializar uma busca local com soluções de melhor qualidade para, com isso, chegar a ótimos locais de forma mais econômica. Assim, a cada iteração GRASP, a fase de construção gera ao invés de uma, p soluções de modo que somente a melhor delas seja enviada para a busca local.

As heurísticas de construção aqui propostas utilizam conceitos clássicos de heurísticas de inserção adaptados ao método GRASP. O objetivo maior deste trabalho, como dito anteriormente, é analisar o impacto de cada método de construção e busca local propostos e os existentes na literatura no desempenho da metaheurística GRASP.

Heurística Aleatorizada das K Maiores Distâncias (HA-KMD)

Esta heurística constrói uma solução inicial de forma iterativa onde um elemento é selecionado para ser inserido na solução parcial de forma aleatória partindo de uma *LRC* que contém K melhores candidatos.

A *LRC* é criada da seguinte maneira: para cada $i \in N$ são selecionados os K elementos $j \in N - \{i\}$ que possuem os maiores índices de diversidade em relação a i, ou seja, os maiores $d(i, j)$. Esses valores são somados, totalizando s_i. Uma lista de índices i ordenada em ordem decrescente de valor de s_i é criada e, a partir dela, são selecionados os K primeiros elementos da lista para compor a *LRC*.

Definida a *LRC*, seleciona-se aleatoriamente um elemento $s \in LRC$.

Para implementar o GRASP reativo com o parâmetro K variável, este procedimento de construção é dividido em dois blocos de iterações. Seja t o número total de iterações do algoritmo GRASP.

O primeiro bloco *B1* corresponde às $0{,}4t$ primeiras iterações e nele *quatro* valores distintos de $K = K_i$, $i = 1, ..., 4$ são usados. Para isto, *B1* é subdividido em quatro blocos r_i contendo $0.1t$ iterações e em cada r_i é usada uma *LRC* de tamanho K_i. Os valores de K_i estão dispostos na Tabela 1 onde m é o número de elementos em uma solução viável e $\mu = (n\text{-}m)/2$.

Tabela 1 – Valores de K_i no bloco *B1*.

i	r_i	K_i
1	1 a $0{,}1t$	$m + \mu - 0.2\mu$
2	$0{,}1t$ a $0{,}2t$	$m + \mu - 0.1\mu$
3	$0{,}2t$ a $0{,}3t$	$m + \mu + 0.1\mu$
4	$0{,}3t$ a $0{,}4t$	$m + \mu + 0.2\mu$

Após a execução da última iteração de *B1*, a qualidade das soluções obtidas com cada K_i é avaliada. Isto é feito calculando-se o custo médio das soluções obtidas em cada r_i, $zm_i = [\sum_{q=1}^{0.1t} z(sol_{iq})]/0.1t$, onde $z(sol_{iq})$ corresponde ao custo da solução encontrada na q-ésima iteração de r_i, $q = 1, ..., 0.1t$ e $i = 1, ..., 4$. Em seguida, é criada uma lista *LK*

contendo os K_i ordenados de forma decrescente em relação a zm_i (lembrando que estamos tratando de um problema de maximização). Desta forma, a primeira posição de *LK* conterá o valor de K_i que obteve maior zm_i e assim por diante.

A partir daí, inicia-se a execução do segundo bloco de iterações (*B2)* composto pelas $0.6t$ iterações restantes, no qual se fará uso das informações coletadas em *B1*. Este bloco também deve ser dividido em quatro subintervalos v_i, mas, ao contrário do ocorre em *B1*, cada subintervalo conterá um número diferente de iterações conforme mostra a Tabela 2 e, além disso, em cada v_i deve ser usado o valor de *K* igual a LK_i. Deve-se ressaltar que a Tabela 2 mostra que os valores de K_i que produziram melhores soluções serão utilizados em um número maior de iterações.

Tabela 2 – Blocos de iterações em *B2*.

i	v_i
1	$0.4t$ a $0.64t$
2	$0.64t$ a $0.82t$
3	$0.82t$ a $0.94t$
4	$0.94t$ a t

Neste método aplicamos o filtro de soluções gerando, em cada iteração GRASP, 400 soluções. Somente a melhor delas é enviada à fase de busca local. O pseudocódigo do procedimento HA-KMD é apresentado na Figura 4.

```
proc construcaoHA_KMD (n, m, iter_corrente, t, custo_medio_sol, r)
    melhor_sol = {}
    custo_melhor_sol = 0
    K = determinaK (iter_corrente, t, LK)
    para cada i=1 ate num_sol_filtro faca
        LRC = constroiLRC (K)
        sol = {}
        para k=1 ate m faca
            escolha aleatoriamente um elemento s da LRC
            sol = sol ∪ {s}
            LRC = LRC - {s}
        fim para
        se z(sol) > custo_melhor_sol entao
            melhor_sol = sol
            custo_melhor_sol = z(sol)
        fim se
    fim para
    custo_medio_sol[r] = custo_medio_sol[r] + custo_melhor_sol
    se iter_cor == 0.4t entao
        LK = constroiLK (custo_medio_sol)
    fim se
    retorna (Solucao)
fim construcaoHA_KMD
```

Figura 4 – Pseudocódigo da construção HA-KMD.

Heurística Aleatorizada das K Maiores Distâncias (HA-KMD2)

Esta heurística é semelhante à **HA-KMD**. A diferença entre as duas está na maneira como a *LRC* é alterada a cada inserção de um novo elemento na solução parcial. Tendo em vista que na estratégia **HA-KMD** a única mudança na *LRC* após sua construção é a retirada do elemento à medida que este é inserido na solução, **HA-KMD2** foi projetada para ser mais adaptativa, fazendo com que a *LRC* seja inteiramente recalculada após a inclusão de cada elemento na solução parcial corrente.

Isto é feito da seguinte maneira. Considere que para obter uma solução completa, com m elementos, teremos m soluções parciais M_c, com c elementos, $1 \le c \le m$. Para construir M_1 o critério usado é o mesmo de **HA-KMD**, ou seja, um elemento é selecionado aleatoriamente a partir da *LRC* que corresponde aos K elementos de maiores somas de diversidade. Nas demais soluções parciais, para cada elemento $i \in N - M_c$ são selecionados os $K - c - 1$ elementos $j \in N - M_c - \{i\}$ que apresentam maiores $d(i, j)$.

Esses valores são somados, resultando em sf_i. Além disso, para cada $i \in N - M_c$ é feita a soma dos índices de diversidade entre i e todos os $j \in M_{c-1}$, resultando em sd_i. Uma lista inicial de candidatos *LIC* contendo todos os índices i em ordem decrescente de $s_i = sf_i + sd_i$, é criada.

A *LRC* é formada pelos K primeiros elementos de *LIC*. Um elemento é selecionado aleatoriamente da *LRC* para fazer parte da solução.

A estratégia de GRASP Reativo é implementada da mesma maneira como em **HA-KMD**, assim como o filtro de soluções, com alteração, porém, no número de soluções construídas a cada iteração GRASP que passa a ser 2 soluções, já que este procedimento exige mais esforço computacional que **HA-KMD**. Somente a solução de maior valor entre as soluções construídas passa pelo procedimento de busca local.

Heurística Aleatorizada da Inserção mais Distante (HA-IMD)

Este método incorpora o conceito clássico da heurística iterativa de inserção mais distante, sendo esta modificada de forma a torná-la adaptativa, gulosa e randômica.

Na construção de cada solução usando **HA-IMD**, tem-se m soluções parciais. Cada uma delas corresponde à solução parcial M_c que conterá c elementos.

O primeiro elemento da solução, m_1, é escolhido aleatoriamente entre todos os elementos pertencentes a N. O segundo elemento a ser selecionado deve ser o indivíduo $j \in N - M_1$ que oferece o maior índice de diversidade entre m_1 e j.

A partir do terceiro elemento, devemos calcular $Dsoma(j)$, $\forall j \in N - M_{c-1}$ apresentado na equação 19, onde o primeiro termo corresponde à soma das diversidades entre todos os elementos já pertencentes a solução parcial M_{c-1} e o segundo, à soma das diversidades entre o candidato j e todos os elementos da solução parcial.

$$Dsoma(j) = \sum_{1 \le y \le c-2} \; \sum_{y+1 \le w \le c-1} d(m_y, m_w) + \sum_{1 \le v \le c-1} d(m_v, j) \qquad (19)$$

A seguir a *LRC* é computada. A partir do cálculo de $Dsoma(j)$ uma lista inicial de candidatos é criada (*LIC*) contendo os índices dos j elementos candidatos ordenados de forma decrescente em relação a *Dsoma*. Os primeiros $\alpha \times n$ elementos são selecionados para compor a *LRC*.

Neste procedimento, o conceito de GRASP reativo é implementado da mesma forma que em **HA-KMD**. Seja t o número total de iterações do GRASP. O primeiro bloco de iterações $B1 = [1, ..., 0.4t]$ é dividido em quatro subintervalos de iterações r_i, $i = \{1, ..., 4\}$ contendo cada um deles um número igual de iterações. Utiliza-se o conjunto $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ onde para cada r_i um α_i é associado. Os valores de α são apresentados na Tabela 3. Considere t o número total de iterações do GRASP.

Tabela 3 – Valores de α no bloco $B1$.

i	r_i	α_i
1	1 a $0{,}1t$	0,03
2	$0{,}1t$ a $0{,}2t$	0,05
3	$0{,}2t$ a $0{,}3t$	0,07
4	$0{,}3t$ a $0{,}4t$	0,1

Após o término da última iteração de $B1$, deve-se avaliar quais os valores de α que ofereceram melhores médias de soluções. Isto é feito como em **HA-KMD**, ou seja, calculando-se o custo médio $zm_i = [\sum_{q=1}^{0.1t} z(sol_{iq})]/0.1t$, onde $z(sol_{iq})$ corresponde ao custo da solução encontrada na q-ésima iteração do intervalo r_i, $q = 1, ..., 0.1t$ e $i = 1, ..., 4$. Os valores de α_i são armazenados em uma lista $L\alpha$ ordenada por zm_i.

O próximo bloco de iterações $B2$ com as $0.6t$ restantes também será dividido em quatro subintervalos v_i, como apresentados na Tabela 2, e nele serão usadas as informações adquiridas em $B1$ e armazenadas em $L\alpha$. Desta forma, para cada v_i é usada uma LRC de tamanho $L\alpha_i$.

A partir da LRC é selecionado, aleatoriamente, um elemento para fazer parte da solução parcial. Este processo é repetido até que uma construção completa seja obtida. O filtro de soluções também é utilizado neste método. Para cada iteração do GRASP são criadas n soluções e somente a solução com maior custo prossegue para a fase de busca local.

O pseudocódigo da **HA-IMD** é apresentado na Figura 5.

```
proc construcaoHA_IMD (n, m, iter_corrente, t, custo_medio_sol, r)
    melhor_sol = {}
    custo_melhor_sol = 0
    α = determinaAlfa (iter_corrente, t, Lα)
    para i=1 ate num_sol_filtro faca
        Solucao = {}
        escolha aleatoriamente o elemento m1 de N
        Solucao = Solucao ∪ {m1}
        para cada j ∈ N − m1 faca
            calcule d(m1,j)
            m2 = j | d(j,m1) ≡ max(d(j,m1))
        fim para
        Solucao = Solucao ∪ {m2}
        para c=3 ate_m faca
            LRC = ConstroiLRC (α)
            escolha aleatoriamente um elemento s da LRC
            Solucao = Solucao ∪ {s}
        fim para
        custo_medio= constroiLRC (α)
    fim para
    custo_medio_sol[r] = custo_medio_sol[r] + custo_melhor_sol
    se iter_cor == 0.4t entao
        α = constroiAlfa (custo_medio_sol)
    fim se
    retorna (Solucao)
fim construcaoHA_IMD
```

Figura 5 – Pseudocódigo da construção HA-IMD.

Método de Busca Local

Este trabalho propõe também um novo procedimento de busca local, que a partir de agora será denotado por Busca Local Proposta (**BLP**). A **BLP** utiliza duas estruturas de vizinhanças: na primeira delas *V1*, é usada a busca local proposta em [Gh96] aqui denominada **BLG**.

Em seguida, a solução obtida na **BLG** passa a ser a solução corrente e daí prossegue-se com a segunda estrutura de vizinhança denotada *V2* que é semelhante à primeira, com a única diferença que, ao invés de a cada passo um elemento *i* da solução ser trocado por outro *j* fora da solução, são permutados simultaneamente dois elementos da solução por dois fora da solução. A cada troca a melhoria obtida pela nova solução $\Delta z(i, j, l, k)$ onde $i, j \in M$ e $l, k \in N - M$ é avaliada, conforme demonstrado na equação *20*.

$$\Delta z(i,j,l,k) = \sum_{u \in M-\{i,j\}} d(l,u) + d(k,u) - [d(i,u) + d(j,u)] \qquad (20)$$

A **BLP** é encerrada quando nenhuma melhora for obtida após permutar todos os pares internos e externos ao conjunto *M* atual. O pseudocódigo da **BLP** é apresentado na Figura 6. Observe que a BLP, embora utilize duas estruturas de vizinhança não é baseado na metaheurística VNS [HM05].

```
proc BLP (Solucao)
    Solucao = BLG (Solucao)
    DeltaZ = INFINITO
    enquanto DeltaZ > 0 faca
        Max_Deltaz = 0
        para cada i ∈ Solucao faca
            j = i + 1
            para cada l ∈ N\Solucao faca
                k = l + 1
                DeltaZ = calculaDeltaZ (i,j,l,k)
                se Deltaz > Max_Deltaz entao
                    Max_Deltaz = Deltaz
                    Max_I = i
                    Max_J = j
                    Max_L = l
                    Max_K = k
                fim se
            fim para
        fim para
        se Max_Deltaz > 0 entao
            Solucao = Solucao - {Max_I, Max_J}
            Solucao = Solucao ∪ {Max_L, Max_K}
            Deltaz = Max_Deltaz
        fim se
    fim enquanto
    retorna (Solucao)
fim BLG
```

Figura 6 – Pseudocódigo da Busca Local Proposta.

7. Resultados Computacionais

A fim de avaliar o desempenho das heurísticas desenvolvidas para o PDM, foram feitas combinações de métodos de construção com os de busca local para serem inseridos na estrutura do método GRASP. As versões propostas são apresentadas na Tabela 4. Estes algoritmos foram implementados usando linguagem de programação C++ e compilados na versão 3.2.2 do compilador GNU g++.

Tabela 4 – Algoritmos GRASP propostos.

Algoritmo GRASP	Construção	Busca Local
KMD+BLG	HA-KMD	BLG
KMD+BLP	HA-KMD	BLP
KMD2+BLG	HA-KMD2	BLG
KMD2+BLP	HA-KMD2	BLP
IMD+BLG	HA-IMD	BLG
IMD+BLP	HA-IMD	BLP

Para que comparações pudessem ser realizadas com heurísticas GRASP da literatura, foi implementado também o algoritmo proposto em [Gh96], aqui denominado **CGh+BLG**. Além disso, foi implementada uma variação deste que usa a construção proposta em [Gh96] e a Busca Local Proposta (**BLP**), chamada **CGh+BLP**. O código gerado em [An03] foi cedido pelos autores ao que se denominou **CAn+BLG**. Os testes foram executados em um AMD Athlon 1.3 GHz com 256 Mb de RAM usando sistema operacional Mandrake Linux 9.1.

Na realização dos testes, dois conjuntos de instâncias foram utilizados. O primeiro a ser apresentado foi desenvolvido em [An03]. Estas instâncias apresentam conjuntos N com cardinalidade n = 50, 100, 150, 200 e 250 elementos. No total são 15 instâncias divididas em:

Grupo I: Índices de diversidade gerados aleatoriamente em uma distribuição de números inteiros entre 1 e 9999.

Grupo J: Índices de diversidade gerados aleatoriamente, com 50% deles obedecendo a uma distribuição de números inteiros entre 1 e 9999 e os demais a partir de uma distribuição de números inteiros entre 1 e 4999.

Grupo K: Índices de diversidade gerados aleatoriamente, com 50% deles obedecendo a uma distribuição de números inteiros entre 1 e 9999 e os demais a partir de uma distribuição de números inteiros entre 5000 e 9999.

O segundo conjunto de instâncias foi proposto em [SOM04]. O número de elementos n destas instâncias tem valores iguais a 10, 20, 30, 40, 50, 100, 200, 300 e 500 e para cada valor de $n = |N|$ foram geradas quatro instâncias distintas, com valores de $m = |M|$ iguais a 10%, 20%, 30% e 40% de n. Desta forma, foi criado um total de 36 instâncias. Estas instâncias têm os índices de diversidade entre cada par de elementos gerados aleatoriamente em uma distribuição uniforme de números entre 0 e 9.

A fim de caracterizar as instâncias, para cada valor de m associamos uma letra: A, B, C e D correspondendo a 10%, 20%, 30% e 40% de n, respectivamente. Assim, a nomenclatura 10B, por exemplo, significa uma instância de $|N| = n = 10$ elementos onde se deseja maximizar a diversidade entre 20% de n (m = dois elementos). Excepcionalmente, no segundo conjunto, a instância 10A é desconsiderada pelo fato de não representar um problema real.

7.1 Testes com Conjunto de Instâncias de [An03]

Para todos os grupos que compõem este conjunto de instâncias, o critério de parada de cada algoritmo GRASP foi o número de 500 iterações. Cada instância foi executada três vezes por cada algoritmo heurístico e os valores médios de execução e tempos médios foram calculados.

As Tabelas 5, 6 e 7 apresentam um desempenho médio de três execuções de cada algoritmo para cada instância dos Grupos I, J e K. Ainda nas Tabelas 5, 6, e 7, a coluna 1 mostra a instância em questão e as colunas 2 - 10 descrevem as diferenças médias entre a melhor solução obtida considerando todos os algoritmos analisados e média das soluções obtida por cada algoritmo, chamada *dif* e calculada conforme a equação 21. Células com o símbolo (-) significam que esta diferença foi igual a zero. A última linha da Tabela 5 mostra a média geral de *dif* para cada algoritmo. Obviamente quanto menor for esta média, melhor será o desempenho do algoritmo.

$$dif = (melhorsolução - médiasoluções) / melhorsolução * 100 \qquad (21)$$

Tabela 5 – Diferença média entre as soluções no Grupo I.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	-	-	-	-	-	-	-	-	-
50B	-	-	-	-	-	-	-	-	-
50C	-	-	-	-	-	-	-	-	-
50D	-	-	-	-	-	-	-	-	-
100A	-	-	-	-	-	-	0,41	0,41	-
100B	-	-	-	-	-	-	-	-	-
100C	-	-	-	-	-	-	-	-	-
100D	-	-	-	-	-	-	-	-	-
150A	-	-	-	-	-	-	2,58	2,58	-
150B	-	-	-	-	-	-	-	-	-
150C	0,01	-	-	-	-	-	-	-	0,09
150D	-	-	-	-	-	-	0,01	-	0,06
200A	0,05	0,12	0,07	0,05	-	0,05	0,72	0,72	0,39
200B	-	-	-	-	0,04	0,04	-	-	0,01
200C	-	-	0,04	0,04	-	-	-	-	0,18
200D	-	-	-	-	-	-	-	-	0,10
250A	-	-	0,04	0,04	-	-	-	-	0,25
250B	0,09	0,14	0,04	0,06	0,05	0,02	0,01	-	0,43
250C	0,04	0,07	-	-	0,14	0,09	0,08	0,08	0,25
250D	0,01	-	0,01	-	0,02	0,02	0,05	0,05	0,10
Média	**0,01**	0,02	**0,01**	**0,01**	**0,01**	**0,01**	0,21	0,21	0,09

Tabela 6 – Diferença média entre as soluções no Grupo J.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	-	-	-	-	-	-	5,20	5,20	-
50B	-	-	-	-	-	-	-	-	-
50C	-	-	-	-	-	-	-	-	-
50D	-	-	-	-	-	-	-	-	-
100A	-	-	-	-	-	-	-	-	-
100B	-	-	-	-	-	-	-	-	-
100C	-	-	-	-	-	-	-	-	-
100D	-	-	-	-	-	-	-	-	-
150A	-	-	-	-	-	-	-	-	-
150B	-	-	-	-	-	-	-	-	-
150C	-	-	-	-	-	-	-	-	0,06
150D	-	-	-	-	-	-	-	-	-
200A	-	-	-	-	-	-	-	-	0,29
200B	-	-	-	-	-	-	-	-	0,09
200C	-	-	-	-	-	-	-	-	0,01
200D	-	-	-	-	-	-	-	-	-
250A	-	-	-	-	-	-	-	-	0,42
250B	-	-	-	-	-	-	-	-	0,08
250C	-	-	-	-	-	-	-	-	0,02
250D	-	-	-	-	-	-	-	-	-
Média	-	-	-	-	-	-	0,26	0,26	0,05

Tabela 7 – Diferença média entre as soluções no Grupo K.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	-	-	-	-	-	-	-	-	-
50B	-	-	-	-	-	-	0,16	-	0,16
50C	-	-	-	-	-	-	-	-	-
50D	-	-	-	-	-	-	-	-	-
100A	-	-	-	-	-	-	-	-	0,26
100B	-	-	-	-	-	-	-	-	-
100C	-	-	-	-	-	-	-	-	-
100D	-	-	-	-	-	-	-	-	-
150A	-	-	-	-	-	-	0,23	-	0,23
150B	-	-	-	-	-	-	0,12	-	0,12
150C	-	-	-	-	0,01	0,01	-	-	-
150D	-	-	-	-	-	-	0,01	-	0,01
200A	-	-	-	-	-	-	0,34	-	0,34
200B	-	-	-	-	-	-	0,12	-	0,12
200C	-	-	-	-	-	-	0,04	-	0,04
200D	-	-	-	-	-	-	0,04	-	0,04
250A	-	-	-	-	0,02	0,02	0,23	0,02	0,23
250B	-	-	-	-	-	-	0,17	-	0,17
250C	-	-	-	-	-	-	0,11	-	0,11
250D	-	-	-	-	-	-	0,09	-	0,09
Média	-	-	-	-	-	-	0,08	-	0,10

É possível observar que em todos os grupos de instâncias analisados até o momento, os algoritmos propostos **KMD+BLG**, **KMD+BLP**, **KMD2+BLG**, **KMD2+BLP**, **IMD+BLG** e **IMD+BLP** obtiveram as melhores soluções na média. Isto pode ser visto na tabela 8 que mostra, para cada grupo, o número de vezes em que cada algoritmo obteve uma solução de melhor valor no total de 20 testes.

Tabela 8 – Número de vezes em que a melhor solução é encontrada por cada algoritmo em 20 instâncias.

Algoritmo	Grupo I	Grupo J	Grupo K
KMD+BLG	15	**20**	**20**
KMD+BLP	17	**20**	**20**
KMD2+BLG	15	**20**	**20**
KMD2+BLP	16	**20**	**20**
IMD+BLG	16	**20**	18
IMD+BLP	15	**20**	18
CGh+BLG	14	**19**	8
CGh+BLP	14	**19**	**19**
CAn+BLG	10	**17**	7

Para as instâncias do Grupo I, as heurísticas propostas estiveram em média a no máximo 0,02% da melhor solução enquanto o algoritmo de [Gh96] se manteve a 0,21% e o de [An03] a 0,09%. Pela Tabela 8, em um total de 20 instâncias, **KMD+BLP** obteve maior sucesso, chegando à melhor solução 17 vezes. Os algoritmos da literatura **CGh+BLG** e sua variação **CGh+BLP** obtiveram melhor soluções em 14 instâncias e **CAn+BLG** em 10 das 20 instâncias.

No Grupo J, os algoritmos propostos **KMD+BLG**, **KMD+BLP**, **KMD2+BLG**, **KMD2+BLP**, **IMD+BLG** e **IMD+BLP** obtiveram, em todos os testes, o melhor valor. Já os algoritmos da literatura **CGh+BLG** e sua variação **CGh+BLP** ficaram, ambos, em média, a 0,26% da melhor solução e **CAn+BLG** a 0,05%.

No Grupo K, os algoritmos propostos **KMD+BLG**, **KMD+BLP**, **KMD2+BLG**, **KMD2+BLP** obtiveram, mais uma vez, as melhores soluções em 100% dos casos. **IMD+BLG** e **IMD+BLP** alcançaram a melhor solução em 18 das 20 instâncias, além disso, as duas heurísticas estiveram a, no máximo, 0,02% da melhor solução. **CGh+BLG** alcançou melhores soluções em somente 8 dos 20 testes. Quanto a **CGh+BLP**, nota-se que a **BLP** pôde melhorar o desempenho apresentado pelo GRASP original de Ghosh, chegando à melhor solução em 19 do total de instâncias do grupo. O algoritmo de [An03] encontrou a melhor solução em 7 dos 20 testes e manteve-se a 0,10% das melhores soluções apresentadas, a pior diferença observada dentre os algoritmos. Um fato que pode ser observado através dos experimentos é que conforme há o aumento do tamanho das instâncias, o algoritmo da literatura **CAn+BLG** tem seu desempenho piorado. As Tabelas 9, 10 e 11 apresentam a média dos tempos em segundos gastos por cada algoritmo nas instâncias do Grupo I, Grupo J e Grupo K, respectivamente.

Tabela 9 – Tempo médio computacional (segundos) no grupo I.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	2,6	2,6	3,9	3,8	3,0	3,0	2,4	2,4	0,0
50B	4,5	4,6	7,2	7,4	5,8	6,0	4,6	4,8	0,1
50C	6,9	7,2	10,7	11,0	8,6	8,9	6,7	7,1	0,4
50D	9,8	10,2	14,0	14,5	11,4	11,8	8,7	9,0	0,7
100A	7,9	8,2	28,7	28,7	22,6	22,8	15,5	15,7	0,3
100B	21,4	22,6	62,1	62,9	50,7	51,7	38,7	38,1	2,3
100C	40,7	43,0	100,9	102,5	84,1	87,3	64,4	66,5	6,7
100D	62,6	66,4	139,4	144,0	125,2	128,9	93,5	94,7	13,0
150A	21,9	23,0	102,7	103,5	83,0	83,9	56,2	56,3	1,9
150B	76,5	80,4	240,8	243,7	202,6	205,8	146,4	155,3	12,4
150C	164,1	173,0	415,4	421,2	346,1	354,2	288,1	292,2	34,6
150D	267,7	283,4	607,0	620,1	542,3	551,2	449,3	431,6	70,1
200A	51,6	54,3	263,1	262,6	213,4	216,2	146,6	148,1	6,7
200B	214,8	221,8	644,6	645,8	556,3	567,0	425,4	427,4	41,1
200C	490,0	508,4	1149,8	1175,5	1006,6	1028,9	820,5	825,4	113,7
200D	796,4	830,4	1700,1	1744,1	1568,1	1603,0	1108,8	1116,2	228,9
250A	108,7	114,5	536,5	542,1	445,4	451,5	322,5	317,6	17,1
250B	505,3	525,2	1384,3	1404,2	1230,0	1238,7	982,7	974,6	101,8
250C	1167,2	1204,8	2568,6	2647,8	2350,8	2373,3	1854,2	1889,7	278,4
250D	1885,5	1931,0	3876,5	4023,3	3298,4	3351,0	2696,0	2744,1	579,1

Tabela 10 – Tempo médio computacional (segundos) no grupo J.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	2,7	2,8	3,8	4,0	3,0	3,0	2,5	2,3	0,0
50B	4,8	7,8	7,4	7,8	5,8	5,9	4,5	4,1	0,1
50C	7,3	10,5	11,2	11,9	8,3	8,5	6,0	5,6	0,3
50D	9,8	8,9	15,0	16,0	10,4	10,8	6,9	6,6	0,3
100A	8,4	23,7	28,5	29,9	22,5	22,8	16,5	14,4	0,3
100B	21,8	43,9	62,8	66,5	49,1	50,2	36,8	32,6	2,0
100C	40,2	69,6	103,1	111,2	77,6	79,9	52,5	48,7	4,2
100D	64,1	23,8	159,3	171,3	102,4	106,0	67,1	63,4	8,2
150A	21,9	82,3	103,0	110,0	83,7	84,4	57,7	52,8	1,7
150B	75,8	174,1	246,2	262,0	189,3	192,7	133,9	126,0	11,6
150C	159,3	207,6	432,5	453,1	320,2	328,7	210,4	198,0	33,1
150D	262,6	275,4	652,3	650,8	419,7	430,9	226,3	263,4	54,0
200A	52,6	57,1	263,5	264,7	213,3	216,2	141,8	131,5	5,7
200B	218,5	241,1	676,6	685,2	499,1	508,9	399,3	375,1	36,8
200C	468,9	507,6	1220,2	1236,4	809,5	835,8	637,9	596,3	106,8
200D	711,4	770,1	1788,4	1812,9	1090,7	1139,5	874,7	825,9	156,3
250A	111,0	115,9	559,3	554,2	454,4	453,7	312,7	308,2	13,9
250B	511,7	527,7	1497,2	1500,2	1091,8	1129,9	919,6	923,9	92,9
250C	1158,9	1177,2	2860,7	2873,8	1779,2	1862,9	1701,9	1785,0	267,2
250D	1735,4	1794,4	4111,5	4176,4	2617,3	2685,2	2472,7	2525,1	467,2

Tabela 11 – Tempo médio computacional (segundos) no grupo K.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
50A	2,3	2,2	3,6	3,6	2,8	2,8	2,3	2,3	0,0
50B	4,0	4,0	6,7	6,8	5,3	5,4	4,1	4,2	0,1
50C	6,1	6,2	9,8	10,1	7,8	8,1	5,7	6,0	0,4
50D	8,9	9,0	12,8	13,2	10,2	10,5	7,5	7,9	0,8
100A	7,6	7,5	26,3	26,6	20,5	20,7	14,3	14,6	0,3
100B	20,6	20,8	58,7	59,6	47,3	48,2	34,4	35,3	2,3
100C	39,1	40,8	96,9	98,9	79,6	81,6	56,9	58,9	6,0
100D	58,6	61,1	133,4	136,5	114,2	117,6	84,0	87,2	11,4
150A	19,5	20,4	94,7	95,8	76,5	77,3	51,0	51,7	1,8
150B	69,1	72,4	223,2	227,2	180,4	184,0	124,8	129,3	11,5
150C	149,3	155,3	388,5	396,0	314,4	321,2	235,7	243,3	30,98
150D	241,9	250,9	564,1	575,2	463,9	474,6	329,1	339,3	53,3
200A	47,8	50,0	244,9	246,9	201,7	204,2	132,8	134,9	6,5
200B	201,8	209,4	623,0	631,7	538,4	547,3	372,4	382,2	38,5
200C	468,5	482,9	1137,5	1154,0	917,6	938,8	715,4	733,4	106,5
200D	716,5	740,4	1626,5	1652,8	1267,8	1298,2	1159,4	1178,4	199,4
250A	143,4	148,8	570,9	577,9	468,3	475,0	301,6	308,6	16,1
250B	605,2	624,1	1578,5	1603,4	1373,3	1390,4	979,0	1002,6	96,2
250C	1291,3	1328,3	2925,0	2975,9	2335,7	2367,3	2000,2	2064,8	264,3
250D	1864,0	1922,6	4273,1	4326,3	3320,0	3377,3	3000,2	3090,6	606,9

A partir dos resultados parciais observa-se que a **BLP** demanda tempo de computação maior. Isto pode ser visto comparando-se os algoritmos que utilizam o mesmo método de construção mas que diferem entre si pelo uso da busca local **BLG** ou **BLP**, como ocorre por exemplo com propostos **KMD+BLG** e **KMD+BLP**, **KMD2+BLG** e **KMD2+BLP**, **IMD+BLG** e **IMD+BLP**. Contudo, este resultado era esperado já que na **BLP** uma busca completa **BLG** é realizada antes da mudança da estrutura de vizinhança.

Verifica-se também que a heurística proposta em [An03] demanda um tempo menor de computação compensando a fraca qualidade das suas soluções. Em seguida, os algoritmos mais econômicos em relação aos tempos exigidos são os propostos **KMD+BLG** e **KMD+BLP.** No topo como o mais exigente em termos de tempos gastos está o **KMD2+BLP**. Nos testes aqui efetuados observou-se também uma grande similaridade nos tempos de **KMD2+BLG**, **IMD+BLG**, **IMD+BLP**, **CGh+BLP** e o algoritmo da literatura **CGh+BLG**.

Deve-se alertar que a análise no desempenho em relação aos tempos computacionais exigidos são aqui feitos a partir do critério de parada utilizado (número máximo de iterações). Entretanto, se colocarmos outros critérios de parada, podemos ter outras interpretações em relação ao tempo computacional, como mostrado na análise probabilística feita adiante.

7.2 Testes com Conjunto de Instâncias de [SOM04]

Com este segundo conjunto de instâncias foram realizadas três diferentes baterias de testes que serão descritas a seguir. Na primeira e segunda bateria de testes, cada algoritmo foi executado três vezes para cada instância usando diferentes sementes para geração de números aleatórios e em cada execução foram usadas novamente 500 iterações GRASP como critério de parada. Na primeira bateria de testes, o valor objetivo da solução obtido pelas heurísticas GRASP é comparado com o valor da solução exata. Para obter soluções exatas foi utilizada a formulação de programação linear inteira (P2) proposta em [KGD95] (e descrita na seção 3) e o *software* GLPK (GNU *Linear Programming Kit*) [Ma04] que utiliza um método *branch and bound* e *dual simplex* em sua implementação. Para estes experimentos foram consideradas somente as instâncias com valores de $n \leq 100$.

As Tabelas 12 e 13 descrevem os resultados deste experimento. A Tabela 12 mostra na coluna 1 a instância em questão, na coluna 2 o valor ótimo (exato) e nas colunas 3-11, a média dos valores das soluções obtidas nas três execuções dos algoritmos heurísticos. Na Tabela 12, valores em negrito denotam o valor ótimo (quando este é conhecido ou a melhor solução obtida considerando todos os algoritmos aqui analisados). Neste último caso, ocorre nas instâncias 40C até 100D, onde o método exato foi truncado após um tempo de execução máxima pré-estabelecido igual a 36.000 segundos.

Para os testes com instâncias de 10B até 40B o GLPK pôde encontrar solução exata em tempo computacional viável.

No entanto, quando o teste foi executado para a instância 40C, o algoritmo ficou em execução por 25 dias e ainda sim não havia terminado o processamento. A partir daí, limitou-se o tempo de processamento em 36.000 segundos e a melhor resposta obtida foi considerada (valores em * na segunda coluna da Tabela 12).

Tabela 12 – Valores médios das soluções exatas e heurísticas onde (*) indica solução de maior valor obtido com tempo de processamento de 10000 segundos.

Inst.	Exato	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
10B	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
10C	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
10D	**47**	**47**	**47**	**47**	**47**	**47**	**47**	**47**	**47**	40
20A	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
20B	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**
20C	**109**	**109**	**109**	**109**	**109**	**109**	**109**	**109**	**109**	**109**
20D	**181**	**181**	**181**	**181**	**181**	**181**	**181**	180	180	**181**
30A	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**27**
30B	**121**	**121**	**121**	**121**	**121**	**121**	**121**	**121**	**121**	**121**
30C	**254**	**254**	**254**	**254**	**254**	**254**	**254**	**254**	**254**	**254**
30D	**412**	**412**	**412**	**412**	**412**	**412**	**412**	**412**	**412**	**412**
40A	**53**	**53**	**53**	**53**	**53**	**53**	**53**	49	49	**53**
40B	**198**	**198**	**198**	**198**	**198**	**198**	**198**	192	192	**198**
40C	**412***	**412**	**412**	**412**	**412**	**412**	**412**	**412**	**412**	**412**
40D	681*	**688**	**688**	**688**	**688**	**688**	**688**	**688**	**688**	**688**
50A	78*	**86**	**86**	**86**	**86**	**86**	**86**	81	81	81
50B	268*	**311**	**311**	**311**	**311**	**311**	**311**	**311**	**311**	309
50C	602*	**652**	**652**	**652**	**652**	**652**	**652**	**652**	**652**	627
50D	974*	**1087**	**1087**	**1087**	**1087**	**1087**	**1087**	**1087**	**1087**	1033
100A	240*	**333**	**333**	**333**	**333**	**333**	**333**	318	318	**333**
100B	911*	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	1178	1178	1191,7
100C	2034*	**2457**	**2457**	**2457**	**2457**	**2457**	**2457**	**2457**	**2457**	**2457**
100D	3574*	**4142**	**4142**	**4142**	**4142**	**4142**	**4142**	**4142**	**4142**	4140,3

Analisando-se os resultados obtidos na Tabela 12, pode-se concluir que os algoritmos **KMD+BLG**, **KMD+BLP**, **KMD2+BLG**, **KMD2+BLP**, **IMD+BLG** e **IMD+BLP** propostos nesse trabalho foram capazes de encontrar uma solução ótima em 100% das instâncias onde se conhece o valor ótimo ou um ***valor médio melhor*** que o valor encontrado pelo algoritmo exato truncado em 100% das instâncias (que tiveram o tempo de computação do GLPK limitado).

A heurística da literatura **CGh+BLG** e sua variação, **CGh+BLP** foram, ambas, capazes de encontrar solução ótima e solução igual ou superior (melhor) à solução encontrada pelo algoritmo exato com tempo limitado em 17 das 23 instâncias testadas. O algoritmo **CAn+BLG** encontrou os melhores valores em 16 das 23 instâncias.

Em média, **CGh+BLG** e **CGh+BLP** estiveram a 0,99% do melhor custo e **CAn+BLG** a 1,32%.

A Tabela 13 mostra na coluna 1 a instância, na coluna 2 o tempo de processamento para obter uma solução exata e nas colunas 3-11, o tempo médio de processamento, em segundos, utilizado para obter as soluções usando os algoritmos GRASP. Quanto a tempo computacional verifica-se um crescimento exponencial do algoritmo exato e tempos bem menores nos algoritmos GRASP.

Tabela 13 – Tempo médio computacional (segundos) onde (**) indica tempo de processamento limitado em 36000 segundos.

Inst.	Exato	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
10B	0,02	0,02	0,03	0,05	0,05	0,03	0,03	0,03	0,03	0,00
10C	0,09	0,03	0,03	0,08	0,07	0,05	0,05	0,04	0,04	0,01
10D	0,14	0,03	0,04	0,09	0,10	0,07	0,07	0,05	0,05	0,01
20A	2,00	0,09	0,09	0,19	0,20	0,13	0,12	0,12	0,12	0,00
20B	10,00	0,12	0,13	0,41	0,42	0,34	0,34	0,22	0,24	0,01
20C	24,00	0,17	0,20	0,59	0,63	0,53	0,54	0,33	0,35	0,02
20D	26,00	0,23	0,27	0,77	0,82	0,70	0,72	0,39	0,42	0,03
30A	126,00	0,22	0,23	0,73	0,75	0,57	0,58	0,40	0,41	0,01
30B	653,00	0,36	0,41	1,54	1,56	1,32	1,36	0,82	0,87	0,03
30C	1413,00	0,63	0,72	2,26	2,29	2,04	2,09	1,20	1,27	0,06
30D	2862,00	0,99	1,13	2,86	2,93	2,70	2,77	1,56	1,66	0,13
40A	2374,00	0,44	0,46	2,18	2,13	1,58	1,55	1,03	1,24	0,02
40B	163931,00	0,90	1,05	4,73	4,57	3,52	3,55	2,07	2,75	0,10
40C	**	0,99	1,69	7,11	7,24	5,42	5,46	3,09	4,50	0,22
40D	**	2,25	2,70	9,27	10,09	7,36	7,39	4,19	6,42	0,48
50A	**	0,58	0,61	3,06	3,10	2,31	2,36	1,76	1,79	0,04
50B	**	1,69	1,87	6,89	7,05	5,55	5,80	4,53	4,67	0,22
50C	**	3,26	3,71	11,34	11,81	9,17	9,62	7,41	7,83	0,59
50D	**	5,14	5,79	16,11	16,70	14,12	14,71	10,55	11,11	1,12
100A	**	10,4	11,8	28,0	30,2	22,3	23,1	18,0	19,3	0,4
100B	**	28,2	37,0	64,8	79,5	53,3	60,5	47,4	64,7	2,8
100C	**	57,4	79,1	111,7	148,3	91,0	113,6	88,3	123,4	7,5
100D	**	89,2	126,2	165,6	234,4	138,8	187,1	130,9	198,7	14,6

Na segunda bateria de testes deste segmento, foram usadas as instâncias 200A até 500D. Estes testes têm como objetivo comparar os resultados obtidos pelos algoritmos propostos com os da literatura, para instâncias de porte maior.

As Tabelas 14, 15 e 16 mostram os resultados médios obtidos. A Tabela 14 apresenta a média das soluções obtidas pelas heurísticas. Em negrito estão os valores das melhores soluções.

Tabela 14 – Valores médios das soluções heurísticas.

Inst.	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
200A	**1247,0**	**1247,0**	**1247,0**	**1247,0**	1245,7	**1247,0**	1233,0	1233,0	1243,0
200B	4446,3	**4449,3**	4447,0	4448,0	4446,7	4446,0	4443,0	4442,3	4443,3
200C	**9437,0**	**9437,0**	**9437,0**	**9437,0**	9434,3	**9437,0**	**9437,0**	**9437,0**	9424,0
200D	16182,3	16170,0	**16225,0**	**16225,0**	16224,7	**16225,0**	**16225,0**	**16225,0**	16200,0
300A	2686,0	2683,0	2688,5	**2692,0**	2684,0	2679,0	2666,0	2666,0	2681,3
300B	9667,7	9658,0	**9682,5**	9676,0	9678,0	9678,3	9652,0	9644,3	9658,0
300C	20640,0	20640,0	20726,0	**20727,0**	20721,3	207223	20725,0	20725,0	20673,7
300D	35871,0	35871,0	35878,0	**35881,0**	35880,0	35877,0	35880,0	35880,3	35853,5
400A	4635,3	4634,7	4647,0	4654,0	4648,0	**4654,7**	4615,0	4611,0	4629,7
400B	16916,3	16902,0	16930,0	16938,0	**16946,7**	16944,7	16900,5	16905,3	16873,3
400C	36175,0	36175,0	36272,5	36290,0	36301,0	**36301,7**	36298,7	36301,0	36187,7
400D	62313,0	62313,0	**62478,0**	62451,5	62450,0	62439,3	62442,3	62432,0	62345,0
500A	**7124,0**	7116,0	7112,5	7107,0	7110,7	7105,3	7079,3	7079,0	7082,7
500B	26197,0	26221,0	26229,0	**26229,5**	262130,0	26220,0	26219,0	26219,0	26219,0
500C	57055,7	56571,7	**56572,0**	56571,0	56549,0	56544,7	56571,0	56571,5	56360,0
500D	97213,0	**97333,0**	97316,5	97310,0	97316,0	97325,5	97255,0	97322,3	97166,7

A Tabela 15 mostra o número de vezes em que cada algoritmo atingiu a melhor solução em um total de 16 instâncias.

Tabela 15 – Número de vezes em que cada algoritmo atingiu a melhor solução.

Algoritmo	#
KMD+BLG	3
KMD+BLP	4
KMD2+BLG	6
KMD2+BLP	7
IMD+BLG	1
IMD+BLP	5
CGh+BLG	2
CGh+BLP	2
CAn+BLG	0

Observa-se que os algoritmos **KMD2+BLP**, **KMD2+BLG** e **IMD+BLP** obtiveram o melhor desempenho médio. Quanto aos algoritmos da literatura, em nenhuma das instâncias **CAn+BLG** chegou a uma melhor solução. **CGh+BLG** e **CGh+BLP** alcançaram melhores soluções somente em 2 instâncias dentre as testadas. A partir da Tabela 14 determinamos a diferença percentual média entre a melhor solução de cada instância e a média de cada algoritmo.

Os de melhores resultados foram obtidos por **KMD2+BLP**, **KMD2+BLG** e **IMD+BLP** com 0,08%, 0,10% e 0,12% da melhor solução e os algoritmos da literatura **CAn+BLG**, **CGh+BLG** e **CGh+BLP** a ficaram a 0,33%, 0,34% e 0,33% da melhor solução, respectivamente.

De uma forma geral, nota-se que, apesar dos algoritmos da literatura serem capazes de obter boas soluções para o problema, os algoritmos propostos se mostraram mais eficientes em termos da qualidade da solução obtida.

A Tabela 16 mostra o tempo de processamento médio em segundos de cada algoritmo. Pode-se observar que o algoritmo da literatura **CAn+BLG** mostra-se novamente como o mais rápido (baseado no critério de parada utilizado) sendo seguido pelo algoritmo proposto **KMD+BLG**.

Tabela 16 – Tempo médio computacional (segundos).

Inst	KMD+BLG	KMD+BLP	KMD2+BLG	KMD2+BLP	IMD+BLG	IMD+BLP	CGh+BLG	CGh+BLP	CAn+BLG
200A	71,7	101,1	263,6	301,1	224,3	247,6	147,5	203,4	6,3
200B	318,3	510,3	784,7	986,3	635,6	835,6	561,9	825,3	39,3
200C	626,6	1006,4	1958,8	2807,0	1224,0	1715,6	1075,3	1665,9	113,5
200D	804,3	1329,0	3051,8	4502,5	1850,0	2673,7	1647,0	2629,7	225,2
300A	412,7	644,8	1159,4	1405,3	979,5	1189,9	712,7	908,1	32,3
300B	1690,5	2800,1	3450,4	4774,9	2954,4	3998,7	2477,2	3557,2	215,3
300C	3260,6	5164,7	6829,5	10514,6	5966,6	8582,0	5603,2	8151,1	605,3
300D	4453,0	8216,8	11458,5	17517,7	10329,5	15842,3	9740,2	15672,3	1184,6
400A	1208,7	2044,7	3202,7	4066,2	2790,0	3371,0	2469,3	3323,3	106,1
400B	5266,5	8077,1	10464,3	15275,1	10048,6	13772,0	9651,9	14833,6	682,5
400C	10097,5	16620,7	22038,4	33594,6	19201,3	28881,7	20210,3	32205,1	1346,0
400D	15750,3	26943,4	32843,6	52042,6	30010,7	45161,1	32217,8	51497,7	5064,3
500A	2634,9	4555,8	6988,6	9171,3	5625,8	7452,6	5122,4	7304,3	254,5
500B	11921,7	19200,8	26513,3	39343,5	22781,3	32888,1	26128,9	31542,5	1611,3
500C	29051,4	65082,3	55898,1	88158,7	49617,7	76443,5	54392,6	76068,6	7922,9
500D	39043,3	86247,0	84678,4	137503,3	75998,8	117932,6	80252,1	115137,3	28842,6

7.2.1 Análise Probabilística

Um gargalo observado em alguns algoritmos heurísticos e metaheurísticas da literatura é a sua instabilidade ou a falta de regularidade em seu desempenho. Em outras palavras, é comum ocorrer que uma determinada heurística apresente um bom desempenho para uma classe de problemas ou, mais especificamente, para algumas instâncias e para outras, tenha um desempenho ruim. Um caso mais extremo é observado nas metaheurísticas que possuem componentes probabilísticas, como é o caso dos algoritmos genéticos, alguns algoritmos evolutivos e o GRASP. Nestes métodos, na solução do mesmo problema teste com o mesmo conjunto de dados de entrada, é possível verificar uma variação considerável na qualidade dos resultados a cada execução do algoritmo. No caso específico do GRASP, em algumas versões vistas na literatura, a cada nova execução tem-se uma grande chance de se obter ao

final uma solução distinta das anteriores. Desta forma, pode-se ter soluções de ótima qualidade e numa segunda execução, soluções muito fracas. Esta variação é extremamente ruim para a confiabilidade do método.

Estes fatos indicam a importância de, a cada nova proposta de heurísticas GRASP ou algoritmos genéticos, procurar mostrar um estudo em termos da sua robustez mesmo que empiricamente. Isto justifica a proposta de uma nova bateria de testes, agora com o objetivo específico de analisar a regularidade dos algoritmos GRASP aqui apresentados.

Para este estudo, foram escolhidos alguns dos algoritmos que apresentaram as melhores soluções nos testes realizados até o momento e/ou aqueles com menor tempo de processamento. Seguindo este critério chegou-se a seguinte escolha: **KMD+BLG**, **KMD+BLP**, **KMD2+BLG** e **KMD2+BLP**, além dos algoritmos da literatura, **CGh+BLG** e **CAn+BLG**.

Neste tipo de experimento, realizado para as instâncias de 100D a 300D, são feitas 100 execuções independentes de cada algoritmo para cada instância, ou seja, um algoritmo é executado 100 vezes usando sementes distintas para a geração dos números aleatórios.

O critério de parada dos algoritmos é modificado para que a execução prossiga até que uma solução de valor igual ou melhor que um valor alvo estabelecido tenha sido encontrado. Em cada execução *i,* o tempo t_i para atingir o valor alvo é armazenado em ordem crescente.

Uma probabilidade empírica $p_i = (i - 0{,}5)/100$ é associada a cada tempo t_i. Desta forma, são *plotados* os pontos $z_i = (t_i, p_i)$, estabelecendo uma distribuição empírica de tempo para que um algoritmo alcance um determinado valor alvo [ARR02].

Neste trabalho, para cada instância foram calculados dois valores alvos definidos com base nos resultados anteriores. O primeiro alvo pode ser considerado mais fácil do que o segundo, já que este equivale ao valor da pior solução obtida dentre os algoritmos em questão enquanto o segundo corresponde à média dos valores das melhores soluções encontradas pelos algoritmos. A Tabela 17 ilustra os valores dos dois alvos usados para cada instância analisada.

Tabela 17 – Valores alvo utilizados.

Inst.	Alvo 1	Alvo 2
100D	4141	4142
200A	1244	1247
200B	4443	4447
200C	9425	9435
200D	16171	16210
300A	2666	2686
300B	9652	9676
300C	20640	20691
300D	35855	35873

Os resultados desta bateria de testes são apresentados nas Figuras 6-14. Nestes gráficos cada algoritmo é representado por uma curva e quanto mais a esquerda estiver a curva, melhor será o seu desempenho. Ainda deve-se observar que a probabilidade empírica de um algoritmo encontrar uma solução de valor melhor ou igual a determinado valor alvo em um

dado tempo de processamento aumenta da esquerda para a direita. De acordo com os experimentos realizados, os algoritmos propostos se mostram na seguinte ordem de desempenho em termos de convergência para valores sub-ótimos: **KMD+BLG**, **KMD+BLP**, **KMD2+BLG** e **KMD2+BLP**. Tomando-se como exemplo o primeiro gráfico apresentado na Figura 6, a probabilidade de que **KMD+BLG** encontre o valor alvo 1 em 10 segundos é de aproximadamente 85%, caindo para aproximadamente 68% no **KMD+BLP**, 40% no **CGh+BLG**, 35% no **KMD2+BLG**, 23% no **KMD2+BLP** e 18% para **CAn+BLG**.

Ao avaliar as Figuras 6-14, percebe-se que, os algoritmos **KMD+BLG** e **KMD+BLP** apresentam, na maioria dos casos, uma convergência mais rápida para os valores alvos estabelecidos. Percebe-se, ainda, que as heurísticas propostas seguem um comportamento bastante semelhante para as diferentes instâncias testadas. Este fato, no entanto, parece não ocorrer com os algoritmos da literatura **CGh+BLG** e **CAn+BLG**. Por exemplo, no segundo gráfico da Figura 7, o algoritmo **CGh+BLG** consegue alcançar probabilidades maiores de chegar à solução alvo que os demais algoritmos para maior parte dos tempos. Algo semelhante se observa no primeiro gráfico da Figura 9, mas, neste caso, **CGh+BLG** passa a apresentar probabilidades ligeiramente superiores aos demais algoritmos após decorridos 10 segundos. Por outro lado, tem-se que no alvo 2 da instância 300A (segundo gráfico da Figura 10) a curva de **CGh+BLG** não pôde ser plotada pois, até um tempo de computação de 150.000 segundos, o algoritmo não conseguiu atingir o valor alvo em diversas tentativas independentes. Isto volta a ocorrer no alvo 2 da instância 300B (segundo gráfico da Figura 11). Em algumas execuções foi possível atingir o alvo, mas o tempo médio para que isto ocorresse foi de 45.245 segundos. Supondo que em todas as execuções o alvo fosse atingido com este tempo médio seriam necessários mais de 45 dias para que somente este teste pudesse ser realizado.

Analisando-se o desempenho de **CAn+BLG** comparado aos demais algoritmos, observa-se que na maior parte dos experimentos o tempo para alcançar determinado valor alvo ao longo das instâncias avaliadas é maior. Isto somente não ocorre na instância 300A (Figura 10) onde **CAn+BLG** consegue alcançar, para um determinado tempo, maiores probabilidades de obter os valores alvos usados. No alvo 2 da instância 300D, o menor tempo para que o algoritmo chegasse ao alvo foi de 175.097 segundos. Como o algoritmo não apresentou convergência para o valor alvo em todas as execuções, esta curva não foi plotada. Como observado em testes anteriores, este algoritmo (**CAn+BLG**) sempre mostrou-se o mais rápido. Contudo nos testes desta seção ele apresenta a necessidade de mais tempo para atingir o valor alvo. Isso mostra o cuidado que é preciso ter na avaliação de desempenho de algoritmos heurísticos, pois um determinado critério de parada pode levar a conclusões equivocadas. Nos testes desta seção podemos concluir que nem sempre a heurística que possui uma iteração mais *lenta*, como os que usam a busca local **BLP** são os mais lentos se o critério de parada for um valor alvo. Ou seja, iterações mais onerosas podem ser compensadas com um menor número de iterações para atingir um determinado valor alvo sub-ótimo.

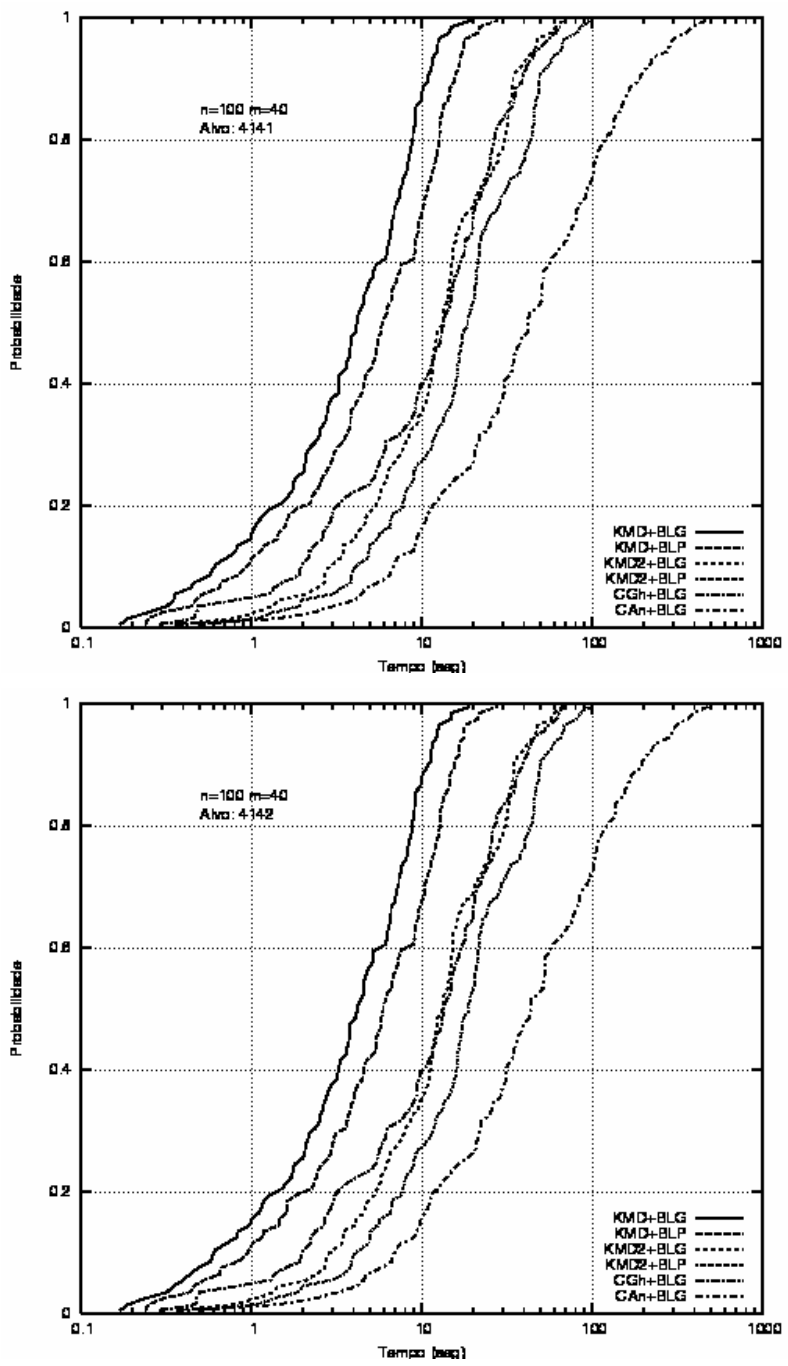

Figura 6 – Gráficos da instância 100D com alvo 1 = 4141 e alvo 2 = 4142.

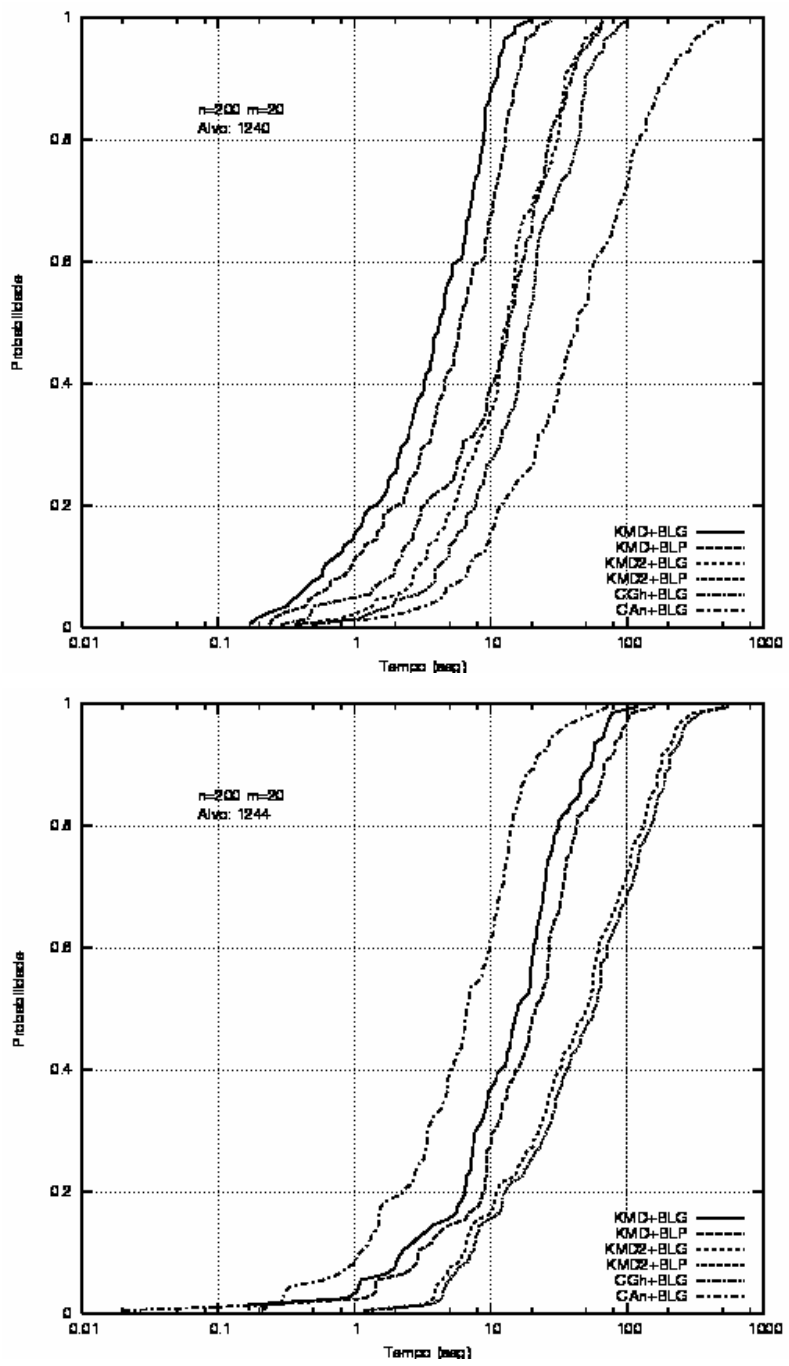

Figura 7 – Gráficos da instância 200A com alvo 1 = 1240 e alvo 2 = 1244.

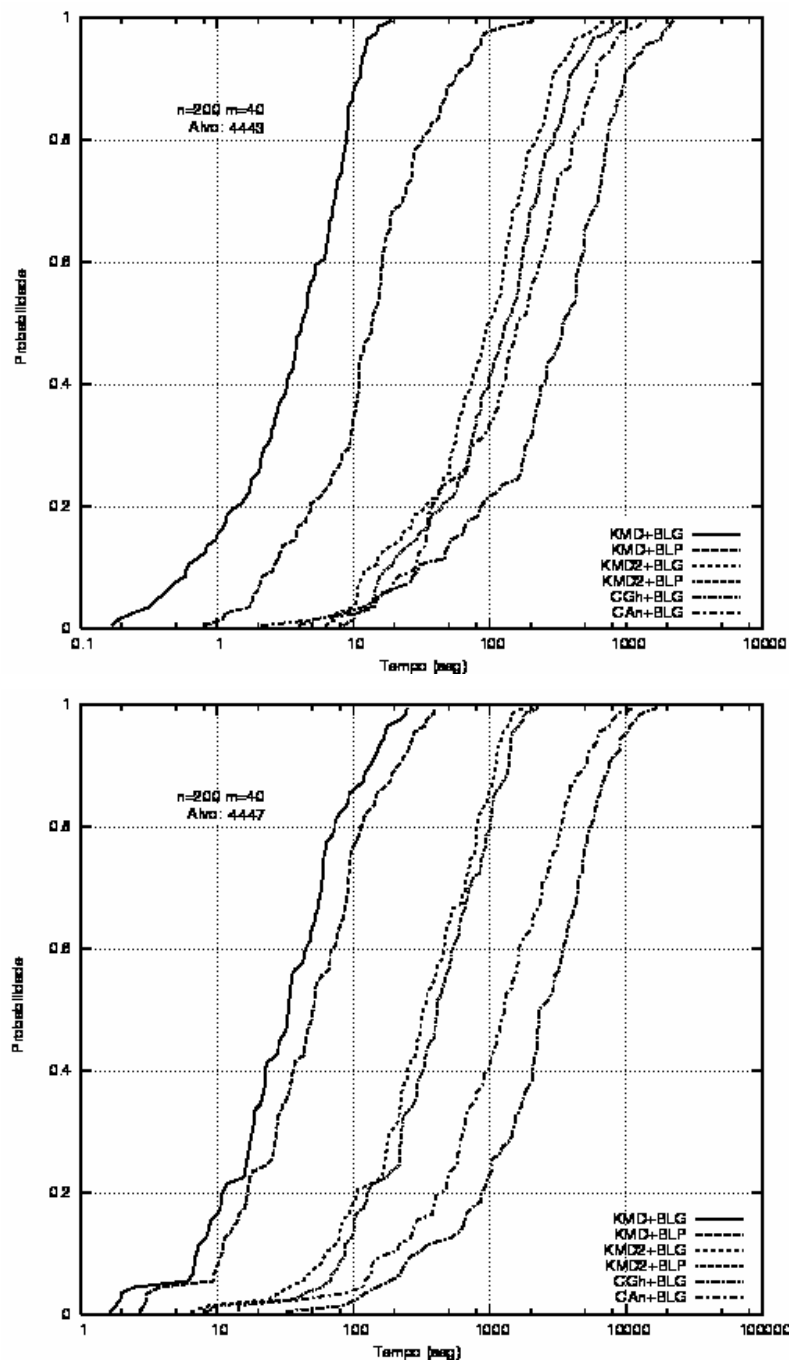

Figura 8 – Gráficos da instância 200B com alvo 1 = 4443 e alvo 2 = 4447.

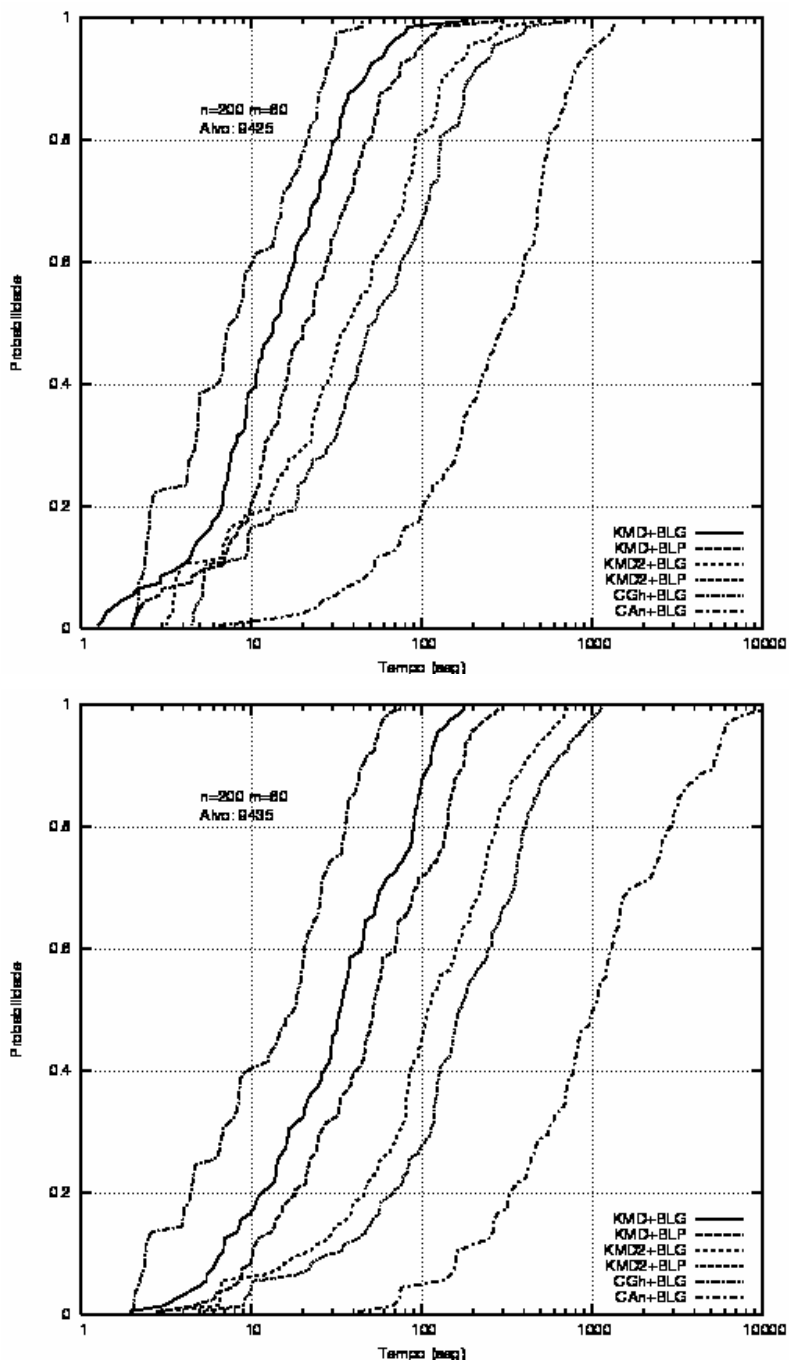

Figura 9 – Gráficos da instância 200C com alvo 1 = 9425 e alvo 2 = 9435.

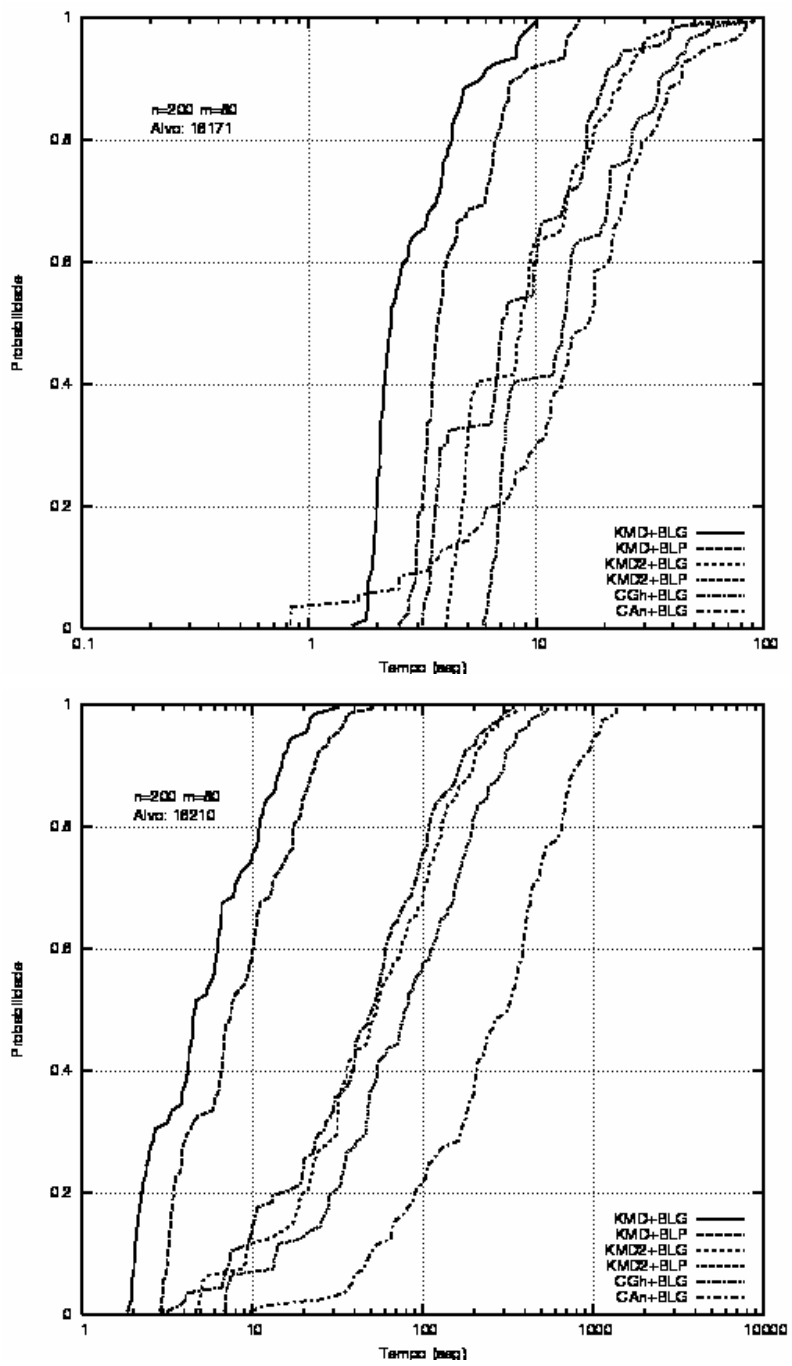

Figura 10 – Gráficos da instância 200D com alvo 1 = 16171 e alvo 2 = 16210.

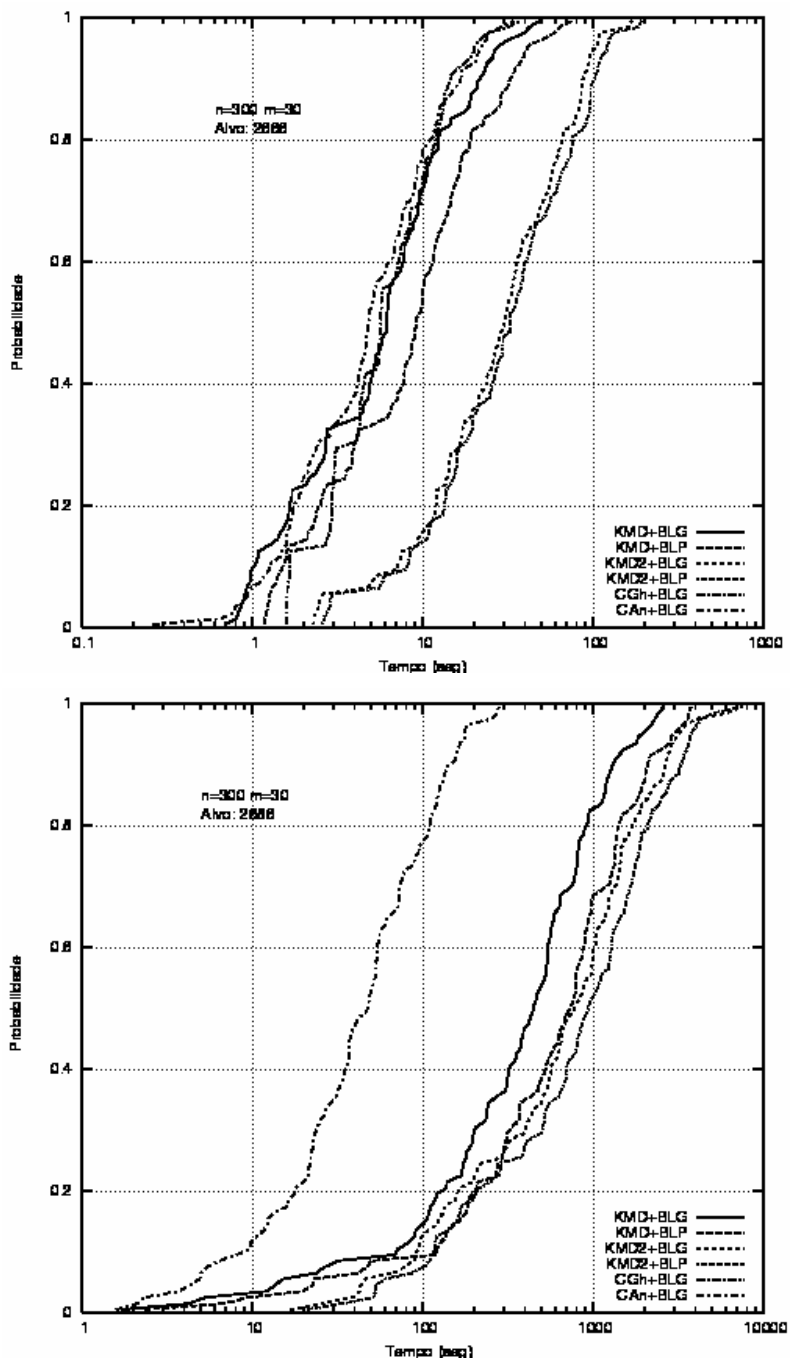

Figura 11 – Gráficos da instância 300A com alvo 1 = 2666 e alvo 2 = 2686.

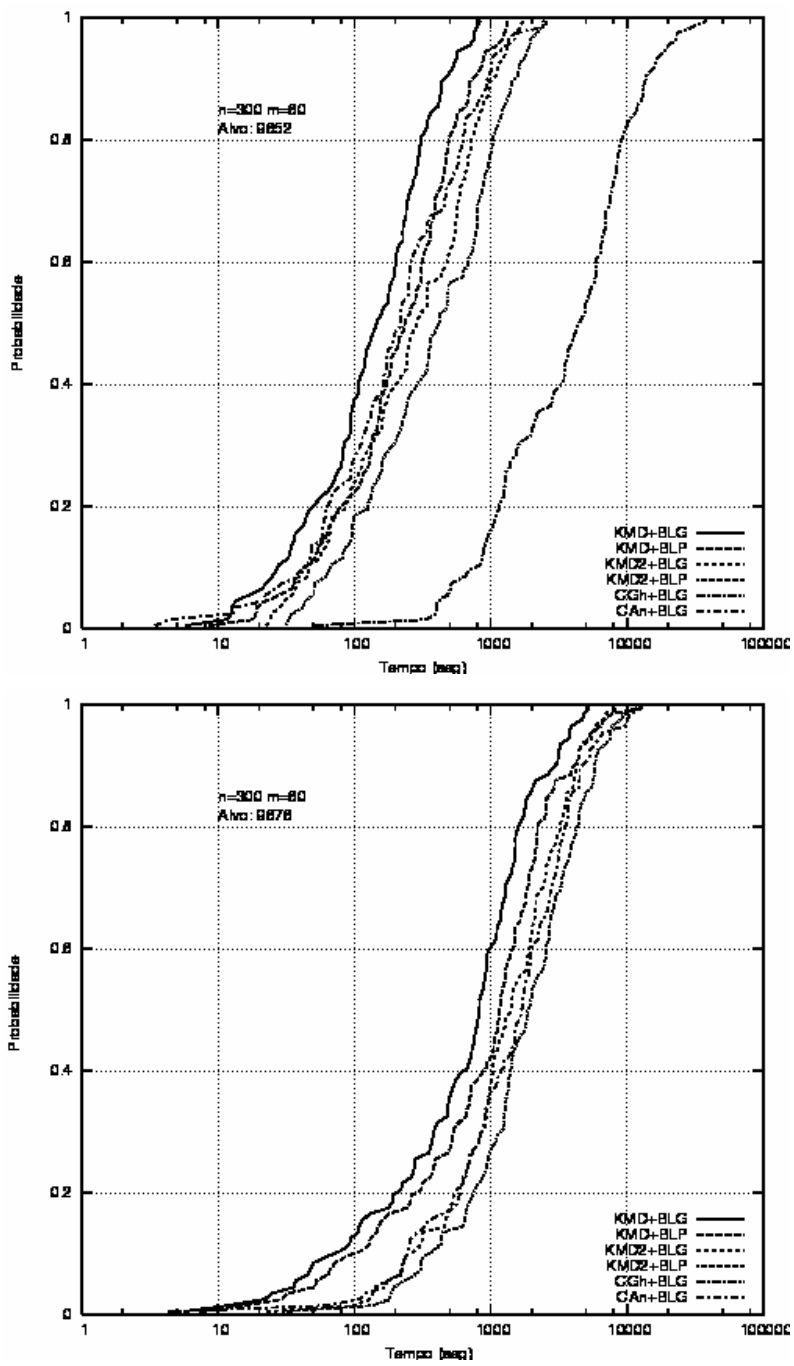

Figura 12 – Gráficos da instância 300B com alvo 1 = 9652 e alvo 2 = 9676.

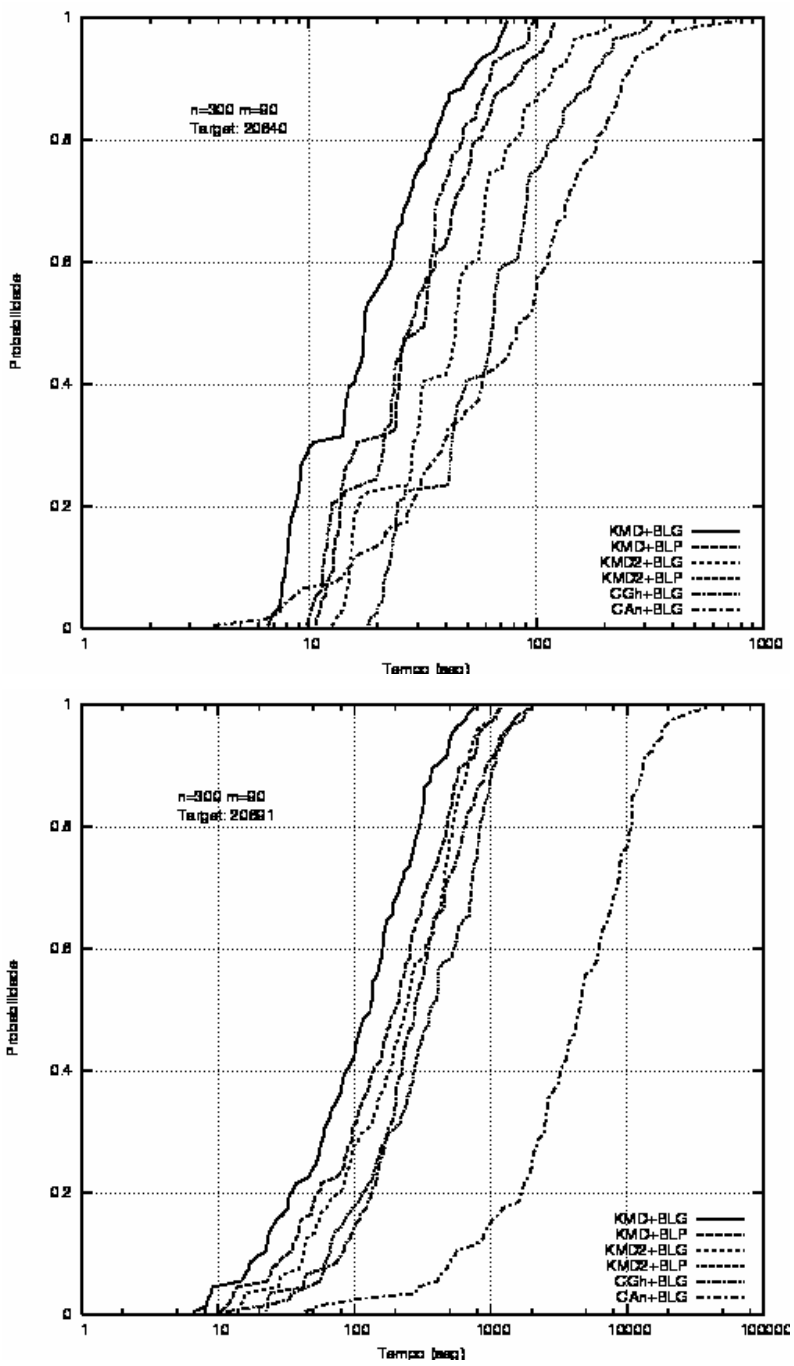

Figura 13 – Gráficos da instância 300C com alvo 1 = 20640 e alvo 2 = 20691.

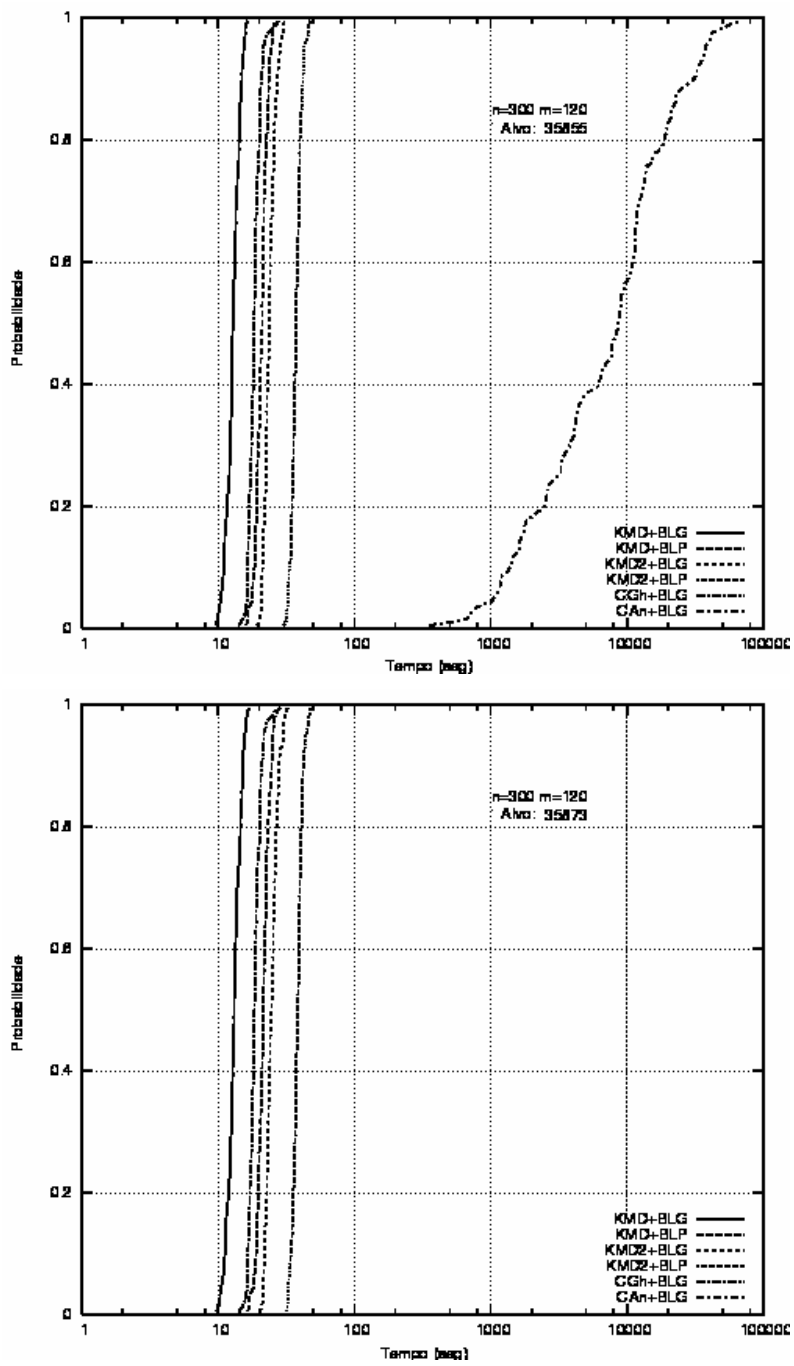

Figura 14 – Gráficos da instância 300D com alvo 1 = 35855 e alvo 2 = 35873.

Uma outra conclusão que podemos tirar a partir das Figuras 6 - 14 é que, avaliando os algoritmos quanto a sua convergência para valores alvos sub-ótimos, os melhores desempenhos foram alcançados com versões que usam na fase de construção o HA-KMD e HA-KMD2. Outra constatação é a de que, em alguns casos, uma busca local mais sofisticada (mais onerosa em termos de tempos computacionais exigidos) pode se tornar mais competitiva se o critério de parada for o alcance de valores alvos pré-definidos.

A Tabela 18 apresenta o tempo de execução (em segundos) necessário para que os algoritmos atinjam determinadas probabilidades (50% e 75%) de convergência para valores alvos mais difíceis. O símbolo (?) significa que o valor alvo 2 somente foi alcançado em um tempo superior a 150 mil segundos ou não foi alcançado. Por exemplo, observando-se a Tabela 18, o algoritmo proposto **KMD+BLG** requer um tempo de 4,23 segundos para atingir com probabilidade empírica de 50%, o valor alvo. Esta análise probabilística mostrou que os algoritmos aqui propostos apresentam uma taxa de convergência empírica em média superior que os da literatura e, além disso, observa-se que todas as propostas sempre convergiram para soluções sub-ótimas num intervalo de tempo razoável, ao contrário dos algoritmos da literatura. Esta característica é extremamente importante para mostrar o quanto os algoritmos aqui propostos são confiáveis, ou seja, sempre atingem a solução alvo dentro de um intervalo de tempo estipulado. Isso mostra um nível de robustez promissor para estas propostas.

Tabela 18 – Tempo (segundos) para que os algoritmos alcance certa probabilidade.

Inst.	**KMD+BLG**		**CGh+BLG**		**CAn+BLG**	
	50%	**75%**	**50%**	**75%**	**50%**	**75%**
100D	**4,23**	**7,37**	13,51	25,09	43,66	94,50
200A	**15,83**	**24,95**	51,70	95,30	6,75	12,45
200B	**33,52**	**58,20**	236,62	4495,05	1276,04	2399,05
200C	33,66	60,00	**18,38**	**25,79**	1040,85	1864,31
200D	**4,67**	**7,99**	51,79	90,16	319,82	447,81
300A	451,18	766,57	**?**	**?**	**47,93**	**74,57**
300B	**826,40**	**1292,93**	**?**	**?**	1707,93	2864,00
300C	**131,05**	**216,71**	275,29	470,82	4511,43	8495,68
300D	**13,13**	**13,90**	18,51	19,32	**?**	**?**

8. Conclusões

O objetivo principal deste trabalho foi o de analisar o impacto de diferentes heurísticas de construção e de busca local no desempenho da metaheurística GRASP para a solução aproximada do Problema da Diversidade Máxima. Para tanto, foram selecionadas algumas heurísticas da literatura e outras foram propostas no sentido de viabilizar as metas programadas.

Nas heurísticas propostas para o GRASP foi apresentada uma maneira de se utilizar o conceito de GRASP reativo. Além disso, fez-se uso de um procedimento de filtro na etapa de construção do GRASP. Estas duas estratégias aliadas permitiram que as heurísticas obtivessem resultados médios de melhor qualidade. Uma extensa bateria de testes foi

realizada com as diferentes heurísticas GRASP propostas neste trabalho e as da literatura. Nestes experimentos, foram geradas instâncias de dimensões bem maiores que as reportadas em [An03], [Gh96] e através destes experimentos foi possível mostrar que as heurísticas propostas promoveram uma melhora na qualidade das soluções encontradas, se comparadas aos da literatura.

Além disso, foi utilizado o *software* GLPK para obter soluções exatas para parte das instâncias testadas e os algoritmos propostos chegaram a soluções ótimas em todas as instâncias onde este é conhecido. Para algumas instâncias, o GLPK teve seu tempo de processamento limitado em 36.000 segundos. Nestes casos, as heurísticas propostas foram capazes de encontrar soluções melhores exigindo um tempo computacional muito menor. Os algoritmos da literatura, no entanto, não foram capazes de chegar ao ótimo em todas as instâncias.

Os gráficos apresentados na análise probabilística demonstram que os algoritmos propostos se portam de maneira semelhante ao longo das instâncias testadas independente do valor alvo. Esta análise também permitiu verificar que as técnicas aqui propostas apresentam na média uma convergência mais rápida para valores sub ótimos quando comparados com versões da literatura, além de se mostrarem mais estáveis.

Dentre os algoritmos propostos ressaltam-se as versões que utilizam as heurísticas de construção **`HA+KMD`** e **`HA+KMD2`**. Além disso, os resultados mostram que **`KMD+BLG`** e **`KMD+BLP`** apresentam bons resultados sendo os mais rápidos dentre os algoritmos propostos enquanto, **`KMD2+BLG`** e **`KMD2+BLP`** produzem melhores resultados, no entanto, demandam um pouco mais de tempo computacional se o critério de parada usado for o de número máximo de iterações GRASP.

Como sugestões para trabalhos futuros podem ser incluídas o uso de outras metaheurísticas (busca tabu, algoritmos evolutivos híbridos com busca local) para o PDM e estudo de variantes do problema.

Referências Bibliográficas

[Ag97] Agrafiotis, D.K. (1997). Stochastic algorithms for maximizing molecular diversity. *Journal Chem. Inf. Comput. Sci*., **37**, 841-851.

[ARR02] Aiex, R.; Resende, M. & Ribeiro, C. (2002). Probability distribution of solution time in GRASP: an experimental investigation. *J. of Heuristics*, **8**, 343-373.

[An03] Andrade, P.M.F.; Plastino, A.; Ochi, L.S. & Martins, S.L. (2003). GRASP for the maximum diversity problem. **In:** *Proc. of the Fifth Metaheuristics International Conference* (MIC) (CD-ROM).

[BMD00] Bhadury, J.; Mighty, E.J. & Damar, H. (2000). Maximing workforce diversity in project teams: a network flow approach. *Omega*, **28**, 143-153.

[CK97] Cutler, M. & Klastorin, T. (1997). The max diversity problem. Technical Report, University of Washington, Department of Management Science.

[DGK94] Dhir, K.; Glover, F. & Kuo, C.-C. (1994). Optimizing diversity for engineering management. **In:** *Proceedings of the IEEE International Engineering Management Conference*.

[FR95] Feo, T. & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**, 109-133.

[FG99] Fluerent, C. & Glover, F. (1999). Improved constructive multicast strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, **11**, 198-204.

[Gh96] Ghosh, J.B. (1996). Computational aspects of maximum diversity problem. *Operations Research Letters*, **19**, 175-181.

[GKD95] Glover, F.; Kuo, C.-C. & Dhir, K. (1995). A discrete optimization model for preserving biological diversity. *Applied Mathematical Modeling*, **19**, 696-701.

[GKD98] Glover, F.; Kuo, C.-C. & Dhir, K. (1998). Heuristic algorithms for the maximum diversity problem. *Journal of Information & Optimization Science*, 109-132.

[GL98] Glover, F. & Laguna, M. (1998). *Tabu Search.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

[HM05] Hansen, P. & Mladenovic, N. (2005). Variable Neighborhood Search. **In:** *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* [edited by E.K. Burk and G. Kendall], Springer, 210-238.

[HRT97] Hassin, R.; Rubinstein, S. & Tamir, A. (1997). Approximation algorithms for maximum dispersion. *Operations Research Letters*, **21**, 133-137.

[KG99] Kochenberger, G. & Glover, F. (1999). Diversity data mining. Technical Report, The University of Mississipi.

[KGD93] Kuo, C.-C., Glover, F. & Dhir, K. (1993). Analysing and modeling the maximum diversity problem by zero-one programming. *Decision Science*, **24**, 1171-1185.

[LM99] Laguna, M. & Martí, R. (1999). GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, **11**, 44-52.

[Ma04] Makhorin, A. (2004). *GNU linear programming kit*.

[Ma99] Maser, C. (1999). Ecological diversity in sustainable development: the vital and forgotten dimension. Saint Lucie Press.

[MRRP00] Martins, S.L.; Resende, M.G.C.; Ribeiro, C.C. & Pardalos, P. (2000). A parallel GRASP for the steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, **17**, 267-283.

[PR97] Prais, M. & Ribeiro, C. (2000). Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, **12**, 164-176.

[PPMR95] Pardalos, P.; Pitsoulis, L.; Mavridou, T. & Resende, M. (1995). Parallel search for combinatorial optimization: genetic algorithms, simulated annealing and GRASP. **In:** *Parallel Algorithms for Irregularly Structured Problems*, Springer Verlag, 317-331.

[Re01] Resende, M. (2001). Greedy randomized adaptive search procedures (GRASP). **In:** *Encyclopedia of Optimization*. Kluwer Academic Publishers, 373-382.

[RR02] Resende, M. & Ribeiro, C. (2002). Greedy randomized adaptive search procedures (GRASP). **In:** *Handbook of Metaheuristics* [edited by F. Glover and G. Kochenberger], Kluwer Academic Publishers, 219-249.

[Sa69] Sammon, J. W. (1969). A non-linear mapping for data structure analysis. *IEEE Trans. Comput*., C-18, 401-409

[SOM04] Silva, G.C.; Ochi, L.S. & Martins, S.L. (2004). Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem. *Lecture Notes on Computer Science*, **3059**, 498-512.

[WL98] Weitz, R. & Lakshminarayanan, S. (1998). An empirical comparison of heuristic methods for creating maximally diverse group. *Journal of the Operational Research Society*, **49**, 635-646.

[Un85] Unkel, W.C. (1985). Natural diversity and national forest. *Natural Areas Journal*, **5**, 8-13.