GRASP PARA O PQA: UM LIMITE DE ACEITAÇÃO PARA SOLUÇÕES INICIAIS

Maria Cristina Rangel[1,2]
Nair Maria Maia de Abreu[2]
Paulo Oswaldo Boaventura-Netto[2]

Resumo

O Problema Quadrático de Alocação (PQA) pertence à classe dos problemas NP-Hard e desafia os pesquisadores tanto em sua teoria quanto em sua parte computacional. Pela sua alta complexidade muitos métodos heurísticos têm sido desenvolvidos para tentar resolvê-lo aproximadamente. A metaheurística GRASP (*greedy randomized adaptive search procedures*) se mostrou bastante eficiente. Neste trabalho, uma proposta para descartar soluções iniciais supostamente ruins é apresentada com base na normalização de custos calculadas num intervalo entre limites de solução. Para este GRASP restrito, foi observada uma redução do tempo computacional para encontrar as soluções ótimas ou soluções viáveis de boa qualidade quando comparado ao GRASP original.

Palavras-chave: GRASP, metaheurísticas, Problema Quadrático de Alocação

Abstract

The Quadratic Assignment Problem (QAP) is an NP-hard problem which has been defying researchers both in theoretical aspects and in what concerns computational results. Owing to its high complexity, several heuristics have been developed trying to approximate its solution. The GRASP metaheuristic (*greedy randomized adaptive search procedures*) shows a good efficiency with it. In this work a proposal to discard initial supposedly bad initial solutionsis based on cost normalization is presented. A reduction of computational time to find optimal or near-optimal solutions was observed with this GRASP, when compared with the original version.

Keywords: GRASP, metaheuristics, Quadratic Assignment Problem.

[1]Departamento de Informática/CT/UFES
Av. Fernado Ferrari, s/n - Campus de Goiabeiras - Vitória - ES - Brasil - CEP 29.060-970
[2]Grupo de Grafos, Combinatória e Aplicações à Pesquisa Operacional
Programa de Engenharia de Produção, COPPE/UFRJ
CP 68.507 - CEP 21.945-970 - Rio de Janeiro - RJ - Brasil
e-mails: {crangel, nair, boaventu}@pep.ufrj.br

1. Introdução

O crescimento exponencial do número de soluções com o tamanho da instância do PQA, torna inviável a procura direta de uma solução de valor ótimo. Nestes casos, métodos heurísticos são empregados para encontrar soluções sub-ótimas de qualidade aceitável. A literatura é rica em estudos dedicados ao PQA, dentre os quais podemos citar dois livros, [PW94] e [Çe98], com extensas referências bibliográficas. Neste trabalho é realizada uma modificação na heurística GRASP aplicada ao Problema Quadrático de Alocação (PQA) desenvolvida por Li, Pardalos e Resende [LPR94] com uma proposta para aceitar ou não a solução inicial gerada na fase de construção evitando uma busca que, eventualmente, exigiria muito esforço computacional. Tal proposta está baseada no cálculo dos custos normalizados em um intervalo de limites das soluções do PQA. No desenvolvimento do tema, o GRASP é apresentado na Seção 2. A Seção 3 discute uma formulação do problema e a normalização dos custos. A Seção 4 consiste na descrição do GRASP aplicado ao PQA, conforme Li et al. [LPR94]. A proposta do trabalho se desenvolve na Seção 5, seguida pela discussão da implementação e resultados dos experimentos aqui realizados. Por fim, apresentam-se as conclusões.

2. O Algoritmo GRASP

Em linhas gerais, o GRASP consiste em um método iterativo probabilístico, onde a cada iteração é obtida uma solução do problema em estudo. Cada iteração GRASP é composta de duas fases: a construtiva, que determina a solução que será submetida à busca local, segunda fase do algoritmo, cujo objetivo é tentar obter alguma melhoria na solução corrente. A seguir, o algoritmo GRASP em pseudo-código é apresentado.

```
Procedimento GRASP( );
1       DadosEntrada( );
2       Enquanto "critério de parada não for satisfeito" faça
3               ConstSolInicGulosaAleatória(sol);
4               BuscaLocal(sol,Viz(sol));
5               AtualizSol(sol,melhorSolEnc);
6       FimEnquanto;
7       Retorna(melhorSolEnc);
Fim GRASP;
```

Na maioria das aplicações, o critério de parada é baseado no número máximo de iterações. Podem-se definir outros critérios, como por exemplo parar quando a solução procurada for encontrada ou estabelecer um tempo máximo de execução.

Na fase de construção, uma solução viável é construída elemento a elemento. Os melhores candidatos a compor a solução são ordenados em uma lista chamada de **lista restrita de candidatos** (LRC). A escolha do próximo elemento é dita **adaptativa,** pois é guiada por uma *função gulosa* que mede, de forma míope, o benefício que o mais recente elemento adiconado à solução concede à parte já construída. O GRASP possui uma componente **probabilística**, em vista da escolha aleatória na lista de candidatos (em um guloso simples seria selecionado o primeiro elemento da lista). Esta técnica de escolha permite que diferentes soluções sejam geradas a cada iteração GRASP. Mostramos, em pseudo-código, a fase de construção da heurística.

```
Procedimento ConstSolInicGulosaAleatória(sol);
1       sol = { };
2       Enquanto "solução não estiver completa" faça
3               ConsLRC(LRC);
4               s   = SelecAleatElem(LRC);
```

```
5           sol = sol ∪ {s};
6           FuncAdapGul(s);
7       FimEnquanto;
Fim ConstSolInicGulosaAleatória;
```

Uma vez obtida uma solução *s*, consulta-se a *estrutura de vizinhança Viz(s)* relativa à essa solução *s*. Uma solução é dita **localmente ótima** se não existir nenhuma solução melhor em *Viz(s)*. As soluções iniciais do GRASP não são necessariamente ótimos locais. Como consequência, faz-se necessária a aplicação de um procedimento de *busca local* para tentar melhorar as soluções advindas da fase construtiva. Esta busca realiza sucessivas trocas da solução corrente, sempre que uma melhor solução é encontrada na vizinhança. Este procedimento termina quando nenhuma solução melhor é encontrada.

Acompanhando o pseudo-código a seguir, pode-se entender a idéia de uma busca local genérica.

```
Procedimento BuscaLocal(sol,Viz(sol));
1       Enquanto "solução não é localmente ótima" faça
2           Encontrar uma melhor solução t ∈ Viz(s);
3           s = t;
4       FimEnquanto;
        Retorna(s como localmente ótima);
Fim BuscaLocal;
```

O procedimento de otimização local pode exigir um tempo exponencial se a busca partir de uma solução inicial qualquer, embora se possa constatar empiricamente a melhoria de seu desempenho de acordo com a qualidade da solução inicial. O tempo gasto pela busca local pode ser diminuído, portanto, através do uso de uma fase de construção que gere uma boa solução inicial. É claro que uma estrutura de dados eficiente e uma implementação cuidadosa são importantes.

Li, Pardalos e Resende [LPR94] submeteram o GRASP a 88 instâncias da biblioteca QAPLIB [BKR97]. Esta metodologia encontrou as melhores soluções até então conhecidas, superando-as em alguns casos. O algoritmo GRASP foi utilizado com diversos problemas, tais como o problema de cobertura [FR95] e do planejamento e escalonamento de produção [LG91], além de problemas de partição em grafos [LFE93] e de localização [Kli92].

A técnica descrita neste trabalho, baseada em limites de aceitação das solucões iniciais no GRASP, pode ser aplicada a qualquer problema de otimização combinatória, desde que sejam conhecidos limites inferiores e superiores para as soluções.

3. O Problema Quadrático de Alocação

Dados o conjunto N = {1,2,...,*n*} e as matrizes (*n* x *n*) simétricas de inteiros não-negativos com diagonais principais nulas F = (f_{ij}) e D = (d_{kl}), o Problema Quadrático de Alocação pode ser estabelecido como

$$\min_{\varphi \in \Pi_n} \sum_{i,j} f_{ij} d_{\varphi(i)\varphi(j)} \ , \qquad (3.1)$$

onde Π_n é o conjunto de todas as permutações dos elementos de N.

Na Teoria da Localização encontramos uma das principais aplicações do PQA. Trata-se de um problema de *lay-out*, onde são dadas n *localidades*, com a respectiva matriz de distância D = (d_{kl}) e n *facilidades*, com sua correspondente matriz de fluxo F = (f_{ij}). O custo de alocar, simultaneamente, as facilidades *i,j* nas localidades *k,l* é o produto $f_{ij}d_{kl}$..O objetivo é encontrar

uma atribuição onde todas as facilidades sejam alocadas à todas as localidades. Deste modo, desejamos determinar uma permutação $\varphi \in \Pi_n$ tal que $\varphi(i) \rightarrow k$ e $\varphi(j) \rightarrow l$ de custo mínimo.

Dentre os problemas de otimização combinatória, o PQA é um dos mais difíceis no que diz respeito à complexidade computacional. Ele pertence a classe de problemas NP-Hard. Por isso, muitos métodos heurísticos tem sido desenvolvidos na tentativa de resolver o PQA, com eficiência e rapidez [BCMP96, CSK93, FF94, QAB97, Sk90, Wi87].

Consideremos a instância de Gavett e Plyter [GP66], dada pelas matrizes

$$F = \begin{bmatrix} 0 & 28 & 25 & 13 \\ 28 & 0 & 15 & 4 \\ 25 & 15 & 0 & 23 \\ 13 & 4 & 23 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 6 & 7 & 2 \\ 6 & 0 & 5 & 6 \\ 7 & 5 & 0 & 1 \\ 2 & 6 & 1 & 0 \end{bmatrix}$$

A figura 3.1 mostra as cliques correspondentes K_F e K_D e a figura 3.2, a atribuição ótima dada pela permutação $\varphi^* = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$ com custo $C(\varphi^*) = 403$. Denotaremos a permutação φ pela imagem sua $\varphi(i) = j$, assim, $\varphi^* = (2\ 4\ 3\ 1)$.

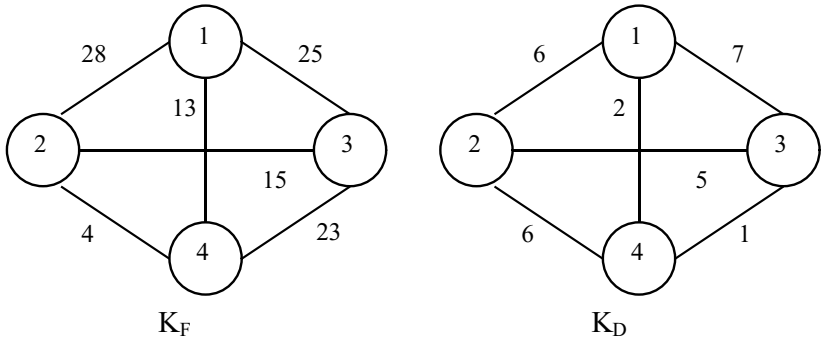

Figura 3.1: Cliques K_F e K_D da instância de Gavett-Plyter

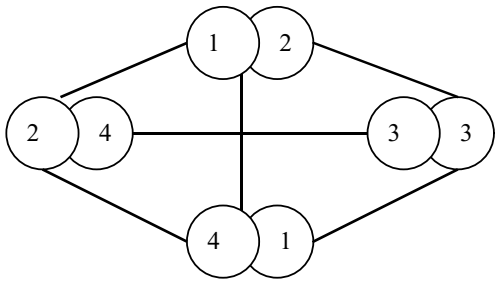

Figura 3.2: Sobreposição ótima das cliques K_F e K_D.

Sejam o conjunto $E = \{(i,j) / 1 \leq i < j \leq n\}$ e $N = C_{n,2}$, a bijeção $\psi_{ij} = n(i-1)-i(i+1)/2+1$, onde ψ_{ij} rotula as arestas das cliques, a partir dos pares de seus vértices, de modo que os vetores definidos como $F^t = (f_z)$ e $D^t = (d_z)$, para $z = 1,...,N$ tem suas coordenadas dispostas segundo a ordem lexicográfica das arestas de K_F e K_D, quando $z = \psi^{-1}(i,j)$, para $(i,j) \in E$.

As coordenadas da matriz quadrada de ordem N,Q = FD^t, contém todos os coeficientes da função objetivo (3.1), para qualquer $\varphi \in \Pi_n$.

O Problema de Alocação Linear (PAL) associado a Q contém todas as soluções viáveis para a instância correspondente PQA. Uma solução $\rho \in \Pi_N$ do PAL (uma permutação das colunas sob as linhas de Q), corresponde a uma atribuição das arestas de K_F sobre K_D, nem sempre é compatível com uma atribuição de vértices de uma clique a vértices da outra. Lawler [La63], Querido, Abreu e Boaventura [QAB97] entendem tal formulação linear como uma relaxação do PQA.

Tomando-se para os vetores $(F^-)^t$ e $(D^+)^t$ as respectivas coordenadas de F^t em ordem não-crescente e das D^t em ordem não-decrescente, definimos a matriz $Q^* = (F^-)(D^+)^t$ e a família QA(Q*), como o conjunto de todas as instâncias *P* do PQA, tendo Q* em comum. O valor de tr(Q*), traço de Q*, é um limite inferior universal (*LimInf*) e a soma das coordenadas da diagonal secundária determina o limite superior universal (*LimSup*) para os custos de todas as instâncias em QA(Q*), [Wi58] e [HLP52].

A instância [GP66] tem Q representada na tabela 3.1 e Q* na tabela 3.2.

$Q=FD^t$	6	7	2	5	6	1
28	168	196	56	140	168	28
25	150	175	50	125	150	25
13	78	91	26	65	78	13
15	90	105	30	75	90	15
4	24	28	8	20	24	4
23	138	161	46	115	138	23

Tabela 3.1: Matriz Q da instância [GP66]

$Q^*=F^+(D)^t$	1	2	5	6	6	7
28	28	56	140	168	168	196
25	25	50	125	150	150	175
23	23	46	115	138	138	161
15	15	30	75	90	90	105
13	13	26	65	78	78	91
4	4	8	20	24	24	28

Tabela 3.2: Matriz Q^*

O custo ótimo da instância PAL de matriz Q é igual a $tr(Q^*)$ que representa *LimInf* = 389 para a instância PQA correspondente, enquanto *LimSup* = 587.

Define-se o *custo normalizado* $|Custo(\varphi)|$ de uma solução φ do PQA por:

$$|Custo(\varphi)| = \frac{Custo(\varphi) - LimInf}{LimSup - LimInf} \qquad (3.2)$$

Como exemplo, consideremos a solução ótima φ* = (2 4 3 1) de Custo(φ*) = 403

e $|Custo(\varphi^*)|$ = 0.0707. Na figura 3.3, podemos verificar que $|Custo(\varphi^*)|$ está bem próximo do *LimInf*. Para comparação, seja uma solução viável φ de Custo(φ) = 479 e custo normalizado $|Custo(\varphi)|$ = 0.4545, o que nos parece estar "relativamente" longe do *LimInf*.

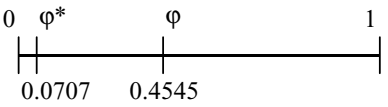

Figura 3.3: Distâncias normalizadas

4. Aplicação do GRASP ao PQA

O algoritmo GRASP é formado essencialmente por quatro componentes básicos, todos utilizados em cada iteração do algoritmo, que constrói uma solução e a submete, como solução inicial, a busca local. Os componentes são: (a) uma função gulosa; (b) uma estratégia de busca adaptativa; (c) um processo probabilístico de seleção de elementos; (d) uma técnica de busca local.

Quando aplicada ao PQA, a fase de construção do GRASP pode ser dividida em duas etapas: a primeira cria uma correspondência entre dois elementos da permutação a ser gerada e a segunda constrói, passo a passo, a correspondência para os *(n - 2)* elementos restantes.

O primeiro par de associações é escolhido a partir de uma lista ordenada de custos resultantes da correspondência entre os vértices do grafo de localidades, com os do grafo de facilidades. A ordenação é guiada por uma função gulosa, que associa facilidades com alto fluxo a localidades próximas. Para determinação dos demais pares, a escolha do próximo elemento de menor custo é feita a partir do último (pertencente à solução) já escolhido. Isto caracteriza uma estratégia adaptativa.

4.1. Fase de Construção

Uma solução viável é construída iterativamente em 2 (dois) estágios. No **estágio 1**, são determinados os primeiros dois pares localidade-facilidade da solução inicial. A lista restrita de candidatos (LRC) é construída, considerando-se para tal, um parâmetro $0<\beta<1$. São ordenadas, em ordem não-decrescente, as $n^2 - n$ entradas da matriz de distâncias D e escolhidas as $\lfloor\beta(n^2-n)\rfloor$ menores. Daí

$$d_{i_1 i_1} \leq d_{i_2 j_2} \leq \ldots \leq d_{i_{\lfloor\beta(n^2-n)\rfloor} j_{\lfloor\beta(n^2-n)\rfloor}}$$

Da mesma forma, as $n^2 - n$ entradas da matriz de fluxos F são ordenadas em ordem não crescente e são escolhidas as $\lfloor\beta(n^2-n)\rfloor$ maiores,

$$f_{kl_{1_1}} \geq f_{kl_{2_2}} \geq \ldots \geq f_{k_{\lfloor\beta(n^2-n)\rfloor} l_{\lfloor\beta(n^2-n)\rfloor}}$$

Finalmente, os produtos das distâncias pelos fluxos são calculados e ordenados em ordem crescente. São considerados os $\lfloor\alpha\beta(n^2-n)\rfloor$ menores. Observe-se que α, $0<\alpha<1$, é o segundo parâmetro utilizado na construção da LRC.

Sendo constantes os dados de entrada, a ordem dos custos na LRC permanece inalterada e todo o processo de ordenação é realizado apenas uma vez.

Em cada iteração do GRASP é selecionado, de forma aleatória, um par localidade-facilidade

dentre os $\lfloor\alpha\beta(n^2-n)\rfloor$ com menores custos, pertencentes à lista e acessar os índices do custo $d_{ij}f_{kl}$ escolhido, correspondentes aos pares de associações {(i,k),(j,l)}. Esses pares constituem parâmetros de entrada para o segundo estágio construção da solução inicial.

Chamaremos de Ω o conjunto de pares localidade-facilidade correspondentes à solução inicial do problema.

O **estágio 2** começa com $|\Omega| = 2$. Nesta etapa, para cada par de associações (i,k) é determinado o valor de c_{ik} dado por:

$$c_{ik} = \sum_{(j,i)\in\Omega} f_{ij}d_{kl} \qquad (4.1)$$

Observe-se que c_{ik} corresponde ao custo da facilidade i com respeito à localidade k, considerando-se as últimas associações feitas. Dentre todas as possibilidades, escolhe-se aquela que minimize o valor do somatório.

Considerando *m* o número de possíveis pares (i,k) a serem ainda escolhidos e o parâmetro α citado anteriormente, a lista restrita de candidatos limita a busca de (i,k) aos $\lfloor\alpha m\rfloor$ pares localidade-facilidade com custos c_{ik} mínimos. Determinado o par (i,k), o conjunto Ω é então atualizado para $\Omega \leftarrow \Omega \cup \{(i,k)\}$.

O algoritmo determina os (n-2) pares localidade-facilidade a partir dos definidos no estágio 1. Para a determinação de cada par, são armazenados em um vetor αm candidatos e uma posição é escolhida aleatoriamente, correspondente ao par que será colocado em Ω, até que toda a solução inicial do problema tenha sido construída. Esta solução é então submetida à segunda fase do GRASP, correspondente à busca local.

O algoritmo que descreve toda a etapa 2 da fase de construção é o seguinte:

```
Procedimento Estágio2 (α, (j1,l1),(j2,l2))
1 Ω= {(j1,l1),(j2,l2)}
2 para associações = 3,...,n
3             m = 0;
4             faça i = 1,...,n
5                   faça k = 1,...n
6                         se (i,k) ∉ Ω → c_ik = Σ_{(i,j)∈Ω} f_ij d_kl
7                                               colocar c_ik no vetor;
8                                               m = m + 1;
9                         fim-se
10                  fim-para
11            fim-para
12 escolher aleatoriamente s no intervalo [1, ⌊αm⌋];
13 determinar o par (i,k) correspondente a posição s do vetor;
14 Ω = Ω ∪ {(i,k)}
15 fim-para
fim Estágio2
```

4.2. Busca Local

A idéia central de um algoritmo de busca local é procurar iterativamente uma solução de melhor custo dentre todas pertencentes à vizinhança da solução corrente. No que diz respeito ao PQA, são propostas na literatura diversas estratégias de busca local.

Dadas duas permutações quaisquer φ_1 e φ_2, a *diferença* entre elas é definida como sendo $\delta(\varphi_1,\varphi_2) = \{i \mid \varphi_1(i) \neq \varphi_2(I)\}$ e a *distância* entre φ_1 e φ_2 é definida por $d(\varphi_1,\varphi_2) = |\ \delta(\varphi_1,\varphi_2)\ |$. A estrutura de vizinhança mais comumente usada chama-se ***k*-troca**. Uma vizinhança *k*-troca para uma permutação $\varphi_1 \in \Pi_n$ é definida por:

$$Viz_k(\varphi_1) = \{\varphi_2 \mid d(\varphi_1,\varphi_2) \leq k\ \}, \text{ onde } 2 \leq k \leq n$$

Neste trabalho foi utilizada a vizinhança 2-troca. No exemplo, para n = 4 se tem $|Viz(\varphi_0)| = C_{4,2} = 6$. Seja φ_0 = (3 1 4 2), temos que $Viz(\varphi_0)$ = { (1 3 4 2), (4 1 3 2), (2 1 4 3), (3 4 1 2), (3 2 4 1), (3 1 2 4)}.

4.2.1. Estratégias de Busca Local

O procedimento de busca local no GRASP desenvolvido em [LPR94] a busca local é iniciado a partir de uma permutação φ_0 gerada aleatoriamente na fase de construção, tal como descrito acima. A cada iteração do GRASP, uma permutação φ_1 de melhor custo é pesquisada na vizinhança da permutação corrente. Se esta existir, φ_0 (corrente) é substituída por φ_1. Vendo essa busca como a construção de uma árvore, podemos percorrê-la explorando sua largura e a profundidade.

Na estratégia de **Busca em Largura e Profundidade**, constrói-se uma árvore por níveis. A partir de φ_0 é construída uma vizinhança com estrutura 2-troca. Dentre as permutações que pertencem ao nível 1, escolhe-se a que fornece menor custo. Se o custo for menor que φ_0, gera-se o nível 2 (permutações geradas por 2-troca sobre a melhor solução do nível 1) e novamente se escolhe a permutação de menor custo. Este processo se repete até que não haja melhora no custo com relação ao nível anterior. A profundidade da árvore é definida pelo custo da permutação (Fig. 4.1). Esta estratégia fornece resultados satisfatórios e tem sido bastante utilizada.

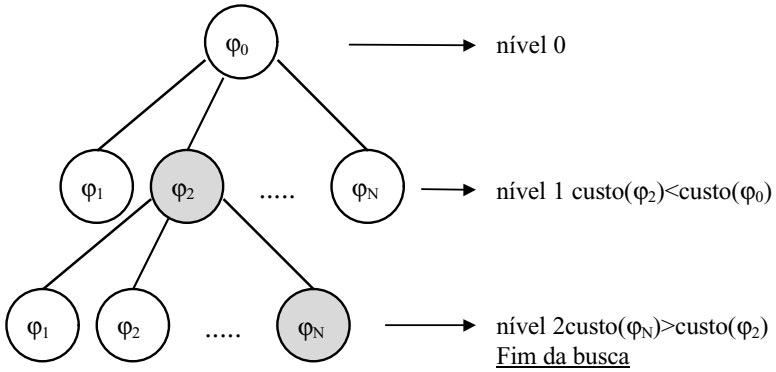

Figura 4.1: Busca em largura e profundidade

5. O GRASP Restrito

Conforme foi dito, um algoritmo de busca local depende da escolha apropriada de uma estrutura de vizinhança, uma técnica eficiente de busca e uma solução inicial de boa qualidade. O procedimento de construção do GRASP descrito na seção 4.1 se preocupa com este último ponto citado. Na primeira fase, escolhe-se aleatoriamente um elemento da LRC (de dimensão $\lfloor \alpha\beta(n^2-n) \rfloor$) de maneira a fornecer as duas primeiras componentes da permutação. Para as (n -

2) componentes restantes, a segunda fase minimiza o custo acumulado das componentes que estão sendo incluídas. Os parâmetros $0 < \alpha, \beta < 1$ são importantes neste processo pois "filtram" as melhores opções de escolha.

O ajuste destes parâmetros não é uma tarefa fácil. A a tendência imediata é de se fixar os valores de α e β bem pequenos para restringir a escolha aos melhores candidatos fazendo com que as soluções geradas sejam de boa qualidade. No entanto, o espaço que tais soluções varrem é pequeno. Relaxando-se os valores desses ,os parâmetros aumenta-se o espaço de busca, mas deteriora-se a média das soluções iniciais. Na tabela 5.1 podemos analisar este ajuste de parâmetros com algumas instâncias da biblioteca QAPLIB. Para melhor comparação, a média dos custos das soluções iniciais (3000 iterações) foi normalizada (seção 3) e os cáculos feitos para $\alpha = \beta = 0.1, 0.25, 0.5, 0.75, 1.0$.

	0.1	**0.25**	**0.5**	**0.75**	**1.0**
Chr12a	0.30	0.41	0.42	0.43	0.49
Chr12b	0.31	0.44	0.45	0.46	0.50
Chr12c	0.26	0.35	0.39	0.42	0.49
Chr15a	0.26	0.35	0.37	0.42	0.50
Chr15b	0.34	0.39	0.39	0.43	0.50
Chr15c	0.28	0.34	0.36	0.42	0.50
Chr18a	0.29	0.34	0.39	0.42	0.49
Chr18b	0.41	0.26	0.29	0.33	0.40
Els19	0.57	0.49	0.45	0.46	0.52
Nug12	0.36	0.40	0.41	0.43	0.46
Nug15	0.39	0.41	0.42	0.44	0.48
Nug20	0.43	0.44	0.44	0.45	0.48
Rou12	0.41	0.43	0.44	0.46	0.49
Rou15	0.42	0.45	0.45	0.48	0.50
Rou20	0.45	0.45	0.46	0.48	0.50
Scr12	0.30	0.38	0.37	0.36	0.40
Scr15	0.37	0.41	0.38	0.40	0.44
Scr20	0.34	0.36	0.37	0.39	0.42

Tabela 5.1: Médias normalizadas das soluções iniciais

Para melhor aproveitamento das soluções geradas por parâmentros mais relaxados, podemos limitar a aplicação do procedimento de busca às soluções mais promissoras advindas da fase de construção. Como exemplo, tomemos a instância Chr12b [BKR97] com $\alpha = \beta = 0.75$, resultando média normalizada de 0.46 e o custo normalizado da solução ótima igual a 0.048. Propomos aceitar as soluções geradas com custo abaixo de um determinado limite, por exemplo $Lim = 0.45$. A figura 5.1 ilustra as soluções aceitas cujo custo normalizado é inferior ou igual a $Lim = 0.45$.

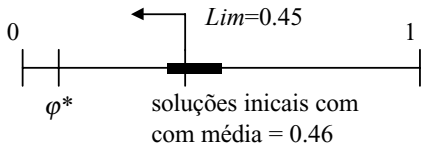

Figura 5.1: Limite de aceitação para soluções iniciais

Com esta poda no espaço das soluções iniciais tem-se um menor tempo de execução do algoritmo pois se descartam várias tentativas de busca que não seriam interessantes, em vista da possibilidade de um longo caminho até o ótimo.

6. Resultados Computacionais e Conclusões

Este trabalho foi implementado em linguagem C e executado em estações de trabalho Digital DEC-3000, com OSF1, 125 Mhz e 32 MRAM. Para comparação dos resultados, foram implementados o GRASP desenvolvido por Li et al [LPR94] e o GRASP Restrito, ambos utilizando a mesma linguagem, os mesmos dados de entrada e a mesma estrutura de dados. Esta restrição foi testada com limites *Lim* = 0.3, 0.45 e 0.5. Ambos os algoritmos terminam quando a solução ótima é encontrada ou quando se atinge o número máximo de iterações, aqui fixado em 3000. Para algumas instâncias, 3000 iterações não foram suficiente para atingir o ótimo (QAPLIB). Para análise de desempenho tais execuções foram desconsideradas.

Os testes foram realizados com um conjunto de 18 instâncias (enumeradas na tabela 5.1), com os parâmetros $\alpha = \beta = 0.1, 0.25, 0.5, 0.75, 1.0$. e *Lim* = 0.3, 0.45 e 0.5 totalizando 360 execuções. O GRASP Restrito apresentou melhor desempemho com $\alpha = \beta = 0.5$ e 0.75 e *Lim* = 0.45 e 0.5. Quando *Lim* = 0.3 o algoritmo descartou muitas soluções, o que prejudicou o seu desempenho, pois houve necessidade de gerar mais soluções iniciais.

Com base nesse estudo utilizando instâncias de dimensão até 20, o algoritmo foi submetido à instâncias de dimensões maiores. Foram executados 32 testes no conjunto de 8 instâncias variando os parâmetros $\alpha = \beta = 0.5$ e 0.75, *Lim* = 0.45 e 0.5 e com número máximo de iterações igual a 500

A tabela 6.1 fornece a média normalizada das soluções iniciais geradas pelos parâmetros $\alpha = \beta = 0.5$ e o número de soluções descartadas pelos limites de aceitação *Lim* = 0.45 e 0.5. Os resultados alcançados considerando $\alpha = \beta = 0.75$ foram semelhantes aos anteriores e estão na tabela 6.2.

	Média normalizada ($\alpha = \beta = 0.5$)	Num. de sol. descartadas (*Lim* = 0.45)	Num. de sol. descartadas (*Lim* = 0.5)
Nug30	0.45	229	21
Kra30a	0.45	276	31
Kra30b	0.45	253	27
Ste36a	0.37	45	7
Ste36b	0.25	1	0
Sko42	0.42	307	2
Sko49	0.46	338	3
Sko64	0.46	408	0

Tabela 6.1: Soluções iniciais geradas com $\alpha = \beta = 0.5$

	Média normalizada ($\alpha = \beta = 0.75$)	Num. de sol. Descartadas (*Lim* =0.45)	Num. de sol. descartadas (*Lim* = 0.5)
Nug30	0.45	278	32
Kra30a	0.46	72	11
Kra30b	0.46	283	30
Ste36a	0.37	33	7
Ste36b	0.24	0	0
Sko42	0.46	353	11
Sko49	0.47	413	0
Sko64	0.46	450	1

Tabela 6.2: Soluções iniciais geradas com $\alpha = \beta = 0.75$

Com relação ao tempo computacional, o algoritmo proposto neste trabalho apresentou um

desempenho melhor. Os gráficos 6.1 e 6.2 mostram a economia do tempo computacional. As instâncias Ste36a e Ste36b, cujas médias normalizadas das soluções iniciais são baixas (tabela 6.1 e 6.2) não apresentaram bons resultados.

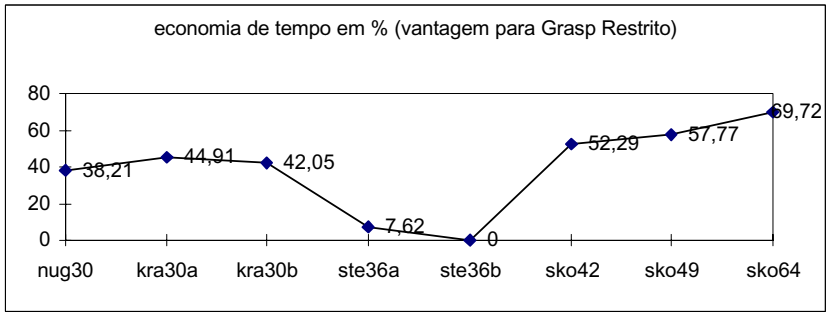

Gráfico 6.1: $\alpha = \beta = 0.5$ e *Lim* = 0.45

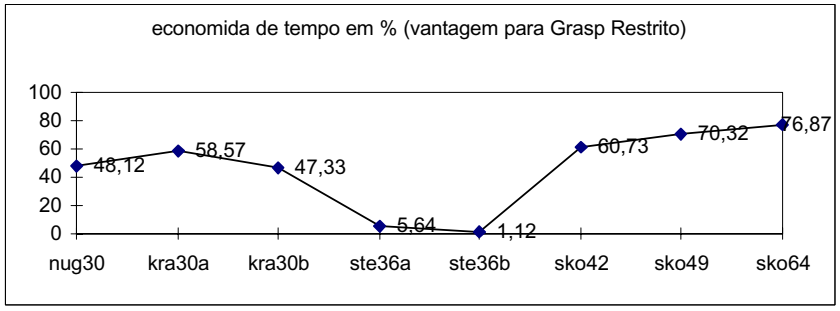

Gráfico 6.2: $\alpha = \beta = 0.75$ e *Lim* = 0.45

A qualidade da solução encontrada pelo GRASP Restrito é tão boa quanto a encontrada pelo GRASP (gráfico 6.3 e 6.4). Em alguns casos houve pequeno aumento da diferença para o valor da melhor solucão viável conhecida, o que é compensado pela significativa economia no tempo computacional.

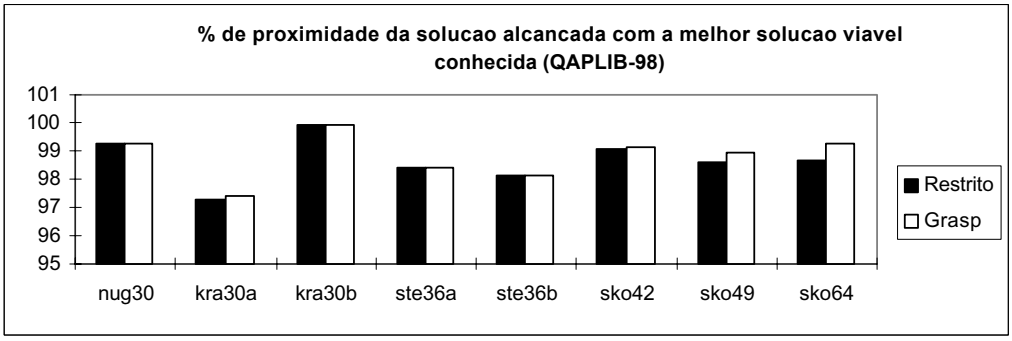

Gráfico 6.3: $\alpha = \beta = 0.5$ e *Lim* = 0.45

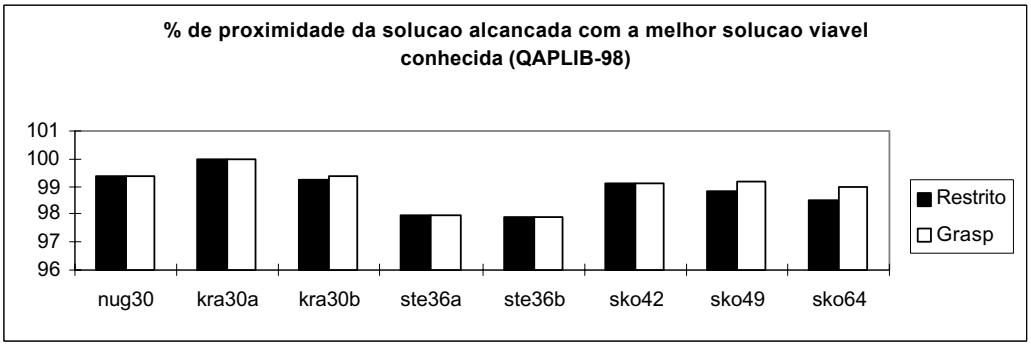

Gráfico 6.4: $\alpha = \beta = 0.75$ e *Lim* = 0.45

Os resultados obtidos nos testes computacionais utilizando o *Lim* = 0.5 não foram vantajosos para o GRASP Restrito devido às médias normalizadas das soluções iniciais . Pode-se observar nas tabelas 6.1 e 6.2 que o número de soluções descartadas é pequeno.

Para melhor escolher o valor do limite, deve-se levar em conta a média normalizada das soluções iniciais geradas pela fase de construção.

Utlizando o algoritmo proposto em instâncias de dimensões n = 100, os resultdados obtidos foram bastante satisfatórios. As instâncias escolhidas foram sko100a-f, disponíveis na QAPLIB. Foram executadas 100 iterações para cada instância. As médias das soluções iniciais estiveram um pouco acima de 0.46 ($\alpha = \beta = 0.5$) e o limite escolhido foi *Lim* = 0.47. Ver tabela 6.3.

	Média com $\alpha = \beta = 0.5$	Num. de sol. descartadas *Lim* = 0.47	Proximidade com a melhor solução viável QAPLIB	Economia de tempo computacional
Sko100a	0.465	26	99.34%	28.50%
Sko100b	0.467	34	98.99%	44.01%
Sko100c	0.465	30	99.82%	29.02%
Sko100d	0.463	21	99.75%	21.77%
Sko100e	0.462	17	98.73%	14.02%
Sko100f	0.465	26	98.74%	29.74%

Tabela 6.3: Resultados com as instâncais sko100a-f com 100 iterações

O uso de *Lim* = 0.47 permitiu economizar o tempo computacional gasto sem qualquer prejuízo à qualidade das soluções alcançadas pelo GRASP Restrito, isto é, a melhor solução viável alcançada em cada instância foi a mesma que o GRASP atingiu. A proximidade com a melhor solução viável conhecida (QAPLIB) foi em média 99%.

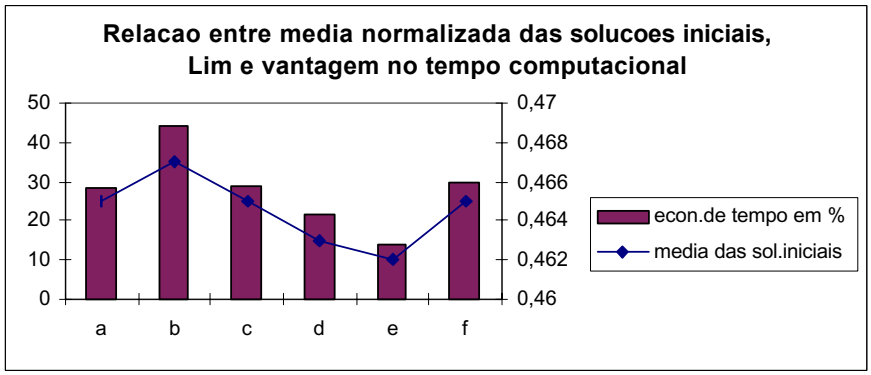

Gráfico 6.5: $\alpha = \beta = 0.5$, *Lim* = 0.47 e instâncias sko100a-f

O gráfico 6.5 mostra que a proximidade da média normalizada das soluções iniciais com *Lim* = 0.47 fornece uma economia diretamente proporcional no tempo computacional

Nota-se que um bom ajuste do limite com a média normalizada das soluções iniciais é um ponto importante para a eficiência do algoritmo proposto. Se fosse escolhido *Lim* = 0.45 não haveria economia alguma e se a escolha fosse *Lim* = 0.5, não se aceitaria nenhuma solução para aplicar a busca local.

Na equação 3.2, os limites usados poderiam ser substituídos por outros mais adequados (ver Li et al [LPRR94], Carraresi e Malucelli [CM94], Gilmore [Gi62], Lawler [La63] e Karisch et al [KÇCE98]). O custo da determinação de melhores limites (todos de ordem igual ou superior a $O(n^3)$), poderia a reduzir a vantagem obtida pelo algoritmo proposto, dada a baixa complexidade da determinação do limite aqui utilizado.

A técnica aqui apresentada pode ser aplicada em outros problemas de otimização combinatória, desde que seja possível a determinação de limites superiores e inferiores para os custos das soluções viáveis.

7. Agradecimentos

Os autores agradecem à CAPES e ao CNPq pelo apoio financeiro à linha de pesquisa que originou este trabalho.

8. Referências

[BCMP96] Bruenegger, Clausen, J., Marzetta, A., and Perregard, M. Joining forces in solving large-scale QAP in parallel, Technical Report DIKU TR-96-23, DCS, U. Copenhagen, Denmark (1996) and it was presented at IPPS 97, the IEEE Int. Parallel Proc. Symp.

[BKR97] Burkard, R.E., Karisch, S.E. e Rendl, F., "*QAPLIB - A Quadratic Assignment Problem Library"*, Journal of Optimization, 10 (1997), 391-403. Internet address: http://opt.math.tu-graz.ac.at/~karisch/qaplib/

[Çe98] Çela, E., "*The Quadratic Assignment Problem - theory and algorithms"*, Kluwer Academic Publishers (1998).

[CM94] Carraresi, P. and Malucelli, F.,"*A reformulation scheme and new lower bound for the QAP* ", DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16 (1994), 147-160.

[CSK93] Chakrapani and Skorin-Kapovi, Massively Parallel Tabu Search for the QAP, Annals of Operation Research 41 (1993), 327-342.

[CP94] Clausen, J. and Perregaard, M. Solving Large QAPs in Parallel, T.R. DIKU TR-94-22, DSC, U. Copenhagen, Denmark (1994). To appear in Comp. Opt. and Apply.

[FR95] Feo, T.A. e Resende, M.G.C., "*Greedy randomized adaptive search procedures*", to appear in Journal of Global Optimization (1995).

[FF94] Fleurent C. and Ferlad, Genetic Hybrids for the QAPs, in Quadratic Assignment and related problems, P.Pardalos and Wolkowicz, eds. DIMACS 16 (1994), 173-187, AMS.

[Gi62] Gilmore, P.C., "*Optimal and suboptimal algorithms for the quadratic assignment problem*", SIAM Journal on Applied Mathematics, 10 (1962), 305-313.

[GP66] Gavett, J.W. e Plyter, N.V., "*The optimal assignment of faciliteis to locations by branch-and-bound"*, Operations Research, 14 (1966), 53-76.

[HLP52] Hardy, G.H., Littlewood, J. E., and Pólya, G., Inequalites, Cambridge Univ. Press, (1952).

[KÇCE98] Karisch, S.E., Çela, E., Clausen, J. e Espersen, T., "*A dual framework for lower bound for the quadratic assignment problem based on linearization*", Technical Report IMM-REP-1998-02, Department of Mathematics Modeling, Technical University of Denmark.

[Kli92] Klincewicz, J., "*Avoiding local optima in p-hub location problem using tabu search and GRASP"*, Annals of Operations Research, 40 (1992), 238-302.

[LPR94] Li,Y., Pardalos, P.M. e Resende, M.G.C., "*A greedy randomized adaptive search procedures for the quadratic assignment problem"*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16 (1994), 237-261.

[La63] Lawer, E.L., "*The quadratic assignment problem*", Management Science, 9 (1963), 586-599.

[LPRR94] Li, Y., Pardalos, P.M., Ramakrishnan, K.G. e Resende, M.G.C., "*Lower bounds for the quadratic assignment problem*", Annals of Operations Research, 50 (1994) - 387-410.

[LG91] Laguna, M e González-Velarde, J., "*A search heuristic for just-in-time scheduling in parallel machines"*, Journal of Intelligent Manufacturing, 2 (1991), 253-260.

[LFE93] Laguna, M., Feo, T.A. e Elrod, H., "*A greedy randomized adaptive search procedures for the 2-partition problem*", tech. report., Graduate School of Business and Administration, The University of Colorado at Boulder, Boulder, CO 80309-0419,(1993).

[PW94] Pardalos, P. e Wolkowicz, H., *Quadratic Assignment and Related Problems*, DIMACS Series no. 16, American Mathematical Society, (1993).

[QAB97] Querido, T.M., Abreu, N.M.M., Boaventura-Netto, P.O., "*RedInv-SA: a simulated annealing for the quadratic assignment problem"*, RAIRO / Oper. Res. 33, 249-273 (1999)

[Sk90] Skorin-Kapov, J., "*Tabu search applied to the quadratic assignment problem*", ORSA Journal on Computing, 2 (1990), n°1, 33-45.

[Wi58] Wimmert, R.J., "*A mathematical model of equipment location"*, J.J.E., (1958), 9, 6, 498-505.

[Wi87] Wilhelm, M.R. e Ward, T.L., "*Solving quadratic assignment problem by simulated annealing*", IEEE Transaction, 19 (1987) n°1, 107-119.