Algoritmo Guloso Adaptativo e Aleatório para o Problema Quadrático de Alocação

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

[1] *Grupo de Grafos, Combinatória e Aplicações à Pesquisa Operacional*
Programa de Engenharia de Produção – COPPE/UFRJ – Rio de Janeiro – Brasil
CP: 68.507 – CEP: 21.945-970
[2] *Departamento de Informática – Centro Tecnológico – UFES*
Av. Fernando Ferrari, s/n – Campus de Goiabeiras – Vitória – ES – Brasil
CEP: 29.060-900
e-mails:{crangel,nair,boaventu,boeres}@pep.ufrj.br

Resumo

O Problema Quadrático de Alocação (PQA) pertence à classe dos problemas NP-hard. Apesar de muitos métodos heurísticos terem sido desenvolvidos visando a resolução de suas instâncias, ainda não é possível encontrar soluções ótimas para instâncias de ordem acima de 25. Uma versão do GRASP, desenvolvida por Li, Pardalos e Resende [LPR94], se mostrou bastante eficiente para o PQA, o que motivou os autores a elaborar uma proposta de modificação na sua busca local, na tentativa de diminuir o número de iterações necessário à obtenção da melhor solução conhecida.

Abstract

The Quadratic Assignment Problem (QAP) is an NP-hard problem. Despite the continuous effort in the formulation of new heuristics, optimal solutions are not generally known yet, for instances of order n > 25. A GRASP-type heuristic developed by Li, Pardalos and Resende [LPR94] showed itself to be very efficient and this fact motivated the authors to propose a perfectioning in its local search reducing the number of iterations.

Palavras Chaves:
Otimização Combinatória, Meta-heurística, Problema Quadrático de Alocação.

Keywords:
Combinatorial Optimization, Metaheuristic, Quadratic Assignment Problem

1. Introdução

Muitos problemas complexos de otimização surgem frequentemente em aplicações na indústria, no governo e na ciência, tais como localização estratégica de reservas de energia, roteamento de veículos, escala de tripulação de linhas aéreas e o projeto eficiente de redes de comunicação. Projetar tais redes é um problema altamente combinatório, tanto pelo porte que apresentam, como pela complexidade dos seus esquemas de interligação.

O crescimento exponencial do número de soluções possíveis, com o tamanho da instância do problema, torna inviável a procura direta de uma solução de valor ótimo. Nestes casos, métodos heurísticos ou aproximativos são empregados para encontrar soluções sub-ótimas (aproximadas) de qualidade aceitável. Algumas dessas técnicas são

mais diretas, preocupadas em achar apenas uma solução, caindo com freqüência em ótimos locais, como os algoritmos gulosos. Outras se destinam a pesquisar com mais cuidado o espaço de soluções, na tentativa de obter resultados mais próximos do ótimo global. Este é o caso da heurística tipo GRASP (***G**reedy **R**andomized **A**daptive **S**earch **P**rocedures)*, desenvolvida por Li, Pardalos e Resende [LPR94], que se mostrou eficiente na abordagem do caso em estudo.

Propomos aqui uma modificação nessa heurística, baseada na expansão do espaço de busca local, utilizando não apenas o custo das soluções, mas também o número de inversões das permutações associadas a essas soluções. Uma formulação do PQA é dada na seção 2. O GRASP é apresentado em detalhe na seção seguinte. A Seção 4 consiste na descrição do GRASP aplicado ao PQA, com as mudanças propostas na busca local. A seguir são discutidas questões de implementação e apresentados os resultados dos experimentos computacionais realizados.

2. O Problema Quadrático de Alocação

O Problema Quadrático de Alocação (PQA) introduzido por Koopmans e Beckmann em 1957 [KB57] visa modelar problemas de *Lay-out* de facilidades, tema importante no contexto da Teoria de Localização. Este problema considera a atribuição de pares de n facilidades, com seus correspondentes fluxos, a pares de n localidades, com distâncias conhecidas, de modo a minimizar o custo total desta instalação.

Muitas aplicações se tornaram referências do PQA, como o problema de *lay-out* de hospitais [El77], o de planejamento de campus universitário [DH72] e o de *lay-out* de placas de circuito impresso [St61]. Outras aplicações também são conhecidas nas áreas de computação paralela [Bo87], de análise estatística de dados [Hu87], de *scheduling* [GG76], de eletrônica [JMRW94], de esportes [Hf77], de arqueologia [GW89] e [KP78] e de mecânica [LM88].

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-Árduo, juntamente com os Problemas do Caixeiro Viajante, do Isomorfismo em Grafos, e do Particionamento, que por sua vez, podem também ser formulados como Problema Quadrático de Alocação. O livro *Annotated Bibliographies in Combinatorial Optimization,* editado por Dell'Amico et al. [DMM97], apresenta uma descrição dos problemas referidos, com suas respectivas formulações e uma vasta bibliografia.

Até o início dos anos 70 o enfoque principal dado a este problema destinava-se ao desenvolvimento de técnicas exatas para a sua resolução *(branch-and-bound)*. Mais tarde, com o surgimento das estratégias meta-heurísticas, o PQA ganhou novo fôlego, sendo apresentado inúmeros trabalhos sobre as mais diferentes abordagens, dentre elas, Simulated Annealing [AQB99], [Wi87] e [MP97], Busca Tabu [Sk90], [Glo89a], [Glo89b] e [Tai91], Algoritmos Genéticos [TS95] e [FF94], GRASP [LPR94] e Colônia de Formigas [MCD94] e [GTD97].

Dados o conjunto de índices $\{1,2,...,n\}$ e as matrizes quadradas de ordem n, $F = (f_{ij})$ e $D = (d_{kl})$, o Problema Quadrático de Alocação pode ser estabelecido como

$$\min_{\varphi \in \Pi_n} \sum_{i,j} f_{ij} d_{\varphi(i)\varphi(j)}, \qquad (2.1)$$

onde Π_n é o conjunto de todas as permutações de n. 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 a todas as localidades.

Deste modo, desejamos determinar uma permutação $\varphi \in \Pi_n$ tal que $\varphi(\iota) \rightarrow k$ e $\varphi(\varphi) \rightarrow l$, cujo custo total seja mínimo. No texto notaremos $\varphi = (\varphi(1)... \varphi(i)... \varphi(j)... \varphi(n))$.

Para a instância de Gavett e Plyter (GP66), dada pelas matrizes

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

A figura 2.1 mostra a atribuição correspondente à permutação $\varphi = (2\ 3\ 4\ 1)$, de custo C = 457.

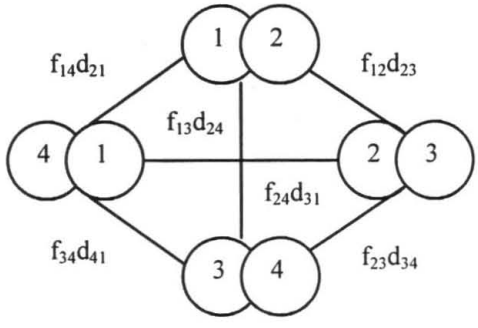


Figura 2.1: Uma solução viável para a instância (GP66).

Uma instância PQA pode ser representada por um par de grafos completos K_F e K_D, de arestas valoradas pelas coordenadas de F e de D respectivamente. O objetivo é achar uma sobreposição de um grafo sobre o outro, que corresponda a uma alocação de custo ótimo ao se somarem os produtos dos valores das arestas sobrepostas. Não havendo restrições, o menor custo seria obtido pela atribuição da aresta de maior valor em K_F à aresta de menor valor em K_D (ou reciprocamente). A dificuldade reside justamente em que nem sempre uma atribuição assim feita corresponde a alguma sobreposição dos vértices de um grafo sobre os do outro.

Consideremos o conjunto $\mathbf{N} = \{ij \mid 1 \leq i < j \leq n\}$, de elementos em ordem lexicográfica e tomemos $f_{ij} = f_r$ e $d_{kl} = d_s$, onde ij e kl são respectivamente os r- e s- ésimos elementos em **N**. Os valores de r e s podem ser determinados pela função ψ abaixo:

$$\psi(i,j) = r, \text{ para } r = (i - 1)n - i(i + 1)/2 + j \quad (2.2)$$

A matriz quadrada $Q = FD^T$, de ordem $N = |\mathbf{N}|$ e de coordenadas iguais aos coeficientes da função objetivo (2.1), para qualquer $\varphi \in \Pi_n$, define um Problema de Alocação Linear (PAL) contendo todas as soluções viáveis para a instância PQA correspondente.

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 o conjunto $QA(Q^*)$, de todas as instâncias P do PQA, tendo Q^* em comum. Seu traço $tr(Q^*)$ é um limite inferior para toda $P \in QA(Q^*)$ e a soma das coordenadas da sua diagonal secundária é um limite superior ([HLP52], [Wi58] e [Ab84]). As Tabelas 2.1 e 2.2 representam Q e Q^* para a instância de Gavett e Plyter [GP66].

	(i,j)	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
Q	$\psi_{i,j}$	1	2	3	4	5	6
(i,j)	$\psi_{i,j}$	F\D	6	7	2	5	6	1
(1,2)	1	28	168	196	**56**	140	168	28
(1,3)	2	25	150	175	50	125	150	**25**
(1,4)	3	13	78	91	26	65	**78**	13
(2,3)	4	15	90	**105**	30	75	90	15
(2,4)	5	4	**24**	28	8	20	24	4
(3,4)	6	23	138	161	46	**115**	138	23

Tabela 2.1: Matriz Q da instância (GP66) - Coordenadas de z^*_ρ

	Q*	(i,j)	(3,4)	(1,4)	(2,3)	(2,4)	(1,2)	(1,3)
		$\phi^{-1}{}_D$	6	3	4	5	1	2
(i,j)	$\phi^{-1}{}_F$	F*\D*	1	2	5	6	6	7
(1,2)	1	28	28	56	140	168	168	194
(1,3)	2	25	25	50	125	150	150	175
(3,4)	6	23	23	46	115	138	138	161
(2,3)	4	15	15	30	75	90	90	105
(1,4)	3	13	13	26	65	78	78	259
(2,4)	5	4	4	8	20	24	24	28

Tabela 2.2: Matriz Q*

São definidas as funções ϕ_F e ϕ_D, com o objetivo de mapear as trocas feitas nas posições dos vetores F^t e D^t, na obtenção dos respectivos $(F^*)^t$ e $(D^*)^t$. Para a instância apresentada, temos:

ϕ_F:	F	$\Rightarrow$	F*	e	ϕ_D:	D	$\Rightarrow$	D*
	1	$\rightarrow$	1			1	$\rightarrow$	5
	2	$\rightarrow$	2			2	$\rightarrow$	6
	3	$\rightarrow$	5			3	$\rightarrow$	2
	4	$\rightarrow$	4			4	$\rightarrow$	3
	5	$\rightarrow$	6			5	$\rightarrow$	4
	6	$\rightarrow$	3			6	$\rightarrow$	1.

O valor do custo ótimo do PAL correspondente é igual a tr(Q*) = 389, cuja atribuição das arestas, dada pela permutação identidade $I_6 \in \Pi_6$, não determina uma solução viável para o GP66.

A permutação ρ = (2 1 3 6 4 5) troca as colunas em Q* e é viável para esta instância. Sua correspondente permutação de vértices é φ^* = (4 1 3 2) resulta no custo ótimo de valor z^*_ρ = 403.

Dada uma permutação ρ em Q*, a permutação ξ_ρ em Q pode ser obtida através de

$$\xi_\rho = \phi_F \circ \rho \circ \phi^{-1}{}_D \qquad (2.3)$$

No exemplo, ξ_ρ = (3 6 5 2 1 4) é uma permutação ótima, cujas parcelas estão em destaque em negrito na Tabela 2.1.

Dada uma instância P $\in$ QA(Q*), para verificar se uma permutação $\rho \in \Pi_N$ é viável determina-se uma outra de vértices $\varphi \in \Pi_n$.

$$\varphi = (\varphi(1)\ \varphi(2)\ \varphi(3) \ldots\ \varphi(n)),$$

tal que

$$\rho = (\psi(\varphi(1),\varphi(2))\ \psi(\varphi(1),\varphi(3))\ \ldots\ \psi(\varphi(n-1),\varphi(n)))$$

Percorrendo o caminho inverso com as funções $\phi^{-1}{}_F$, $\phi^{-1}{}_D$ e ψ^{-1}, temos

$$f_r d_{\rho(r)} \rightarrow f_{\phi_F^{-1}(r)} d_{\phi_D^{-1}(\rho(r))} \rightarrow f_p d_q \rightarrow f_{\psi^{-1}(\varphi)} d_{\psi^{-1}(\varphi)} \rightarrow f_{ij} d_{kl} \qquad (2.4)$$

para $\phi^{-1}{}_F(r) = p$; $\phi^{-1}{}_D(\rho(r)) = q$; $\varphi(i) = k$ e $\varphi(j) = l$, $1 \le r, p, q \le N$ e $1 \le i, j, k, l \le n$.

O custo de ρ em Q* é igual a

$$Z(\rho) = \sum f_i d_{\rho(i)} \qquad (2.5)$$

e seu acréscimo em relação ao limite inferior z_I = tr(Q*), é

$$\Delta(\rho) = Z(\rho) - Z_I \qquad (2.6)$$

Um par de permutações $(\rho_1,\rho_2) \in \Pi_N \times \Pi_N$ pode ter seus custos comparáveis, independentemente dos valores dos fluxos e distâncias da instância considerada. Neste caso, ou $\Delta(\rho_1) \le \Delta(\rho_2)$ ou $\Delta(\rho_1) \ge \Delta(\rho_2)$ e se diz que ρ_1 e ρ_2 são livremente comparáveis. A comparabilidade livre desses acréscimos é equivalente a dos produtos escalares definidos por $\langle F,\rho(D)\rangle$, para $\rho \in \Pi_N$ e sua discussão pode ser encontrada em [Ab84] e [AQB99].

Uma inversão em uma permutação ρ é um par ordenado de posições (i,j), i,j $\in$ {1,2,...,N} tal

que $i < j$ e $\rho(i) > \rho(j)$. Seja $\Im(\rho)$ o conjunto das inversões em ρ. Sua cardinadinalidade $|\Im(\rho)|$ é o número de inversões de ρ.

Lema 2.1:

Se $\forall$ k, $1 \leq k \leq N$, k≠i, $i+1$, $\rho_1(k) = \rho_2(k)$;

$\rho_1(i) = \rho_2(i+1)$;

$\rho_1(i+1) = \rho_2(i)$ e $|\Im(\rho_1)| < |\Im(\rho_2)|$

então ρ_1 e ρ_2 são livremente comparáveis e $Z(\rho_1) \leq Z(\rho_2)$.

Prova: Das hipóteses do lema e de (2.5) e (2.6) segue-se,

$$\Delta(\rho_1) - \Delta(\rho_2) = (f_i - f_{i+1}) (d_{\rho_1(i)} - d_{\rho_2(i)}).$$

Dada a ordenação não-crescente das coordenadas de F*, o primeiro fator é não-negativo. Como o número de inversões de ρ_1 é menor que o de ρ_2, temos $\rho_2(i+1) < \rho_1(i+1)$. Dada a ordenação não-decrescente das coordenadas de D*, o segundo fator é não-positivo, acarretando

$$\Delta(\rho_1) \leq \Delta(\rho_2) \Rightarrow Z(\rho_1) \leq Z(\rho_2)$$

Exemplo:

Sejam $\rho_1 = (2\ 1\ 3\ 5\ 4\ 6)$ e $\rho_2 = (2\ 3\ 1\ 5\ 4\ 6)$.

Temos $|\Im(\rho_1)| = 2$ e $|\Im(\rho_2)| = 3$;

então $\Delta(\rho_1) - \Delta(\rho_2) = (f_2 - f_3)(d_1 - d_3) \Rightarrow \Delta(\rho_1) \leq \Delta(\rho_2)$.

Combinando as idéias acima e atendidas as hipóteses do Lema 2.1 concluímos que pares de permutações livremente comparáveis tem número de inversões crescente, com os custos das soluções associadas. Com esta informação nos foi possível ampliar o espaço de busca do algoritmo GRASP desenvolvido em [LPR94]. As modificações propostas estão descritas na seção 5.

3. 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: **construção**, que determina a solução que será submetida segunda fase do algoritmo, a **busca local**, cujo objetivo é obter alguma melhoria na solução inicial construída. O critério de parada definido, na maioria das aplicações, consiste em estipular o número máximo de iterações. Pode-se definir outros critérios, por exemplo: parar quando a solução procurada for encontrada, ou determinar o tempo de execução do algoritmo. O algoritmo GRASP em pseudo-código é apresentado abaixo.

```
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 fase de construção, uma solução viável é construída iterativamente (um elemento da solução de cada vez). Os candidatos que compõem a solução são ordenados em uma lista (onde são sempre armazenados os melhores). A escolha do próximo elemento que será adicionado à solução é guiada por uma *função gulosa* que mede o benefício que o mais recente elemento escolhido concede à parte da solução já construída.

O GRASP possui uma componente **probabilística**, em vista da escolha aleatória de um elemento da lista de candidatos. A lista dos melhores candidatos é chamada de **lista restrita de candidatos** (LRC). Esta técnica de escolha

permite que diferentes soluções sejam geradas a cada iteração do algoritmo. A fase de construção é descrita abaixo.

```
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;
```

Pode-se observar que a linha 6 mede o efeito da seleção de s sobre a solução que está sendo construída, caracterizando a componente **adaptativa** do algoritmo GRASP *(greedy randomized adaptive search procedures).*

As soluções iniciais geradas nesta fase não são em geral ótimos locais, tornando-se necessário a realização de uma busca local para melhorar tais soluções. Esta busca trabalha de modo iterativo, realizando-se sucessivas trocas da solução corrente, por uma melhor solução encontrada na sua vizinhança. Este procedimento termina quando nenhuma solução melhor é encontrada.

A estrutura de vizinhança *Viz(s)* relativa à solução s do problema é um subconjunto *Viz(s)* de soluções do problema. Uma solução é dita localmente ótima, se não existir nenhuma solução melhor em *Viz(s).*

Uma busca local genérica pode ser entendida observando-se o procedimento abaixo. Se considerados alguns fatores tais como escolha apropriada de uma estrutura de vizinhança, técnica de busca eficiente e uma boa solução inicial, é possível obter-se maior eficiência na determinação de ótimos locais de boa qualidade.

```
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;
```

Uma vantagem inerente ao GRASP é a sua facilidade de implementação, inclusive paralela. O tamanho da lista LRC e o número de iterações são parâmetros que devem ser definidos e ajustados.

4. O Algoritmo GRASP aplicado ao PQA

A fase de construção do GRASP no PQA 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 já escolhido. Para a obtenção de uma solução viável, são determinados os primeiros dois pares localidade-facilidade da solução inicial. Uma lista restrita de candidatos (LRC) é construída, considerando-se para tal um parâmetro β, $0 < \beta < 1$. Seja $\lfloor x \rfloor$ o maior inteiro menor ou igual a x. São ordenadas, em ordem crescente, as $(n^2 - n)$ entradas da matriz de distâncias D e escolhidas as $\lfloor \beta(n^2-n) \rfloor$ menores.

Da mesma forma, as $(n^2 - n)$ entradas da matriz de fluxos F são ordenadas em ordem não

crescente e escolhidas as $\lfloor\beta(n^2-n)\rfloor$ maiores. Os produtos das distâncias, pelos fluxos são calculados e dispostos em ordem crescente, considerando-se 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.

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, que é processado em tempo linear [LPR94]. Esses pares constituem parâmetros de entrada para a segunda etapa de construção da solução inicial.

Chamaremos de Ω o conjunto de pares localidade-facilidade correspondentes à solução inicial do problema e inicializamos com $|\Omega| = 2$. A cada par de associações (i,k) é determinado o valor de c_{ik} dado por:

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

Observe-se que c_{ik} corresponde ao custo da facilidade i com respeito à localidade k, considerando-se as últimas associações feitas.

Considerando-se o parâmetro a e sendo m o número de possíveis pares (i,k) a serem ainda escolhidos, a lista restrita de candidatos limita a busca aleatória de um par (i,k) aos $\lfloor\alpha m\rfloor$ pares de custos c_{ik} com ordenação crescente. O conjunto Ω é então atualizado para $\Omega \leftarrow \Omega \cup \{(i,k)\}$. Esta etapa se repete (n-2) vezes, até que toda a solução inicial do problema tenha sido construída. Pode-se garantir, pela maneira como c_{ik} é determinado, que a solução gerada é viável. Esta solução é então submetida à busca local, segunda fase do GRASP.

A definição de uma boa estratégia de busca local depende de uma escolha adequada da estrutura de vizinhança, de técnicas de busca eficientes e da qualidade da solução inicial. A fase de construção do GRASP é bastante relevante para obtenção de uma solução inicial de qualidade. Para a construção da vizinhança usada por Li, Pardalos e Resende [LPR94] é necessário definir a *diferença* entre duas permutações,

$$\delta(\varphi_1,\varphi_2) = \{i \mid \varphi_1(i) \neq \varphi_2(i)\} \qquad (4.2)$$

e a *distância* entre elas,

$$d(\varphi_1,\varphi_2) = |\, \delta(\varphi_1,\varphi_2)\,|. \qquad (4.3)$$

A estrutura de vizinhança conhecida como *vizinhança k-trocas* é então definida para $\varphi_1 \in \Pi_n$ como

$$Viz_k(\varphi_1) = \{\varphi_2 \mid d(\varphi_1,\varphi_2) \leq k\}, \text{ onde } 2 \leq k \leq n. \qquad (4.4)$$

No GRASP a busca local é iniciada a partir de uma permutação φ_0 gerada aleatoriamente na fase de construção. A cada iteração, uma permutação φ_1 de melhor custo é pesquisada na *vizinhança 2-trocas* da permutação corrente. Se esta existir, φ_0 é substituída por φ_1. Vendo essa busca como a construção de uma árvore, podemos percorrê-la explorando sua largura e a profundidade. A largura corresponde a $|\, Viz_k(\varphi)|$ e a profundidade é determinada pelo custo de φ (Fig. 4.1).

5. Busca Local Modificada

Este trabalho propõe uma alteração na construção da árvore visando aumentar o espaço de busca local e, consequentemente, sua eficiência. Atendidas as hipóteses do Lema 3.1, o custo da permutação cresce, com o número de inversões em permutações livremente comparáveis. Portanto, esta informação foi introduzida no algoritmo de busca.

Na estratégia anterior, a cada nível *i (i≥ 1)* a escolha cai apenas na permutação de menor custo. Pela alteração proposta, escolhe-se também a permutação que possui o menor número de

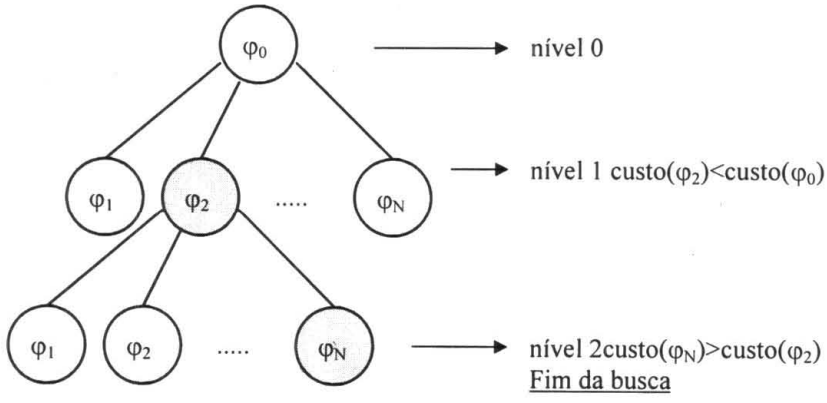


Figura 4.1: Busca em largura e profundidade

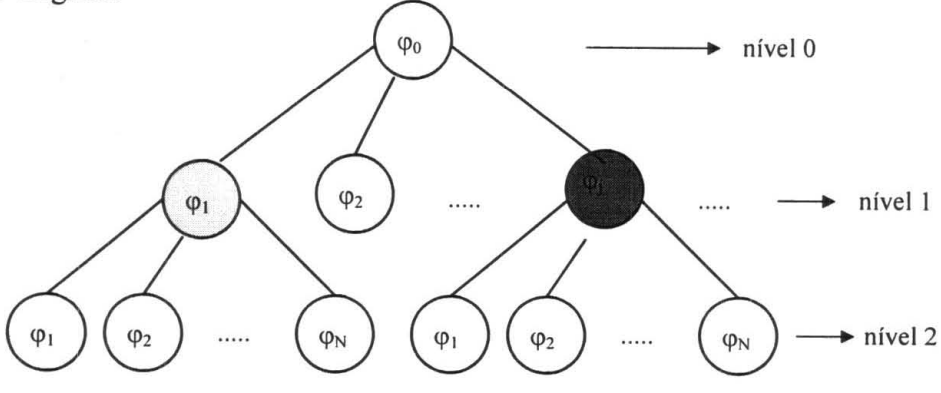
inversões, gerando as respectivas vizinhanças pela estrutura 2-trocas, nos dois nós (Fig. 5.1).

Com relação à profundidade da árvore, foi feita uma alteração no critério de parada. Na busca anterior, apenas o custo era levado em consideração. Nesta, a busca local termina quando não houver mais chances de melhorar também o número de inversões.

A cardinalidade do nível 1 é $C_{n,2}$ (Fig. 5.1). A partir do nível 2, o número de permutações pode duplicar para atender ao duplo critério. Estas permutações podem ser a mesma, diminuindo o custo operacional. Na modificação proposta, no pior caso, cada iteração gera o dobro de nós gerados no Grasp original.

6. Resultados Computacionais

Este trabalho foi implementado em linguagem C e executado em estações de trabalho Digital DEC-3000, OSF1, 125 Mhz e 32 MRAM. Para comparação dos resultados, foram implementados o GRASP desenvolvido por Li et al. [LPR94] e o Grasp Ampliado, utilizando a mesma estrutura e leitura dos dados, levando-se em consideração os tipos diferentes de busca. Ambos os algoritmos terminam quando a solução desejada é encontrada ou quando é alcançado um número máximo de iterações.



($\varphi_1 \rightarrow$ menor custo e $\varphi_j \rightarrow$ menor número de inversões)

Figura 5.1: Nova árvore de busca local

A tabela 6.1 mostra os resultados na seguinte ordem: solução ótima alcançada pelo Grasp Ampliado, número de iterações necessárias e tempo (em segundos). Na sequência, a ordem de apresentação se repete para o GRASP original. As instâncias estão disponíveis na QAPLIB, organizada por Burkard et al. [BKR97]. As instâncias relacionadas na tabela 6.1 possuem ótimos conhecidos (QAPLIB - [BKR97]) e ambos os algoritmos foram capazes de encontrá-los.

Instâncias	QAPLIB	GRASP	Ampliado	GRASP	Original
	ótimo	iteração	tempo(seg)	iteração	tempo(seg)
Chr12a	9552	301	214	326	32
Chr12b	9742	10	7	5	1
Chr12c	11156	39	26	149	14
Nug12	578	60	89	60	5
Rou12	235528	5	1	62	6
Scr12	31410	7	2	4	1
Chr15a	9896	87	256	261	80
Chr15b	7990	131	369	285	99
Chr15c	9504	470	1380	389	117
Nug15	1150	42	100	81	25
Rou15	354210	89	100	299	90
Scr15	51140	5	10	45	15
Chr18a	11098	1418	13857	11489	8578
Chr18b	1534	79	838	107	76
Els19	17212548	3	41	5	6
Nug20	2570	89	1333	277	401
Rou20	725522	11413	53898	13603	17621
Scr20	110030	370	2766	3877	5725

Tabela 6.1: Instâncias onde são conhecidos os ótimos

A tabela 6.2 faz uma comparação com instâncias de dimensão 30. Estas não são conhecidos os ótimos. O GRASP Ampliado encontrou solução melhores ou iguais ao GRASP desenvolvido por Li et al.[LPR94] porém, o tempo computacional é bem maior. Nesses testes foi estabelecido um número máximo de 500 iteração e o tempo apresentado é o gasto para executá-lo. A iteração nesta tabela mostra em qual iteração foi atingida a melhor solução viável.

Pode-se concluir que o GRASP Ampliado é capaz de atingir soluções tão boas quanto o GRASP Original com um número de iterações bem menor. A útima coluna da tabela 6.3 mostra relação, em porcentagem, de quanto a solução encontrada pelo GRASP Ampliado ficou acima do GRASP Original.

Instâncias	QAPLIB	GRASP Ampliado			GRASP Original		
	melhor viável	solução	iteração	tempo(seg)	solução	iteração	tempo(seg)
Nug30	6124	6156	432	99697.93	6170	59	1980.98
Kra30a	88900	90200	152	99481.18	91290	90	1807.93
Kra30b	91420	91490	446	99719.40	91490	446	1822.53

Tabela 6.2: Instâncias n = 30 onde não são conhecidos os ótimos

Para a comparação de instâncias maiores, o GRASP Ampliado executou o número de iterações possíveis dentro de um limite de tempo. Tal limite foi estabelecido pelo tempo gasto pelo GRASP Original para a execução de 500 iterações.

7. Conclusão

Neste trabalho foi discutida uma proposta de expansão do espaço na busca local do algoritmo GRASP desenvolvido por Li, Pardalos e Resende [LPR94]. Foi possível obter a solução ótima de

Instâncias	QAPLIB melhor solução	GRASP Ampliado	% acima da QAPLIB	GRASP Original	% acima da QAPLIB	Comparação Orig/Ampl
Ste36a	9526	9894	3.9%	9682	1.6%	2.2%
Ste36b	15852	17536	10.6%	16154	1.9%	8.6%
Sko42	15812	16202	2.5%	15946	0.8%	1.6%
Sko49	23386	23800	1.8%	23630	1.0%	0.7%
Sko64	48498	49604	2.3%	48860	0.7%	1.5%

Tabela 6.3: Instâncias maiores onde não são conhecidos os ótimos

instância do QAPLIB em um número menor de iterações. A busca nesta nova vizinhança é mais custosa pois, no pior caso, explora-se dois nós em cada nível da árvore (seção 5) duplicando o número de soluções geradas a cada nível. Entretanto, foi possível obter o ótimo (tabela 6.1) em um número 2.14 vezes menor de iterações, em média, que o GRASP Original. Com respeito ao tempo computacional, o GRASP é 2.29 vezes mais rápido, em média. As tabelas 6.2 e 6.3 mostram que é necessário diminuir o tempo computacional do GRASP Ampliado. Uma sugestão é a implementação paralela onde um número de iterações poderia ser definido para cada processador e apenas uma variável, que armazenará a melhor solução é compartilhada pelos processadores.

8. Referências Bibliográficas

[Ab84] Abreu, N.M.M., *"Um estudo algébrico e combinatório do problema quadrático de alocação segundo Koopmans e Beckmann"*, (1984), Tese de Doutorado, COPPE/UFRJ.

[AQB99] Abreu, N.M.M , Querido, T.M., e Boaventura-Netto, P.O., *"RedInv-SA: a simulated annealing for the quadratic assignment problem"*, RAIRO Oper. Res. 33 (1999),249-273.

[Bo87] Bokari, S.H., *"Assignment problems in parallel and distribuited computing"*, Kluwer Academic Publishers, Boston, 1987.

[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/

[DH72] Dickey, J.W. e Hopkins, J.W., *"Campus building arrangement using TOPAZ"*, Transportation Research, 6, (1972), 59-68.

[DMM97] M. Dell'Amico, F. Maffioli and S. Martello, *"Annotated bibliographies in combinatorial optimization"*, Jonh Wiley & Sons, Chichester (1997)

[El77] Elshafei, A,N, *"Hospital layout as a quadratic assignment problem"*, Operations Research Quaterly, 28, (1977), 55-86.

[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.

[GG76] Geoffrion, A.M. e Graves, G.W., *"Scheduling parallel production lines with changeover costs: practical applications of a quadratic assignment problem/LP approach"*, Op. Res. 24 (1976), 595-610.

[Glo89a] Glover, F., *"Tabu search - Part I"*, ORSA J. Comp. 1 (1989), 190-206.

[Glo89b] Glover, F., *"Tabu search - Part II"*, ORSA J. Comp. 2 (1989), 4-32.

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

[GTD97] Gambardella, L.M., Taillard, E.D. e Dorigo, M., *"Ant colonies for the QAP"*, Technical Report IDSIA 97-4, IDSA, Switzerland (1997).

[GW89] Grötschel, M. e Wakabayashi, *"A cutting plane algorithm for a clustering problem"*, Mathematical Programming, 45, (1989), 59-96.

[Hf77] Heffey, D.R., *"Assignment runners to a relay team"* in Optimal Strategies in Sports S.P. Ladany and R.E. Macholeds, Noth – Holland, Amsterdam, (1977), 260-171.

[HLP52] Hardy, G.H., Littlewood, J.E. e Pólya, G., *"Inequalities"*, Cambridge University Press, Cambridge, 1954.

[Hu87] Hubert, L.J., *"Assignment method in combinatorial data analysis"*, Marcel Dekker, Inc, New York, NY 10016 (1987).

[JMRW94] Jünger, M., Martin, A., Reinelt, G. e Weissmantel, R., *"Quadratic 0-1 optimization and a decomposition approach for the placement of eletronic circuits"*, Mathematical Programming, 63, (1994), 257-279.

[KB57] T. C. Koopmans and M. Beckmann, *"Assignment Problems and the Location of Economic Activities"*, Econometrica, (1957), 25, no. 1, pp. 53-76.

[KP78] Krarup, J. e Pruzan, P.M., *"Computed-aided layout design"*, Mathematical Programming Study, 9, (1978), 75-94.

[LM88] Laporte, G. e Mercure, H., *"Balancing hydraulic turbine runners: a quadratic assignment problem"*, EJOR, 35, (1988), 378-382.

[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.

[MCD94] Maniezzo, V., Colorni, A. e Dorigo, M., *"The ant system applied to the quadratic assignment problem"*, Technical Report 94/28, IRIDIA, Université Libre de Bruxelles (1994)

[MP97] Mavridou, T. e Pardalos P.M., *"Simulated annealing and genetic algorithms for the facility layout problem: a survey"*, Computational Optmization and Applications, 7, (1997), 111-126.

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

[St61] Steinberg, L., *"The blackboard wiring problem: a placement algorithm"*, SIAM Review, 3, (1961), 37-50.

[Ta91] Taillard, E., *"Robust Taboo Search for the quadratic assignment problem"*, Parallel Comput., 1991, 17, 4-5, pp. 443-455.

[TS95] Tate, D.M. e Smith, A. E., *"A genetic approach to the quadratic assignment problem"*, Computers and Operations Research, 22, (1995), 73-83.

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

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