CONTRIBUIÇÕES ALGÉBRICAS AO PROBLEMA
QUADRÁTICO DE ALOCAÇÃO

Maria Cristina Rangel

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE PRODUÇÃO.

Aprovada por:

Profa. Nair Maria Maia de Abreu, D.Sc.

Prof. Paulo Oswaldo Boaventura-Netto, Dr.Ing.

Prof. Luiz Satoru Ochi, D.Sc.

Profa. Tânia Maia Querido, D.Sc.

Prof. Arlindo Gomes de Alvarenga, D.Sc

Prof. Oswaldo Vernet de Souza Pires, D.Sc

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2000

RANGEL, MARIA CRISTINA

Contribuições Algébricas ao Problema Quadrático de Alocação [Rio deJaneiro] 2000

xi, 104 p., 29.7 cm, (COPPE/UFRJ, D.Sc., Engenharia de Produção, 2000)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1 - Problema Quadrático de Alocação

2 - Otimização Combinatória

3 - Meta-Heurística

I. COPPE/UFRJ II. Título (Série)

Aos meus pais Elmo e Conceição

Agradecimentos

Gostaria de agradecer a todos que, direta ou indiretamente, contribuiram para que este trabalho se realizasse. Em especial, agradeço:

À Profa. Nair Maria Maia de Abreu, pelo apoio, amizade e dedicação demonstrado durante todo o período de orientação.

Ao PICD/CAPES pelo apoio financeiro através da bolsa de doutorado, indispensável à realização deste trabalho.

Ao Departamento de Informática da Universidade Federal do Espírito Santo que possibilitou meu afastamento para realização do doutorado.

Ao amigo Luis Guillermo Coca Velarde, professor do Departamento de Estatística da Universidade Federal Fluminense, pelo suporte técnico dispensado nas questões em estatística.

Aos professores, funcionários e alunos do Programa de Engenharia de Produção da COPPE-UFRJ pelo apoio oferecido de várias formas e nas várias etapas da realização desta pesquisa. Destacando Andréia Lima da Silva, secretária da PO.

Aos meus queridos amigos que muito contribuíram para encarar com tranquilidade e bom humor tantas horas de trabalho.

Aos meus familiares, que pela simples existência, são a força da minha vida.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CONTRIBUIÇÕES ALGÉBRICAS AO PROBLEMA QUADRÁTICO DE ALOCAÇÃO

Maria Cristina Rangel

Novembro/2000

Orientador: Nair Maria Maia de Abreu
Paulo Oswaldo Boaventura-Netto
Programa: Engenharia de Produção

Utilizando uma abordagem algébrica, estudamos o Problema Quadrático de Alocação por uma relaxação ao Problema de Alocação Linear. Apresentamos uma versão mais simples da prova de um teorema que compara as soluções lineares, independente dos dados de entrada do problema. Este resultado, conhecido como Teorema da Ordenação Parcial Livre, permite definir um *poset* no conjunto desses custos. Tal conjunto contém os custos das soluções viáveis das instâncias numa classe PQA. Quando a comparação é possível, dizemos que tais custos são custos ou soluções livremente comparáveis. Sendo essas soluções representadas por permutações, um outro *poset* é então introduzido no conjunto das permutações cujo grafo de comparabilidade é particionado em níveis pelo número de inversões de cada permutação. Abreu [Ab84] apresenta uma conjectura que relaciona esse número com os custos das permutações livremente comparáveis. Provamos sua conjectura no Teorema das Inversões. As informações daí decorrentes foram utilizadas em procedimentos desenvolvidos, baseados em técnicas meta-heurísticas tais como Grasp e *Simulated Annealing*, aqui também apresentadas. Finalmente, discutimos a possibilidade de avaliar cada solução do PQA, pelo uso do número de inversões da permutação a ela correspondente.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ALGEBRAIC CONTRIBUITIONS FOR THE QUADRATIC ASSIGNMENT PROBLEM

Maria Cristina Rangel

November/2000

Advisor: Nair Maria Maia de Abreu

Paulo Oswaldo Boaventura-Netto

Department: Production Engineering

Through the use of an algebraic approach, we consider the Quadratic Assignment Problem (QAP) a relaxation form of the Linear Assignmet Problem (LAP). We present a simpler version of a theorem proof which compare linear solutions regardless of the problem datas. This result is known as Free Partial Ordering Theorem and lets us define a poset in the set of the problem costs. This set contains all of costs of the available solutions of the QAP-instances. If it's possible to compare pairs of costs or solutions, we call them freely comparable costs or solutions. Once each solution can be represented by only one permutation, another poset is then introduced in the permutation set. This comparability graph is partitioned in levels according to the inversion number of each permutation. Abreu [Ab84] suspects that this inversion number can be linked to the cost of the solution according to the free partial order. Her suspicion is proved right in the Inversion Theorem, which constitutes one of the most important contribuition in this work. We developed some proceedings, based on these algebraic results and metaheuristics technique such as Grasp and Simulated Annealing. Finally, we discuss the possibility of a qualifying each solution according to the number of inversions of its respective permutations.

Liberdade - palavra que o sonho humano alimenta.
Não há ninguém que explique.
Não há ninguém que não entenda.
Cecília Meireles

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nesta seção apresentamos o Problema Quadrático de Alocação, uma breve descrição de suas origens, as formulações matemáticas mais utilizadas, suas principais aplicações, os mais importantes métodos de resolução e ainda, as principais publicações a ele relacionadas. A seguir apresentamos, capítulo a capítulo, o nosso trabalho de tese, com ênfase para as nossas principais contribuições ao referido problema.

1.1 O Problema Quadrático de Alocação

Koopmans e Beckmann [KB57] introduziram o Problema Quadrático de Alocação (PQA), no contexto de localização de atividades econômicas. O objetivo é determinar a alocação ótima de pares de atividades a pares de localidades, quando são conhecidas as distâncias entre as localidades e o fluxo entre as atividades.

Importantes aplicações deste problema combinatório residem nos problemas de arranjo físico, conhecidos como problemas de *Lay-Out*, que buscam a alocação ótima para um conjunto de instalações de uma empresa, em diferentes locais previamente especificados para cada instalação. Nestes casos, os custos das atribuições correspondem aos produtos das distâncias entre localidades, pelos fluxos entre as instalações. Com base nesta abordagem, podemos citar diversos trabalhos: o do próprio precursor Koopmans e Beckmann [KB57], o de Elshafei [El77] que discute a localização de departamentos de um hospital no Cairo; o de Steinberg [St61] que estuda a otimização de circuitos impressos; e ainda, o problema de otimizar os teclados de máquinas de escrever, apresentado por McCormick [Mc70].

O Problema Quadrático de Alocação é um dos problemas *NP-Hard* mais difíceis, no sentido de que, instâncias de ordem superior a 25 ainda são consideradas de grande porte, apesar dos avançados recursos computacionais disponíveis. Recentemente, Anstreicher e Brixius [AB00] encontraram o ótimo da instância proposta por Nugent há 32 anos, conhecida por Nug 30, disponível na QAPLIB [BKR97]. Para isso, os pesquisadores desenvolveram um algoritmo **branch and bound**, utilizando um limite inferior, baseado em programação quadrática convexa.

Diversos problemas de otimização combinatória podem ser tratados como casos particulares do PQA. Os mais conhecidos são: o clássico Problema do Caixeiro Viajante, e os problemas da Clique Máxima, do Particionamento e Isomorfismo em Grafos.

As primeiras contribuições teóricas para o PQA surgiram com os trabalhos de Gilmore [Gi62] e Lawler [La63]. Em seu artigo, Gilmore introduz um limite inferior para o custo da solução ótima do PQA e Lawler, além de apresentar uma formulação mais genérica que a primeira dada em [KB57] para o PQA, modificou o limite de Gilmore, resultando num dos mais importantes limites inferiores do PQA, conhecido por **Gilmore-Lawler**, que até hoje, apesar de inúmeros limites teóricos desenvolvidos, é amplamente empregado pelos pesquisadores para o estudo do PQA.

Com o surgimento desses trabalhos, muitos pesquisadores partiram para estudar este problema, que ainda desafia a ciência, tanto pela sua natureza teórica, quanto pela dificuldade computacional na determinação de valores ótimos para instâncias de ordem relativamente pequenas. A literatura de otimização é rica em artigos dedicados ao PQA: o t rabalho de Finke, Burkard e Rendl [FBR87] apresentam uma diferente formulação, conhecida por formulação do traço do PQA; procedimentos para a determinação de limites inferiores são revisados por Hadley, Rendl e Wolkowicz [HRW90]; o *survey* de Burkard [Bur91] apresenta os resultados mais conhecidos, até aquele momento, e descreve o comportamento assintótico do PQA. Em 1994, Pardalos, Rendl e Wolkowicz [PRW94] editaram o primeiro livro, dedicado exclusivamente ao Problema Quadrático de Alocação, com artigos sobre limite inferiores, complexidade computacional, aproximações heurísticas e generalizações do problema. Mais recentemente, Burkard e Çela [BC97] organizaram um novo *survey* sobre o PQA, encontrado no capítulo 21, do livro de Dell'Amico, Maffioli e Martelo [DMM97], que também apresenta uma vasta bibliografia para problemas de otimização combinatória. Parecendo não esgotar o tema em Alocação Quadrática,

novamente Burkard e Çela, em parceria com Pardalos e Pitsoulis, escreveram um outro *survey*, publicado em [BCPP98]. A importância da pesquisa atual deste problema se consolida com a publicação do livro de Eranda Çela [Ce98]. Neste trabalho, fruto de sua dissertação de tese de doutorado, sua principal contribuição reside no estudo das propriedades estruturais das matrizes que definem instâncias do PQA, permitindo classificá-las em instâncias fáceis ou difíceis, considerando-se o tempo, polinomial ou não, para a sua resolução. Cabe ainda registrar que trabalhos recentes em Otimização Discreta enfocam interesse na obtenção de limites inferiores por meio de Programação Semidefinida, como citado em Anstreicher e Brixius [AB00]. Relaxações para o PQA, via Programação Semidefinida, pode ser encontrada em [18] e [27] do artigo de Anstreicher e Brixius. O mais recente trabalho que encontramos na literatura para o PQA foi desenvolvido por Anstreicher e Brixius, [AB00], que descreve um novo limite inferior, baseado em programação quadrática convexa, que compete com importantes limites existentes em qualidade e esforço computacional.

Finalmente, resta-nos citar a abordagem algébrica e combinatória do Problema Quadrático de Alocação que vem sendo estudada pelos pesquisadores do Grupo de Grafos e Combinatória da COPPE/UFRJ, tendo como início a tese de doutorado de Abreu [Ab84]. A partir daí, vários trabalhos vêm sendo publicados, destacando-se Vernet et al. [VRA95], Abreu et al. [AQB99], Abreu et al. [ABQG00], Rangel et al. [RAB00] e Rangel et al. [RABB00]. Além desses trabalhos citados, algumas teses já foram desenvolvidas Querido [Que94], Firmo [Fi99] e Moreira [Mo98]. Esta tese aqui apresentada, segue essa linha de pesquisa.

1.2 Formulações do PQA

O Problema Quadrático de Alocação genérico, como apresentado por Lawler [La63], trata da alocação de pares (i, j), das n atividades a pares (p, q), das n localidades, de forma a minimizar o custo operacional da instalação completa. O problema pode ser expresso por

$$\min \sum_{i,j,p,q=1}^{n} c_{ijpq} x_{ip} x_{jq}, \tag{1.1}$$

sujeito a

$$\sum_{i=1}^{n} x_{ip} = 1, \quad p = 1, \ldots, n; \tag{1.2}$$

$$\sum_{p=1}^{n} x_{ip} = 1, \qquad i = 1, \ldots, n; \tag{1.3}$$

onde $x_{ip} = \begin{cases} 1, & \text{se a atividade } i \text{ é alocada em } p; \\ 0, & \text{caso contrário.} \end{cases}$

As restrições (1.2) e (1.3) obrigam a alocação de uma, e somente uma, atividade i a um único local p. Neste modelo pode-se adicionar à função objetivo (1.1), os custos de instalação b_{ip} da atividade i à localidade p.

Koopmans e Beckmann em [KB57] consideram os custos c_{ijpq}, como produtos $f_{ij}d_{pq}$, onde $F = [f_{ij}]$ e $D = [d_{pq}]$ são matrizes reais $n \times n$ que representam respectivamente, os fluxos entre as atividades i e j e as distâncias entre as localidades p e q.

Burkard e Stratman [BS78] formulam o PQA genérico, utilizando-se permutações dos elementos do conjunto $\Omega^n = \{1, \ldots, n\}$, para representar as alocações. Desta forma, as restrições (1.2) e (1.3) tornam-se desnecessárias. A expressão matemática para a função objetivo é simplesmente

$$\min_{\varphi \in \Pi_n} \sum_{i,j=1}^{n} c_{ij\varphi(i)\varphi(j)}, \tag{1.4}$$

onde Π_n é o conjunto das permutações dos elementos de Ω^n, $\varphi(i) = p$ e $\varphi(j) = q$

Com a representação das alocações por permutações, os custos na formulação de Koopmans e Beckmann [KB57] são dados por $f_{ij}d_{\varphi(i)\varphi(j)}$, com $\varphi \in \Pi_n$.

Lawler [La63] apresentou uma formulação linear para o PQA, introduzindo variáveis binárias $y_{ijpq} = x_{ip}x_{jq}$, de forma a se obter:

$$\min \sum_{i,j,p,q=1}^{n} c_{ijpq}y_{ijpq}, \tag{1.5}$$

sujeito a

$$\sum_{i=1}^{n} x_{ip} = 1, \qquad p = 1, \ldots, n; \tag{1.6}$$

$$\sum_{i,j,p,q=1}^{n} y_{ijpq} = n^2; \tag{1.7}$$

$$\sum_{p=1}^{n} x_{ip} = 1, \qquad i = 1, \ldots, n; \tag{1.8}$$

$$x_{ip} + x_{jq} - 2y_{ijpq} \geq 0, \quad (i, j, p, q = 1 \ldots, n); \tag{1.9}$$

onde, $x_{ip} = \begin{cases} 1, \text{ se a atividade } i \text{ é alocada em } p; \\ 0, \text{ caso contrário;} \end{cases}$

e $y_{ijpq} = \begin{cases} 1, \text{ se as unidades } i \text{ e } j \text{ são alocadas em } p \text{ e } q, \text{ simultaneamente;} \\ 0, \text{ caso contrário.} \end{cases}$

No entanto, esta formulação linear não torna o problema PQA mais fácil, pois o número de variáveis é $(n^4 + n^2)$ e de restrições é $(n^4 + 2n + 1)$, inviável para uma formulação eficiente.

Existem outras formulações para o PQA, que podem ser encontradas em [Ce98]. Dentre elas cabe destacar a **formulação do traço**, apresentada por Edwards [Ed77], que utiliza propriedades do traço de matrizes simétricas, permitindo derivar a formulação do **Produto de Kronecker**, que é uma extensão da formulação do traço, onde as manipulações algébricas podem ser feitas através do produto de Kronecker (ver detalhes em [Gr81]).

1.3 Algoritmos exatos e heurísticos

Uma grande variedade de algoritmos exatos tem sido proposta para o PQA, contudo, os baseados na técnica **branch and bound** são os que fornecem os melhores resultados.

O desempenho de algoritmos **branch and bound** depende, significativamente, tanto da qualidade e eficiência do limite inferior escolhido, quanto do tipo de implementação utilizada. Edwards [Ed80] faz uma abordagem **branch and bound** baseada na formulação do traço para o PQA que permite o uso eficiente de uma árvore binária de busca; Roucairol [Rou87], Pardalos e Crouse [PC89] e Clausen e Perregaard [CP94] fazem uso da implementação paralela; Mautor e Roucairol em [MR94] exploram a simetria da matriz de custo para reduzir a árvore de um **branch and bound**. Vale a pena ressaltar a dificuldade na resolução da instância Nug 30, resolvido após um período de sete dias com um conjunto de mais de 1000 computadores ao redor do mundo. A técnica exata, do tipo **branch and bound**, utiliza um novo limite determinado por Anstreicher e Brixius [AB00]. O algoritmo reduz o número de alocações eliminando-as quando há grande possibilidade destas não pertencerem à alocação ótima. As **workstations** utilizadas para esse

procedimento são conectadas via **internet**. Tal sistema é conhecido como **Condor**, desenvolvido por Miron Livny com colaboradores na Universidade de Wisconsin. Para implementação do algoritmo, Anstreicher e Brixius utilizaram de uma interface, chamada de **Master-Workes**, com o sistema **Condor**. Pode-se encontrar maiores detalhes sobre a solução da instância Nug 30 no site: http://www-unix.mcs.anl.gov/metaneos/nug30/, onde **MetaNEOS** é um projeto que reúne pesquisadores da Universidade de Wisconsin, Argonne, Universidade de Nortwestern e outras instituições.

Christofides e Benavent [CB89] formulam o PQA como um problema de programação inteira e aplicam a relaxação Lagrangeana, que é resolvida por um esquema de programação dinâmica. Esta abordagem produz limites inferiores bem ajustados para a solução ótima do PQA.

Além dos algoritmos exatos e daqueles que se preocupam em produzir bons limites inferiores para o PQA, a maioria das técnicas de resolução são heurísticas cujas abordagens vão desde métodos de melhoramentos determinísticos e construtivos à busca tabu e algoritmos baseados em simulação.

Li, Pardalos e Resende [LPR94] apresentam uma meta-heurística conhecida por GRASP - *Greedy Randomize Adaptive Search Procedures*, para resolver o Problema Quadrático de Alocação. Esta meta-heurística é um processo iterativo composto de duas fases: construção de soluções iniciais e aplicação de um procedimento de busca local à solução advinda da fase de construção. A primeira fase (soluções iniciais) é aleatória porém, com alguns cuidados que visam gerar boas soluções para economizar esforço computacional no procedimento de busca local. Pardalos et al. propõem uma versão paralela para o GRASP em [PPR95]. O algoritmo GRASP para o PQA é apresentado, em detalhes, no capítulo 5, dedicado às heurísticas. Para esta heurística apresentamos algumas contribuições inserindo informações baseadas na linha de pesquisa que adotamos e que podem ser vistos nos itens 5.3 e 5.4. Vale ressaltar que na proposta de ampliação do espaço de busca, na secção 5.3, o ótimo da instância Nug 30 foi alcançado.

Burkard e Rendl [BR83] aplicaram o *Simulated Annealing (SA)* ao PQA e conseguiram os melhores resultados até aquele momento, para os problemas testes da literatura. Whilhelm e Ward [WW87] e Connolly [Co90] propuseram uma melhoria àquele SA.

Outra meta-heurística aplicada ao PQA é a Busca Tabu. Uma implementação paralela foi desenvolvida por Chakrapani e Skorin-Kapov em [CSk93]. Cuidados com o tamanho da lista tabu e um compromisso entre a diversificação e intensificação da busca são fatores importantes que agem diretamente no desempenho desta meta-heurística. Os pesquisadores Battiti e Tecchiolli [BT94] e Taillard [Tai91] propuseram uma Busca Tabu com essas especificações, levando a bons resultados.

Os algoritmos genéticos (AG's) também são aplicados ao PQA. Tate e Smith [TS94], Fleurent e Ferland [FF94] e Ahuja et al. [AOT95] utilizaram essa abordagem. Fleurent e Ferland [FF94] apresentaram um algoritmo híbrido, tentando melhorar o desempenho dos AG's. Muitos esquemas para os operadores genéticos são encontrados na literatura, favorecendo às muitas versões para os AG's, não só quando aplicado ao PQA, mas para muitos outros problemas de otimização combinatória.

No início desta década, surge uma nova técnica algorítmica conhecida por Sistema de Colônia de Formigas. Este sistema foi aplicado ao PQA por Maniezzo et al. [MCD94] e Gambardella et al. [GTD97].

1.4 Objetivos do trabalho

O Problema Quadrático de Alocação é um problema combinatório que, teoricamente, pode ser resolvido pela enumeração de suas soluções viáveis. Contudo, Sanhi e Gonzalez, [SG76], demonstram que o PQA é fortemente *NP-Hard*, significando que até mesmo se existir um algoritmo heurístico que garanta uma ϵ-aproximação, ϵ uma constante, então $P = NP$. Pode-se observar a dificuldade de resolução do PQA, comparando-o, por exemplo, com o problema do Caixeiro Viajante (PCV) que é *NP-Hard*. Hoje em dia é possível resolver instâncias do PCV com milhares de cidades, enquanto instâncias do PQA com dimensão maior que 25 são consideradas praticamente intratáveis. A instância conhecida por Nug 30, proposta por Nugent há 32 anos, foi resolvida ótimamente em junho de 2000 por Anstreicher e Brixius, [AB00].

Um grande número de pesquisadores vem se dedicando ao desenvolvimento de heurísticas na tentativa de alcançar soluções viáveis de boa qualidade, com tempo computacional aceitável. Alguns métodos de enumeração implícita, que resolvem otimamente as instâncias do PQA, são eficientes para dimensão até 20.

Como já dissemos no item 1.1, este trabalho é baseado na teoria desenvolvida pelos pesquisadores do Grupo de Grafos e Combinatória da área de Pesquisa Operacional (PO-PEP/COPPE/UFRJ).

No capítulo 2, definimos uma relaxação para o Problema Quadrático de Alocação, numa forma correspondente do Problema de Alocação Linear (PAL), no sentido de que o conjunto das soluções do PQA está contido no das soluções do PAL. A dificuldade desse enfoque reside no fato de que o número de soluções do PQA, de dimensão n, é muito menor que o número de soluções do PAL, sendo $N = C_{n,2}$, a sua dimensão. As vantagens desse conjunto de soluções do PAL são vistas no capítulo 3 onde introduzimos uma ordenação parcial no conjunto de permutações, associadas às soluções do PQA, independente da instância do problema. Tal ordenação é chamada de **Ordenação Parcial Livre**. A partir daí, desenvolvemos um algoritmo polinomial que caracteriza pares de permutações comparáveis por esta ordem, que denominamos de **pares de permutações livremente comparáveis**. Esses pares definem as arestas de um grafo cujo conjunto de vértices são as permutações correspondentes às soluções da relaxação linear PAL. Denotamos tal grafo por G_{Liv}. Uma das vantagens de estudarmos o conjunto das soluções do PAL é que tais soluções desenham uma estrutura de Permutaedro, o diagrama de Hasse do conhecido Reticulado das Permutações, que lhe confere diversas propriedades matemáticas a serem exploradas. Com o objetivo de aplicar tais propriedades ao grafo G_{Liv} é preciso conhecer a relação entre as soluções quadráticas e as soluções lineares viáveis ao problema quadrático. Tal relação está descrita no capítulo 2.

Acreditamos que uma das principais contribuições teóricas da tese é a demonstração da conjectura proposta por Abreu [Ab84] que relaciona número de inversões e custo das permutações. A este resultado chamamos **Teorema das Inversões** que utilizamos como medida para avaliar soluções do PQA existentes na literatura e obtidas por algoritmos apresentados no capítulo 5 . O teorema é utilizado na modificação proposta no Grasp Ampliado, [RABB98].

Também no capítulo 5, apresentamos um estudo sobre algumas heurísticas conhecidas aplicadas ao PQA, tais como GRASP, Algoritmos Genéticos e Simulated Annealing. Algumas modificações são propostas para o GRASP e um Algoritmo Genético híbrido é apresentado. Discussões, analisando a eficiência de tais modificações e os resultados obtidos pelo AG, são encontradas nas seções reservadas à apresentação e análise dos resultados computacionais.

De acordo com os estudos no capítulo 4, vemos que o grafo G_{Liv} é particionado em níveis onde cada um deles é constituído por permutações que possuem o mesmo número de inversões. No capítulo 6 apresentamos um estudo para validar o número de inversões como parâmetro de avaliação da qualidade de soluções do Problema Quadrático de Alocação. Para isso, fizemos três testes empíricos: o primeiro determina a posição relativa das permutações que representam soluções ótimas ou viáveis de boa qualidade no grafo G_{Liv} para diversas instâncias do Problema Quadrático de Alocação, disponíveis na QAPLIB [BKR97], o segundo teste compara a curva descrita pela variação dos custos das soluções do PQA, geradas aleatoriamente, com a descrita pela variação do número de inversões das respectivas permutações associadas no grafo G_{Liv}. O terceiro teste utiliza a técnica estatística, conhecida por análise de regressão, para avaliar a tendência de crescimento dos custos das soluções, geradas aleatoriamente, com o número de inversões das suas permutações correspondentes em G_{Liv}.

No capítulo 7, traçamos as conclusões, sugerindo também questões para trabalhos futuros que possam dar continuidade a pesquisa desta tese.

Capítulo 2

Relaxação Linear do Problema Quadrático de Alocação

Neste capítulo apresentamos uma relaxação linear do **Problema Quadrático de Alocação** (PQA) na forma de um **Problema de Alocação Linear** (PAL). Propriedades do conjunto de soluções do PAL são estudadas e definidas soluções viáveis e pseudo-viáveis para uma correspondente instância do PQA. Condições de caracterização destas soluções são apresentadas.

2.1 Relaxação Linear

Resolver uma instância $PQA(F, D)$ do **Problema Quadrático de Alocação**, dada por matrizes quadradas de ordem n F e D, de coordenadas reais (inteiras) não-negativas é achar o valor

$$Z_{\varphi^*} = \min_{\varphi \in \Pi_n} \sum_{i<j} f_{ij} d_{\varphi(i)\varphi(j)} \tag{2.1}$$

onde Π_n é o conjunto de todas as permutações dos elementos de $\Omega^n = \{1, \ldots, n\}$. Este valor é o menor custo para uma atribuição em que pares de facilidades são alocadas a pares de localidades e reciprocamente. Trata-se de um problema de *Lay-Out* que pode ser representado por sobreposição dos vértices de uma clique K_F com arestas valoradas pelos fluxos e uma outra, K_D, com arestas valoradas pelas distâncias, ambas de ordem n. A sobreposição é definida por uma permutação dos vértices $\varphi \in \Pi_n$.

Considere a instância de Gavett-Plyter, denotada por GP66, definida pelas matrizes

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

A figura 2.1 ilustra a sobreposição dos vértices das cliques K_F e K_D estabelecida pela permutação $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}$, denotada apenas pela imagem $\varphi = (2\ 3\ 4\ 1)$. O custo de atribuir, par a par, facilidades às localidades é determinado por $\varphi \in \Pi_n$ e corresponde o valor do somatório em (2.1) expresso por Z_φ. O valor ótimo Z^*_φ é dado pela permutação que minimiza todos os custos Z_φ, $\varphi \in \Pi_n$ e notada por φ^*.

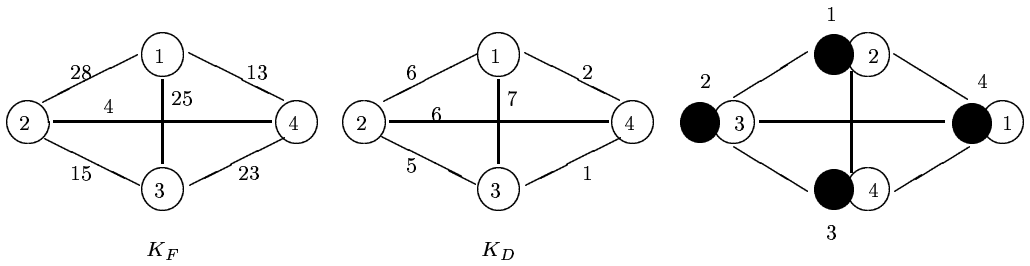


Figura 2.1: A solução para GP66 corresponde a $\varphi = (2\ 3\ 4\ 1)$

Os elementos das matrizes $F = (f_{ij})$ e $D = (d_{kl})$ são armazenados em vetores linhas (de mesmo nome) $\mathbf{F} = (f_z)$ e $\mathbf{D} = (d_z)$, ambos de ordem $N = C_{n,2}$, com coordenadas dispostas segundo a ordem lexicográfica das arestas das cliques K_F e K_D. Para isso, define-se uma bijeção

$$\psi(i,j) = (i-1)n - i(i+1)/2 + j \tag{2.2}$$

que associa o par (i,j), $i < j$ e $i,j = 1,\ldots,n$ a um número natural $z = 1,\ldots,N$, posição que a aresta ocupa no correspondente vetor. Na instância GP66, tem-se $\mathbf{F} = (28\ 25\ 13\ 15\ 4\ 23)$ e $\mathbf{D} = (6\ 7\ 2\ 5\ 6\ 1)$.

Seja a matriz $Q = \mathbf{F}^t\mathbf{D}$ também de ordem N, cujos elementos são os produtos $f_{ij}d_{kl}$, resultantes da sobreposição das arestas das cliques K_F e K_D. Portanto, cada coeficiente de Q corresponde a uma parcela da função objetivo (2.1) e reciprocamente, para cada $\varphi \in \Pi_n$ os coeficientes desta função estão em Q. O **Problema de Alocação Linear**, $PAL(Q)$, de formulação matemática dada por

$$\min_{\xi \in \Pi_N} \sum_{i=1}^{N} f_i d_{\xi(i)}, \tag{2.3}$$

onde Π_N é o conjunto de todas as permutações dos elementos de $\Omega^N = \{1,\ldots,N\}$, pode ser considerado como uma relaxação linear para o problema quadrático, no sentido de o conjunto de soluções viáveis do primeiro conter as do segundo.

A partir de uma dada permutação de vértices, $\varphi \in \Pi_n$, de K_F sobre K_D,

$$\varphi = \begin{pmatrix} 1 & 2 & \dots & n \\ \varphi(1) & \varphi(2) & \dots & \varphi(n) \end{pmatrix},$$

determina-se a correspondente permutação de arestas $\xi_\varphi \in \Pi_N$,

$$\xi_\varphi = \begin{pmatrix} \psi(1,2) & \psi(1,3) & \dots & \psi(n-1,n) \\ \psi(\varphi(1),\varphi(2)) & \psi(\varphi(1),\varphi(3)) & \dots & \psi(\varphi(n-1),\varphi(n)), \end{pmatrix} \tag{2.4}$$

através de ψ em (2.2).

Infelizmente, nem sempre uma permutação $\xi \in \Pi_N$, solução do PAL, representa uma sobreposição de arestas de uma clique sobre a outra compatível com uma de vértices $\varphi \in \Pi_n$. Quando isso ocorre, dizemos que ξ é uma *solução não viável* para a correspondente instância $PQA(F, D)$. Para que ξ seja uma *solução viável* para o Problema Quadrático de Alocação, é necessário que exista $\varphi \in \Pi_n$, tal que se tenha ξ_φ pela expressão (2.4).

O custo da solução do $PAL(Q)$ associado a $\xi \in \Pi_N$ coincide com o produto escalar

$$Z_\xi = \langle \mathbf{F}, \xi(\mathbf{D}) \rangle, \tag{2.5}$$

para $\xi(\mathbf{D}) = (d_{\xi(1)}, d_{\xi(2)}, \dots, d_{\xi(N)})$.

Encontrar uma solução ótima para o $PAL(Q)$ é muito fácil: basta associar a aresta de maior fluxo à de menor distância, sucessivamente, até que a aresta de menor fluxo seja associada à de maior distância. Desta forma, o custo mínimo é o produto escalar $Z^* = \langle \mathbf{F}^-, \mathbf{D}^+ \rangle$, quando os vetores $\mathbf{F}^-$ e $\mathbf{D}^+$ são resultados da ordenação não-crescente e não-decrescente de $\mathbf{F}$ e $\mathbf{D}$ respectivamente.

A instância do PAL definida pela matriz $Q^* = (\mathbf{F}^-)^t(\mathbf{D}^+)$, notada por $PAL(Q^*)$, possui o mesmo conjunto de soluções viáveis que a instância $PAL(Q)$ e portanto, a mesma solução ótima. O traço de Q^*, denotado por $tr(Q^*)$, é o custo mínimo do $PAL(Q)$ e define um limite inferior para o $PQA(F, D)$. Baseando-se neste fato, Abreu et. al, [ABQG00], definiram uma família de instâncias PQA que possuem a matriz Q^* em comum, chamada $RelClass(Q^*)$, definida por

$$RelClass(Q^*) = \{PQA(F, D)/(\mathbf{F}^-)^t(\mathbf{D}^+) = Q^*\}. \tag{2.6}$$

onde $\forall \mathbf{F}$ e $\forall \mathbf{D}$ os elementos são os mesmos, conseqüentemente, o resultado da ordenação são os vetores $\mathbf{F}^-$ e $\mathbf{D}^+$.

Tem-se que o $tr(Q^*)$ é o limite inferior universal para todas as soluções viáveis de qualquer instância dessa família. Analogamente, a soma da diagonal secundária de Q^* é o limite superior universal das instâncias de $RelClass(Q^*)$.

Na instância GP66, os vetores ordenados são

$$\mathbf{F}^- = (28 \;\; 25 \;\; 23 \;\; 15 \;\; 13 \;\; 4) \quad \text{e} \quad \mathbf{D}^+ = (1 \;\; 2 \;\; 5 \;\; 6 \;\; 6 \;\; 7).$$

A tabela 2.1 mostra as matrizes Q e Q^*. Em negrito, os coeficientes em Q estão associados a $\xi = (6\ 3\ 5\ 1\ 2\ 4) \in \Pi_6$. Estes valores correspondem às parcelas do $tr(Q^*)$ na segunda matriz.

Q	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

Q^*	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 2.1: Matrizes Q e Q^* para GP66

O Teorema da Ordenação Parcial Livre (TOPL), enunciado no capítulo 3, compara produtos escalares de dois vetores quaisquer independentemente das suas coordenadas dos vetores. Este teorema pode ser diretamente aplicado ao conjunto de soluções do $PAL(Q^*)$, pois cada solução $\rho \in \Pi_N$, corresponde ao produto escalar

$$Z_\rho = \langle \mathbf{F}^-, \rho(\mathbf{D}^+) \rangle = \sum_{i=1}^{N} f_i^- d_{\rho(i)}^+. \tag{2.7}$$

Tal raciocínio é levado a qualquer instância de $RelClass(Q^*)$ cuja relaxação, como vimos, é a instância do Problema de Alocação Linear $PAL(Q^*)$.

2.2 Viabilidade das soluções lineares

No decorrer do texto notamos por $\xi \in \Pi_N$ uma solução do $PAL(Q)$ e por $\rho \in \Pi_N$, uma solução do $PAL(Q^*)$. Assim, ρ determina uma permutação de colunas em Q^*, e a partir dela pode-se determinar uma correspondente $\xi_\rho \in \Pi_N$ permutação que troca colunas em Q. Para isso, ordenamos $\mathbf{F}$ e $\mathbf{D}$ para obtermos $\mathbf{F}^-$ e $\mathbf{D}^+$, armazenando as trocas feitas nas posições desses vetores nas respectivas permutações auxiliares ϕ_F e ϕ_D.

Exemplo 2.1 *Considere a instância de GP66.*

As referidas permutações são:

$$\phi_F = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 5 & 4 & 6 & 3 \end{pmatrix} \quad \text{e} \quad \phi_D = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 6 & 2 & 3 & 5 & 1 \end{pmatrix}. \tag{2.8}$$

De posse de ϕ_F e ϕ_D e dada uma permutação ρ em Q^*, obtemos a permutação correspondente ξ_ρ em Q pela relação

$$\xi_\rho = \phi_D^{-1} \circ \rho \circ \phi_F. \tag{2.9}$$

Na tabela 2.1, da seção anterior, a permutação identidade representa o $tr(Q^*)$ cuja notação é $id = (1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6)$. Utilizando a relação (2.9), chega-se a ξ_{id} sem dificuldades:

$\xi_{id} = \phi_D^{-1} \circ id \circ \phi_F$;

$\xi_{id} = (6\ \ 3\ \ 4\ \ 1\ \ 5\ \ 2) \circ (1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6) \circ (1\ \ 2\ \ 5\ \ 4\ \ 6\ \ 3)$;

$\xi_{id} = (6\ \ 3\ \ 5\ \ 1\ \ 2\ \ 4)$.

Uma questão importante que nos motiva é:

Dada uma $\rho \in \Pi_N$, que troca colunas em Q^*, como saber se existe uma permutação de vértices da clique K_F sobre os de K_D a ela correspondente?

Para responder esta questão, precisamos da definição a seguir.

Definição 2.1 *Sejam $\rho \in \Pi_N$, o vetor $\mathbf{S}_\rho = (f_1^- d_{\rho(1)}^+, \ldots, f_N^- d_{\rho(N)}^+)$ e um par de cliques K_F e K_D, valoradas pelas coordenadas que determinam $\mathbf{F}^-$ e $\mathbf{D}^+$. Se existe uma permutação $\varphi \in \Pi_n$ de vértices de uma clique sobre a outra, tal que*

$$\mathbf{f}_r^- \mathbf{d}_{\rho(r)}^+ = \mathbf{f}_{\phi_F^{-1}(r)} \mathbf{d}_{\phi_D^{-1}(\rho(r))} = \mathbf{f}_p \mathbf{d}_{\xi_\rho(p)} = f_{\psi^{-1}(p)} d_{\psi^{-1}(\xi_\rho(p))} = f_{ij} d_{kl} \tag{2.10}$$

para $r = 1, \ldots, N$; $\varphi(i) = k$ e $\varphi(j) = l$; $i, j, k, l = 1, \ldots, n$; dizemos que ρ é uma solução **viável** *para a instância $PQA(F, D)$ do conjunto $RelClass(Q^*)$, definida pelo par de cliques.*

O procedimento, denotado por `SolViável`, verifica a viabilidade das permutações em Π_N. Este procedimento utiliza as listas $ListaIJ$ e $ListaKL$ onde são armazenados pares ordenados, representando os vértices das arestas (i, j) e (k, l), resultantes da aplicação da função ψ^{-1} no domínio e na imagem da permutação ξ_ρ em

(2.9). A sobreposição dessas arestas define uma combinação de vértices que pode ser visualizada na figura 2.2. A expressão matemática formal que a representa é $[\varphi(i) = k \wedge \varphi(j) = l] \vee [\varphi(i) = l \wedge \varphi(j) = k]$.

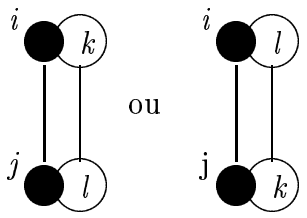


Figura 2.2: Combinação de vértices

Para que uma permutação $\rho \in \Pi_N$ seja viável, seguindo a definição 2.1, é preciso que todas as N combinações dos vértices das arestas pertencentes às $ListaIJ$ e $ListaKL$ sejam compatíveis com a sobreposição de vértices determinada por $\varphi \in \Pi_n$.

SolViável(ρ,ϕ_F,ϕ_D^{-1});

1 calcular $\xi_\rho = \phi_D^{-1} \circ \rho \circ \phi_F$;
2 Para $t = 1, \ldots, N$ faça
3 $ListaIJ[t] \leftarrow \psi^{-1}(t)$;
4 $ListaKL[t] \leftarrow \psi^{-1}(\xi_\rho(t))$;
5 combinar o par (i, j) da $ListaIJ[t]$, com o par (k, l) da $ListaKL[t]$;
6 FimPara;
7 Se $\exists\ p \in \{1, \ldots, n\}$ tal que $\varphi(1) = p$ para as $(n-1)$ primeiras combinações então
8 $\varphi(1) = p$;
9 Para $i = 2, \ldots, n$ faça
10 $[\varphi(i) = l \neq p] \vee (\varphi(i) = k \neq p]$ para $ListaKL[i-1]$;
11 FimPara;
12 Se φ construída possui todas as N combinações compatíveis
13 então ρ é viável;
14 senão ρ não é viável;
15 FimSe;
16 senão ρ não é viável;
17 FimSe;

FimSolViável;

A seguir apresentamos os exemplos 2.2, 2.3 e 2.4 para ilustrar a aplicação do procedimento `SolViável` com a instância GP66. Neste caso, $n = 4$ e $N = C_{4,2} = 6$. As permutações ϕ_F e ϕ_D que armazenam as trocas nas posições de **F** e **D** para ordenar $\mathbf{F}^-$ e $\mathbf{D}^+$ foram apresentadas em (2.8).

Exemplo 2.2 *Seja a permutação* $\rho_1 = (2\ \ 1\ \ 3\ \ 4\ \ 5\ \ 6)$ *em* Π_6.

O cálculo de $\xi_{\rho_1} = \phi_D^{-1} \circ \rho \circ \phi_F$, resulta em

$\xi_{\rho_1} = (6\ \ 3\ \ 4\ \ 1\ \ 5\ \ 2) \circ (2\ \ 1\ \ 3\ \ 4\ \ 5\ \ 6) \circ (1\ \ 2\ \ 5\ \ 4\ \ 6\ \ 3)$;

$\xi_{\rho_1} = (3\ \ 6\ \ 5\ \ 1\ \ 2\ \ 4)$.

A tabela 2.2 auxilia na verificação da viabilidade. Conhecendo-se ξ_{ρ_1} aplica-se ψ^{-1} e, em seguida, faz-se as combinações de vértices.

Seguindo o procedimento `SolViável`, das três primeiras linhas da tabela 2.2, já se determina $\varphi = (4\ \ 1\ \ 3\ \ 2)$. Mas essa permutação não corresponde uma troca de vértices compatível com as combinações nas $N - (n-1)$ linhas restantes da tabela 2.2. Por exemplo, da linha 4 resulta $\varphi(2) = 1$ e $\varphi(3) = 2$ ou $\varphi(2) = 2$ e $\varphi(3) = 1$. E isso não condiz com a φ já construída. Portanto, ρ_1 **é não viável**.

ListaIJ	*ListaKL*	combinações de vértices
$\psi^{-1}(1) = (1,2)$	$\psi^{-1}(3) = (1,4)$	$[\varphi(1) = 1 \wedge \varphi(2) = 4] \vee [\varphi(1) = 4 \wedge \varphi(2) = 1]$
$\psi^{-1}(2) = (1,3)$	$\psi^{-1}(6) = (3,4)$	$[\varphi(1) = 3 \wedge \varphi(3) = 4] \vee [\varphi(1) = 4 \wedge \varphi(3) = 3]$
$\psi^{-1}(3) = (1,4)$	$\psi^{-1}(5) = (2,4)$	$[\varphi(1) = 2 \wedge \varphi(4) = 4] \vee [\varphi(1) = 4 \wedge \varphi(4) = 2]$
$\psi^{-1}(4) = (2,3)$	$\psi^{-1}(1) = (1,2)$	$[\varphi(2) = 1 \wedge \varphi(3) = 2] \vee [\varphi(2) = 2 \wedge \varphi(3) = 1]$
$\psi^{-1}(5) = (2,4)$	$\psi^{-1}(2) = (1,3)$	$[\varphi(2) = 1 \wedge \varphi(4) = 3] \vee [\varphi(2) = 3 \wedge \varphi(4) = 1]$
$\psi^{-1}(6) = (3,4)$	$\psi^{-1}(4) = (2,3)$	$[\varphi(3) = 2 \wedge \varphi(4) = 3] \vee [\varphi(3) = 3 \wedge \varphi(4) = 2]$

Tabela 2.2: Verificação da viabilidade de ρ_1.

Exemplo 2.3 *Seja* $\rho_2 = (2\ \ 1\ \ 3\ \ 6\ \ 5\ \ 4)$ *em* Π_6.

Para este caso, temos $\xi_{\rho_2} = (3\ \ 6\ \ 5\ \ 2\ \ 1\ \ 4)$. Considere a tabela 2.3.

Das três primeiras linhas tem-se a mesma permutação anterior $\varphi = (4\ \ 1\ \ 3\ \ 2)$. Mas agora φ possui todas as N combinações de vértices compatíveis, estabelecidas na terceira coluna da tabela 2.3, resultando em ρ_2 **viável**.

$ListaIJ$	$ListaKL$	combinações de vértices
$\psi^{-1}(1) = (1,2)$	$\psi^{-1}(3) = (1,4)$	$[\varphi(1) = 1 \wedge \varphi(2) = 4] \vee [\varphi(1) = 4 \wedge \varphi(2) = 1]$
$\psi^{-1}(2) = (1,3)$	$\psi^{-1}(6) = (3,4)$	$[\varphi(1) = 3 \wedge \varphi(3) = 4] \vee [\varphi(1) = 4 \wedge \varphi(3) = 3]$
$\psi^{-1}(3) = (1,4)$	$\psi^{-1}(5) = (2,4)$	$[\varphi(1) = 2 \wedge \varphi(4) = 4] \vee [\varphi(1) = 4 \wedge \varphi(4) = 2]$
$\psi^{-1}(4) = (2,3)$	$\psi^{-1}(2) = (1,3)$	$[\varphi(2) = 1 \wedge \varphi(3) = 3] \vee [\varphi(2) = 3 \wedge \varphi(3) = 1]$
$\psi^{-1}(5) = (2,4)$	$\psi^{-1}(1) = (1,2)$	$[\varphi(2) = 1 \wedge \varphi(4) = 2] \vee [\varphi(2) = 2 \wedge \varphi(4) = 1]$
$\psi^{-1}(6) = (3,4)$	$\psi^{-1}(4) = (2,3)$	$[\varphi(3) = 2 \wedge \varphi(4) = 3] \vee [\varphi(3) = 3 \wedge \varphi(4) = 2]$

Tabela 2.3: Verificação da viabilidade de ρ_2.

Exemplo 2.4 *Seja ρ_3 = (4 1 6 5 3 2) permutação de Π_6.*

Calculando a correspondente através de (2.9), tem-se $\xi_{\rho_3} = (1\ \ 6\ \ 4\ \ 5\ \ 3\ \ 2)$.

$ListaIJ$	$ListaKL$	combinações de vértices
$\psi^{-1}(1) = (1,2)$	$\psi^{-1}(1) = (1,2)$	$[\varphi(1) = 1 \wedge \varphi(2) = 2] \vee [\varphi(1) = 2 \wedge \varphi(2) = 1]$
$\psi^{-1}(2) = (1,3)$	$\psi^{-1}(6) = (3,4)$	$[\varphi(1) = 3 \wedge \varphi(3) = 4] \vee [\varphi(1) = 4 \wedge \varphi(3) = 3]$
$\psi^{-1}(3) = (1,4)$	$\psi^{-1}(4) = (2,3)$	$[\varphi(1) = 2 \wedge \varphi(4) = 3] \vee [\varphi(1) = 3 \wedge \varphi(4) = 2]$
$\psi^{-1}(4) = (2,3)$	$\psi^{-1}(5) = (2,4)$	$[\varphi(2) = 2 \wedge \varphi(3) = 4] \vee [\varphi(2) = 4 \wedge \varphi(3) = 2]$
$\psi^{-1}(5) = (2,4)$	$\psi^{-1}(3) = (1,4)$	$[\varphi(2) = 1 \wedge \varphi(4) = 4] \vee [\varphi(2) = 4 \wedge \varphi(4) = 1]$
$\psi^{-1}(6) = (3,4)$	$\psi^{-1}(2) = (1,3)$	$[\varphi(3) = 1 \wedge \varphi(4) = 3] \vee [\varphi(3) = 3 \wedge \varphi(4) = 1]$

Tabela 2.4: Verificação da viabilidade de ρ_3.

Agora não é possível construir φ utilizando-se as $(n-1)$ primeira linhas da tabela 2.4. Neste caso, o procedimento termina com a resposta de que ρ_3 **é não viável**.

2.2.1 Soluções pseudo-viáveis

Pode-se notar que algumas permutações $\rho \in \Pi_N$ permitem construir uma $\varphi \in \Pi_n$, como no exemplo 2.3 mas infelizmente não atendem a definiç ao 2.1 e portanto, não caracterizam uma solução viável para a instância PQA correspondente. Isso conduz a uma solução não viável diferente do que acontece no exemplo 2.4, em que não é possível construir nenhuma permutação de ordem n. Isto nos motiva a introduzir uma nova definição: soluções **pseudo-viáveis**.

Seja $\mathcal{I} \in RelClass(Q^*)$. Note que $\mathcal{I}$ é uma instância $PQA(F, D)$, para um par de matrizes F e D que tenham $\mathbf{F}^-$ e $\mathbf{D}^+$ fatores de Q^*. As questões aqui abordadas, valem para qualquer $\mathcal{I}$, cujas matrizes $n \times n$ sejam simétricas, constituídas por elementos reais não-negativos e diagonais principais nulas. Assim, podemos

representar tais matrizes por vetores de $N = C_{n,2}$ componentes, cujas posições são definidas pela bijeção $\psi(i,j)$ definida pela equação (**??**).

Sabe-se que dentre as $N!$ soluções da relaxação linear, apenas $n!$ obedecem a condição de viabilidade (2.10). Quanto maior o valor de n, menor é a relação entre o número de viáveis e o número total de permutações $\rho \in \Pi_N$, ou seja $\frac{n!}{N!}$.

O procedimento `SolViável` constrói, quando possível, uma solução $\varphi \in \Pi_n$, mas é preciso observar todas as combinações de sobreposição de vértices para saber se são todas compatíveis. No exemplo 2.3, o procedimento determina que ρ é viável para a instância $\mathcal{I} \in RelClass(Q^*)$ pois todas as sobreposições de vértices são compatíveis. Observando o 2.2, construiu-se uma solução $\varphi \in \Pi_n$, porém as sobreposições contidas nas $N - (n-1)$ linhas restantes da tabela 2.2,não condizem com a permutação gerada. Neste caso, tem-se uma permutação $\rho \in \Pi_N$ **pseudo-viável**, cuja definição é apresentada a seguir.

Definição 2.2 *Seja $\rho \in \Pi_N$ solução do $PAL(Q^*)$. Se for possível construir uma permutação $\varphi \in \Pi_n$ utilizando as $(n-1)$ primeiras informações contidas nas listas $ListaIJ$ e $ListaKL$ e se as N combinações de vértices* **não** *forem compatíveis, diz-se que ρ é* **pseudo-viável** *para uma instância $\mathcal{I}$ do conjunto $RelClass(Q^*)$.*

Para melhor descrever a propriedade apresentada a seguir, é necessário particionar a permutação $\rho \in \Pi_N$, em duas partes. Para tanto, introduzimos mais uma definição:

Definição 2.3 *Seja n um número natural e $N = C_{n,2}$. Considere uma permutação $\rho \in \Pi_N$. Chamamos de* cabeça *de ρ as primeiras $(n-1)$ componentes e, de* cauda *de ρ as $N - (n-1)$ componentes restantes dessa permutação.*

Propriedade 2.1 *Dada uma instância $\mathcal{I} \in RelClass(Q^*)$, $n!$ é o número de permutações viáveis $\rho \in \Pi_N$ de $\mathcal{I}$ e o número de suas permutações pseudo-viáveis é $n![(N-(n-1))! - 1]$.*

Prova: O conjunto das $n!$ soluções da instância $PQA(F,D)$ está contido no conjunto das $N!$ soluções do $PAL(Q^*)$. Logo, existem $n!$ permutações em Π_N que são viáveis para $\mathcal{I}$. De acordo com o procedimento `SolViável`, dada $\rho \in \Pi_N$, para

construir $\varphi \in \Pi_n$, é suficiente considerar a cabeça da permutação ξ_ρ. Fixando-se os $(n-1)$ elementos, $\xi_\rho(1), \ldots, \xi_\rho(n-1)$, existem $(N-(n-1))!$ permutações em Π_N que possuem esta mesma cabeça. No entanto, uma única cauda fornece uma sobreposição de vértices compatíveis, restando $(N-(n-1))!-1)$ sobreposições de vértices não compatíveis. Mas isso ocorre $n!$ vezes, o que resulta $n![(N-(n-1))!-1]$ permutações pseudo-viáveis. □

No exemplo 2.4, $\rho_2 = (2\ \ 1\ \ 3\ \ 6\ \ 4\ \ 5)$ é viável para GP66, cuja permutação das arestas de K_F e K_D é $\xi_{\rho_2} = (3\ \ 6\ \ 5\ \ 2\ \ 1\ \ 4)$. Através de `SolViável`, tem-se a permutação de vértices $\varphi = (4\ \ 1\ \ 3\ \ 2)$, cujas imagens $\xi_{\rho_2}(1) = 3$, $\xi_{\rho_2}(2) = 6$ e $\xi_{\rho_2}(3) = 5$ formam a cabeça da permutação. A cauda são as $N-(n-1)$ componentes restantes $\xi_{\rho_2}(4) = 2$, $\xi_{\rho_2}(5) = 1$ e $\xi_{\rho_2}(6) = 4$. Rearranjando as componentes da cauda, tem-se as $[(N-(n-1))!-1] = 5$ permutações pseudo-viáveis associadas a φ. São elas:

1. $\xi = (3\ \ 6\ \ 5\ \ 1\ \ 2\ \ 4)$
   $\rho_\xi = (4\ \ 6\ \ 2\ \ 3\ \ 5\ \ 1) \circ (3\ \ 6\ \ 5\ \ 1\ \ 2\ \ 4) \circ (1\ \ 2\ \ 6\ \ 4\ \ 3\ \ 5)$
   $\rho_\xi = (2\ \ 1\ \ 3\ \ 4\ \ 5\ \ 6)$

2. $\xi = (3\ \ 6\ \ 5\ \ 4\ \ 2\ \ 1)$
   $\rho_\xi = (4\ \ 6\ \ 2\ \ 3\ \ 5\ \ 1) \circ (3\ \ 6\ \ 5\ \ 4\ \ 2\ \ 1) \circ (1\ \ 2\ \ 6\ \ 4\ \ 3\ \ 5)$
   $\rho_\xi = (2\ \ 1\ \ 4\ \ 3\ \ 5\ \ 6)$

3. $\xi = (3\ \ 6\ \ 5\ \ 4\ \ 1\ \ 2)$
   $\rho_\xi = (4\ \ 6\ \ 2\ \ 3\ \ 5\ \ 1) \circ (3\ \ 6\ \ 5\ \ 4\ \ 1\ \ 2) \circ (1\ \ 2\ \ 6\ \ 4\ \ 3\ \ 5)$
   $\rho_\xi = (2\ \ 1\ \ 6\ \ 3\ \ 5\ \ 4)$

4. $\xi = (3\ \ 6\ \ 5\ \ 2\ \ 4\ \ 1)$
   $\rho_\xi = (4\ \ 6\ \ 2\ \ 3\ \ 5\ \ 1) \circ (3\ \ 6\ \ 5\ \ 2\ \ 4\ \ 1) \circ (1\ \ 2\ \ 6\ \ 4\ \ 3\ \ 5)$
   $\rho_\xi = (2\ \ 1\ \ 4\ \ 6\ \ 5\ \ 3)$

5. $\xi = (3\ \ 6\ \ 5\ \ 1\ \ 4\ \ 2)$
   $\rho_\xi = (4\ \ 6\ \ 2\ \ 3\ \ 5\ \ 1) \circ (3\ \ 6\ \ 5\ \ 1\ \ 4\ \ 2) \circ (1\ \ 2\ \ 6\ \ 4\ \ 3\ \ 5)$
   $\rho_\xi = (2\ \ 1\ \ 6\ \ 4\ \ 5\ \ 3)$

Considere então uma instância $PQA(F, D)$, cuja relaxação linear é dada por $PAL(Q^*)$. Podemos agora particionar o conjunto das soluções do $PAL(Q^*)$ em três subconjuntos distintos:

1. o conjunto das soluções **viáveis** da instância $PQA(F, D)$, de acordo com a definição 2.1;

2. o conjunto das soluções **pseudo-viáveis** de $PQA(F, D)$, de acordo com a definição 2.2;

3. e o conjunto das soluções **não viáveis**.

A relação $\frac{n!}{N!}$ tende a zero com muito mais rapidez que a relação $\frac{n![(N-(n-1))!]}{N!}$, quando considerando as soluções pseudo-viáveis.

Capítulo 3

O Teorema da Ordenação Parcial Livre

Todas as instâncias $PQA(F, D)$ da família $RelClass(Q^*)$ têm como relaxação linear o $PAL(Q^*)$ definido no capítulo 2, de modo que cada solução linear é viável para alguma instância desta família. Como já visto, o custo mínimo do Problema de Alocação Linear é o traço da matriz Q^* e seu custo máximo é a soma dos elementos da sua diagonal secundária cujas soluções correspondem às permutações identidade (id) e sua reversa (rev), ambas em Π_N. A comparação desses limites com os demais custos do $PAL(Q^*)$ é independente das coordenadas dos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$. Comparar outros pares de soluções de instâncias da família $RelClass(Q^*)$, independentemente das matrizes F e D que as definem, motivou o desenvolvimento da construção algébrica do *poset* no conjunto de soluções da relaxação do PQA, resultante do **Teorema da Ordenação Parcial Livre** (TOPL). Tal teorema é aqui apresentado com uma prova muito mais simples que a apresentada em [Ab84] e define uma **Ordem Parcial Livre**, denotada por " $\leq_l$ ", determinando o *poset* $(C(Z_\rho), \leq_l)$ no conjunto $C(Z_\rho)$ dos produtos escalares $\langle \mathbf{F}^-, \rho(\mathbf{D}^+) \rangle$, $\rho \in \Pi_N$. Neste capítulo, apresentamos ainda um algoritmo polinomial que caracteriza pares de permutações livremente comparáveis, determinando assim um novo *poset* a ele compatível. O grafo de comparabilidade deste novo *poset*, objeto de estudo do capítulo seguinte, inclui o já conhecido Reticulado das Permutações.

3.1 Ordenação parcial de produtos escalares

Nesta seção apresentamos propriedades dos produtos escalares entre vetores que podem ser aplicadas às soluções da relaxação linear $PAL(Q^*)$ das instâncias da $RelClass(Q^*)$. O custo de cada $\rho \in \Pi_N$, solução do $PAL(Q^*)$, é dado pelo produto escalar $Z_\rho = \langle \mathbf{F}^-, \rho(\mathbf{D}^+) \rangle$.

3.1.1 A partição canônica

Sejam $\mathbf{F}^-$ e $\mathbf{D}^+$ vetores de N coordenadas reais não-negativas dispostas em ordem não-crescente e não-decrescente respectivamente. Considere $Z_{\rho_1} = \langle \mathbf{F}^-, \rho_1(\mathbf{D}^+) \rangle$ e $Z_{\rho_2} = \langle \mathbf{F}^-, \rho_2(\mathbf{D}^+) \rangle$ os produtos escalares entre o vetor $\mathbf{F}^-$ e aqueles resultantes da aplicação de ρ_1 e $\rho_2 \in \Pi_N$ às posições das coordenadas de $\mathbf{D}^+$.

Uma questão importante:

Será possível concluir que $Z_{\rho_1} \leq Z_{\rho_2}$ ou $Z_{\rho_2} \leq Z_{\rho_1}$ sem o conhecimento prévio das componentes dos vetores?

Da diferença entre os produtos escalares tem-se:

$$Z_{\rho_1} - Z_{\rho_2} = \langle \mathbf{F}^-, \rho_1(\mathbf{D}^+) \rangle - \langle \mathbf{F}^-, \rho_2(\mathbf{D}^+) \rangle = \langle \mathbf{F}^-, (\rho_1(\mathbf{D}^+) - \rho_2(\mathbf{D}^+)) \rangle$$

e conseqüentemente,

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t=1}^{N} f_t^-(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}). \tag{3.1}$$

Podemos particionar $\Omega^N = \{1, \ldots, N\}$ em três subconjuntos disjuntos, comparando-se os valores de $\rho_1(t)$ e $\rho_2(t)$, o que resulta em:

(*i*) $P_N = \{t \in \Omega^N / \rho_1(t) - \rho_2(t) < 0\}$, índices correspondentes aos termos não-positivos em (3.1);

(*ii*) $P_P = \{t' \in \Omega^N / \rho_1(t') - \rho_2(t') > 0\}$, índices correspondentes aos termos não-negativos em (3.1);

(*iii*) $P_0 = \{t'' \in \Omega^N / \rho_1(t'') - \rho_2(t'') = 0\}$, índices correspondentes aos termos chamados de livremente nulos em (3.1).

Chamamos de **união ordenada** no conjunto de índices de Ω^N, $P_N \cup P_0 \cup P_P$, induzida pelos subconjuntos de índices estabelecidos em (i), (ii) e (iii).

As parcelas de $Z_{\rho_1} - Z_{\rho_2}$ em (3.1) são particionadas em **não-positivas**, **não-negativas** e **livremente nulas**, estendendo-se a partição induzida pela união ordenada à diferença dos produtos escalares $Z_{\rho_1} - Z_{\rho_2}$, seguindo os índices $t's$ de suas componentes.

Eliminando-se o somatório referente à parcela livremente nula, (3.1) resulta em

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t \in P_N} f_t^-(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) + \sum_{t' \in P_P} f_{t'}^-(d^+_{\rho_1(t')} - d^+_{\rho_2(t')}). \tag{3.2}$$

Definimos como **partição canônica** da equação (3.2) reescrita como em (3.3), pela inserção de termos simétricos a cada fator-diferença em cada parcela de (3.2), de modo que não haja alteração no seu valor, resultando em

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} = & \sum_{t \in P_N} f_t^- \sum_{i=0}^{\rho_2(t)-\rho_1(t)-1} (d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) + \\ & \sum_{t' \in P_P} f_{t'}^- \sum_{j=0}^{\rho_1(t')-\rho_2(t')-1} (d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1}). \end{aligned} \tag{3.3}$$

Segue-se um exemplo de partição canônica para a diferença dos custos Z_{ρ_1} e Z_{ρ_2}, onde ρ_1 e ρ_2 são dados no exemplo 3.1.

Exemplo 3.1 *Sejam* ρ_1 *=(2 4 6 1 3 5) e* ρ_2 *=(6 2 3 4 1 5).*

Considerando-se $\Omega^N = \{1, \ldots, N\}$ tem-se $P_N = \{1, 4\}$, $P_0 = \{6\}$ e $P_P = \{2, 3, 5\}$. Em seguida, calcula-se $Z_{\rho_1} - Z_{\rho_2}$, descartando os termos relativos a P_0.

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} = \; & f_1^-(d^+_{\rho_1(1)} - d^+_{\rho_2(1)}) + f_4^-(d^+_{\rho_1(4)} - d^+_{\rho_2(4)}) + \\ & f_2^-(d^+_{\rho_1(2)} - d^+_{\rho_2(2)}) + f_3^-(d^+_{\rho_1(3)} - d^+_{\rho_2(3)}) + \\ & f_5^-(d^+_{\rho_1(5)} - d^+_{\rho_2(5)}); \end{aligned}$$

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} = \; & f_1^-(d_2^+ - d_6^+) + f_4^-(d_1^+ - d_4^+) + \\ & f_2^-(d_4^+ - d_2^+) + f_3^-(d_6^+ - d_3^+) + \\ & f_5^-(d_3^+ - d_1^+). \end{aligned}$$

Inserindo-se os termos simétricos tem-se:

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} = \quad & f_1^-(d_2^+ - d_3^+ + d_3^+ - d_4^+ + d_4^+ - d_5^+ + d_5^+ - d_6^+) + \\ & f_4^-(d_1^+ - d_2^+ + d_2^+ - d_3^+ + d_3^+ - d_4^+) + \\ & f_2^-(d_4^+ - d_3^+ + d_3^+ - d_2^+) + \\ & f_3^-(d_6^+ - d_5^+ + d_5^+ - d_4^+ + d_4^+ - d_3^+) + \\ & f_5^-(d_3^+ - d_2^+ + d_2^+ - d_1^+). \end{aligned}$$

Lema 3.1 *Na partição canônica de $Z_{\rho_1} - Z_{\rho_2}$, como em (3.3), o simétrico do somatório dos fatores-diferença das parcelas negativas é igual ao somatório dos fatores-diferença das parcelas positivas, ou seja,*

$$-\sum_{t\in P_N}\sum_{i=0}^{\rho_2(t)-\rho_1(t)-1}(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) = \sum_{t'\in P_P}\sum_{j=0}^{\rho_1(t')-\rho_2(t')-1}(d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1}).$$

Além disso, o valor absoluto de cada fator-diferença do primeiro membro corresponde a um fator-diferença igual no segundo membro e reciprocamente.

Prova: Sendo ρ_1 e ρ_2 pertencentes a Π_N, temos

$$\sum_{t\in\Omega_N}(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) = 0.$$

Esta equação pode ser reescrita utilizando-se os subconjuntos de índices da união ordenada $P_N \cup P_0 \cup P_P$ o que nos conduz a

$$\sum_{t\in P_N}(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) + \sum_{t'\in P_P}(d^+_{\rho_1(t')} - d^+_{\rho_2(t')}) = 0,$$

e consequentemente a

$$-\sum_{t\in P_N}(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) = \sum_{t'\in P_P}(d^+_{\rho_1(t')} - d^+_{\rho_2(t')}).$$

Introduzindo-se os termos simétricos, o valor da expressão não se altera:

$$-\sum_{t\in P_N}\sum_{i=0}^{\rho_2(t)-\rho_1(t)-1}(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) = \sum_{t'\in P_P}\sum_{j=0}^{\rho_1(t')-\rho_2(t')-1}(d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1}). \tag{3.4}$$

Para provar a segunda parte do lema 3.1, observamos a expressão (3.4). Nela, $\forall\ t \in P_N$ e $i \in \{0, \ldots, \rho_2(t) - \rho_1(t) - 1\}$, existe um $t' \in P_P$ e $j \in \{0, \ldots, \rho_1(t') - \rho_2(t') - 1\}$ tal que

$$\rho_1(t) + i = \rho_1(t') - j - 1 \text{ e } \rho_1(t) + i + 1 = \rho_1(t') - j,$$

resultando em

$$-(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) = (d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1}).$$

□

Na expressão da partição canônica de $Z_{\rho_1} - Z_{\rho_2}$ no exemplo 3.1, para cada parcela associada a $t \in P_N$, há sempre uma correspondente associada a $t' \in P_P$.

3.1.2 Propriedades da partição canônica

Considere na expressão (3.3) da partição canônica de $Z_{\rho_1} - Z_{\rho_2}$ as seguintes definições:

- **Parcela Não-Positiva** é aquela na forma $N_t^i = f_t^-(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1})$, $t \in P_N$ e $i = 0, \ldots, \rho_2(t) - \rho_1(t) - 1$ e
- **Parcela Não-Negativa** é aquela na forma $P_{t'}^j = f_t^-(d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1})$, $t' \in P_P$ e $j = 0, \ldots, \rho_1(t') - \rho_2(t') - 1$.

Propriedade 3.1 *Existe pelo menos uma bijeção β que associa cada parcela N_t^i a uma $P_{t'}^j$ e reciprocamente, tal que seus respectivos fatores-diferença, em valor absoluto, sejam iguais.*

Prova: Basta construir uma bijeção β tal que $\beta(N_t^i) = P_{t'}^j$, $\forall\ t \in P_N$ e $i = 0, \ldots, \rho_2(t) - \rho_1(t) - 1$. A existência da parcela não-negativa $P_{t'}^j$, para $t' \in P_P$ e $j = 0, \ldots, \rho_1(t') - \rho_2(t') - 1$ e sua correspondente N_t^i, é garantida pelo lema 3.1.□

No exemplo 3.1, a expressão canônica $Z_{\rho_1} - Z_{\rho_2}$ permite construir a bijeção β, associando parcelas não-negativas às não-positivas pelos seus fatores-diferença, seguindo a propriedade 3.1. Podemos dizer que β induz o par ordenado (t, t') tal que $\beta(N_t^i) = P_{t'}^j$, onde $t \in P_N$ e $t' \in P_P$.

A unicidade de β não é garantida, dada a possível repetição de fatores-diferença. Observando a tabela 3.1, temos o fator-diferença $(d_2^+ - d_3^+)$ nas parcelas N_1^0 e N_4^1 e cujos correspondentes, P_2^1 e P_5^0, estão na segunda coluna. A bijeção β então associa $N_1^0 \to P_2^1$ e $N_4^1 \to P_5^0$, induzindo os pares (1,2) e (4,5), de acordo com tabela 3.1.

N_t^i	$P_{t'}^j$	β	(t,t')
$N_1^0 = f_1^-(d_2^+ - d_3^+)$	$P_2^0 = f_2^-(d_4^+ - d_3^+)$	$\beta(N_1^0) = P_2^1$	$(1,2)$
$N_1^1 = f_1^-(d_3^+ - d_4^+)$	$P_2^1 = f_2^-(d_3^+ - d_2^+)$	$\beta(N_1^1) = P_2^0$	$(1,2)$
$N_1^2 = f_1^-(d_4^+ - d_5^+)$	$P_3^0 = f_3^-(d_6^+ - d_5^+)$	$\beta(N_1^2) = P_3^1$	$(1,3)$
$N_1^3 = f_1^-(d_5^+ - d_6^+)$	$P_3^1 = f_3^-(d_5^+ - d_4^+)$	$\beta(N_1^3) = P_3^0$	$(1,3)$
$N_4^0 = f_4^-(d_1^+ - d_2^+)$	$P_3^2 = f_3^-(d_4^+ - d_3^+)$	$\beta(N_4^0) = P_5^1$	$(4,5)$
$N_4^1 = f_4^-(d_2^+ - d_3^+)$	$P_5^0 = f_5^-(d_3^+ - d_2^+)$	$\beta(N_4^1) = P_5^0$	$(4,5)$
$N_4^2 = f_4^-(d_3^+ - d_4^+)$	$P_5^1 = f_5^-(d_2^+ - d_1^+)$	$\beta(N_4^2) = P_3^2$	$(4,3)$

Tabela 3.1: Parcelas não-positivas, não-negativas, bijeção β e par ordenado (t,t')

No entanto, poderíamos ter optado pela construção de outra bijeção que resultasse na seguinte correspondência entre as parcelas: $N_1^0 \rightarrow P_5^0$ e $N_4^1 \rightarrow P_2^1$, com seus respectivos pares (1,5) e (4,2).

Propriedade 3.2 *Para toda bijeção β tal que $\beta(N_t^i) = P_{t'}^j$ pode-se induzir pares ordenados (t,t') com $t \in P_N$ e $t' \in P_P$. Tem-se a partição canônica dada por*

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t \in P_N} \sum_{i=0}^{\rho_2(t)-\rho_1(t)-1} (f_t^- - f_{t'}^-)(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}),$$

onde f_t^- e $f_{t'}^-$ são as componentes t e t' do vetor $\mathbf{F}^-$ correspondente ao par ordenado (t,t').

Prova: Da expressão (3.3)

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} = \; & \textstyle\sum_{t \in P_N} f_t^- \sum_{i=0}^{\rho_2(t)-\rho_1(t)-1} (d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) + \\ & \textstyle\sum_{t' \in P_P} f_{t'}^- \sum_{j=0}^{\rho_1(t')-\rho_2(t')-1} (d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1}). \end{aligned}$$

Da propriedade 3.1, $\beta(N_t^i) = P_{t'}^j$ associa parcelas de fatores-diferença iguais em valor absoluto. Portanto, colocando-se em evidência tais fatores-diferença, induzindo os pares (t,t'), a expressão anterior resulta em

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t \in P_N} \sum_{i=0}^{\rho_2(t)-\rho_1(t)-1} (f_t^- - f_{t'}^-)(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}). \qquad (3.5)$$

□

3.1.3 Produtos escalares livremente comparáveis

Sejam $\mathbf{F}^-$ e $\mathbf{D}^+$ vetores constituídos por N coordenadas reais e não-negativas com ordenação não-crescente e não-decrescente, respectivamente. Seja Π_N o conjunto das permutações de $\Omega^N = \{1, \ldots, N\}$ e considere o conjunto dos produtos escalares como definido abaixo:

$$C(Z_\rho) = \{Z_\rho = \langle \mathbf{F}^-, \rho(\mathbf{D}^+)\rangle / \rho \in \Pi_N\}. \tag{3.6}$$

Dizemos que Z_{ρ_1} e Z_{ρ_2} são **livremente comparáveis**, se e somente se, uma das condições for satisfeita: $Z_{\rho_1} \leq Z_{\rho_2}$ ou $Z_{\rho_2} \leq Z_{\rho_1}$, independentemente das coordenadas dos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$.

Notação: Para $Z_{\rho_1} \leq Z_{\rho_2}$ ou $Z_{\rho_2} \leq Z_{\rho_1}$, independente das entradas dos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$, estas relações serão denotadas por $Z_{\rho_1} \leq_l Z_{\rho_2}$ ou $Z_{\rho_2} \leq_l Z_{\rho_1}$, respectivamente.

A bijeção β resultante da propriedade 3.1 permite verificar se dois produtos escalares são livremente comparáveis, a partir do Teorema da Ordenação Parcial Livre (TOPL), enunciado a seguir.

Teorema 3.1 (Teorema da Ordenação Parcial Livre) *Temos $Z_{\rho_1} \leq_l Z_{\rho_2}$ ($Z_{\rho_2} \leq_l Z_{\rho_1}$), se e somente se, existir uma bijeção β tal que $\beta(N_t^i) = P_{t'}^j$; $t \in P_N$; $t' \in P_P$; $i = 0, \ldots, \rho_2(t) - \rho_1(t) - 1$ e $j = 0, \ldots, \rho_1(t') - \rho_2(t') - 1$ capaz de induzir o par ordenado (t, t'), com $t < t'$ ($t' < t$), $\forall\ t \in P_N$ e $\forall\ t' \in P_P$.*

Prova:

($\Rightarrow$) Segue diretamente do fato de se ter $\mathbf{F}^-$ e $\mathbf{D}^+$ ordenados em ordem não-crescente e não-decrescente, e aplicando-se a propriedade 3.2.

($\Leftarrow$) Considere $Z_{\rho_1} \leq_l Z_{\rho_2}$. Da propriedade 3.1 temos a existência da bijeção β tal que $\beta(N_t^i) = P_{t'}^j$ com $t \in P_N$; $t' \in P_P$; $i = 0, \ldots, \rho_2(t) - \rho_1(t) - 1$ e $j = 0, \ldots, \rho_1(t') - \rho_2(t') - 1$.

Suponhamos a possibilidade de termos, ao mesmo tempo, $Z_{\rho_1} \leq_l Z_{\rho_2}$ e $t'^* < t^*$ para algum $t^* \in P_N$ e $t'^* \in P_P$ e uma bijeção β.

Da propriedade 3.2 tem-se a partição canônica

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t \in P_N} \sum_{i=0}^{\rho_2(t) - \rho_1(t) - 1} (f_t^- - f_{t'}^-)(d_{\rho_1(t)+i}^+ - d_{\rho_1(t)+i+1}^+) \leq 0.$$

As parcelas associadas ao par (t^*, t'^*) induzem ao fator $(f^-_{t^*} - f^-_{t'^*}) \leq 0$. Como todos os fatores-parcela $(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}) \leq 0$, então

$$\sum_{i=0}^{\rho_2(t^*)-\rho_1(t^*)-1} (f^-_{t^*} - f^-_{t'^*})(d^+_{\rho_1(t^*)+i} - d^+_{\rho_1(t^*)+i+1}) \geq 0.$$

Tomemos uma instância PQA tal que

$$L = \sum_{i=0}^{\rho_2(t^*)-\rho_1(t^*)-1} (f^-_{t^*} - f^-_{t'^*})(d^+_{\rho_1(t^*)+i} - d^+_{\rho_1(t^*)+i+1})$$

seja suficientemente grande para termos

$$L \geq \sum_{t \in P_N - \{t^*\}} \sum_{i=0}^{\rho_2(t)-\rho_1(t)-1} (f^-_{t} - f^-_{t'})(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1}).$$

Para essa instância, ocorre a contradição, dado que $Z_{\rho_1} - Z_{\rho_2} \leq 0$.

Portanto, se $Z_{\rho_1} \leq_l Z_{\rho_2}$, o par ordenado (t, t') induzido por β é tal que $t < t'$, $\forall\ t \in P_N$ e $\forall\ t' \in P_P$. □

O Teorema da Ordenação Parcial Livre estabelece uma relação de ordem "$\leq_l$" entre os produtos escalares no conjunto $C(Z_\rho)$, definido em (3.6), resultando um *poset* $(C(Z_\rho), \leq_l)$.

Seja a função f que associa o conjunto Π_N ao $C(Z_\rho)$, definida por

$$\begin{array}{rccl} f: & \Pi_N & \rightarrow & C(Z_\rho) \\ & \rho & \mapsto & Z_\rho = \langle \mathbf{F}^-, \rho(\mathbf{D}^+) \rangle. \end{array} \tag{3.7}$$

Podemos induzir uma nova ordenação entre as permutações de Π_N, através desta função f, dada pela relação

$$\rho_i \leq_l \rho_j, \text{ se e somente se, } f(\rho_i) \leq_l f(\rho_j), i, j = 1, \ldots, N!, \tag{3.8}$$

onde $f(\rho) = Z_\rho$. Tal relação de ordem é chamada de *livremente comparável*, também denotada por "$\leq_l$", resultando um *poset* $(\Pi_N, \leq_l)$ compatível com $(C(Z_\rho), \leq_l)$. Dizemos então que permutações livremente comparáveis são aquelas que pertencem ao *poset* $(\Pi_N, \leq_l)$.

Definição 3.1 *Considere um conjunto V e uma relação de ordem "$<$". Um poset é um conjunto parcialmente ordenado pela relação "$<$". Notamos por $P = (V, <)$. Então $G_P = (V, E_P)$ com $xy \in E_P$ se $x < y$ ou $y < x$ é o* **Grafo de Comparabilidade** *do poset P. Tem-se que $G = (V, E)$ é um grafo de comparabilidade se existir um poset P tal que $G \sim G_P$.*

De acordo com a definição 3.1, o grafo de comparabilidade deste novo *poset* $(\Pi_N, \leq_l)$ é dado por $G_{Liv} = (\Pi_N, M)$, onde Π_N é o conjunto das permutações de $\Omega^N = \{1, \ldots, N\}$ e M é o conjunto dos arcos $\{(\rho_i, \rho_j)/Z_{\rho_i} \leq_l Z_{\rho_j},\ i, j = 1, \ldots, N!\}$. Dizemos que a ordenação é "livre" pelo fato de envolver permutações cujos custos (produtos escalares) podem ser comparados independentemente das entradas dos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$.

3.2 A bijeção canônica β_c

Para verificação de pares de soluções livremente comparáveis é preciso determinar uma bijeção β que pela propriedade 3.1, não é necessariamente única. Com o objetivo de buscar uma bijeção β que satifaça o teorema 3.1 (TOPL), evitando-se enumerar todas as possíveis bijeções, propomos o algoritmo denominado `AlgBijeção`, composto por três procedimentos que exploram a ordenação das coordenadas dos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$.

O primeiro procedimento `Proc1` determina os subconjuntos P_N, P_0 e P_P de Ω^N.

Proc1(ρ_1,ρ_2);
1 $P_N = P_0 = P_P = \emptyset$;
2 Para $t = 1, \ldots, N$ faça
3 Se $\rho_1(t) < \rho_2(t)$ então
4 $P_N \leftarrow P_N \cup \{t\}$;
5 senão
6 Se $\rho_1(t) = \rho_2(t)$ então
7 $P_0 \leftarrow P_0 \cup \{t\}$;
8 senão $P_P \leftarrow P_P \cup \{t\}$;
9 FimSe;
10 FimSe;
11 FimPara;
FimProc1;

Analisando os elementos dos conjuntos gerados pelo procedimento `Proc1` segue o lema 3.2 enunciado a seguir.

Lema 3.2 *Sejam os subconjuntos P_N, P_0 e P_P resultantes da aplicação do* `Proc1`. *Tais subconjuntos possuem seus elementos em ordem crescente e $\forall t \in P_N$ e $\forall t' \in P_P$,*

se tivermos exclusivamente uma das desigualdades, $t' < t$ ou $t < t'$, então os produtos escalares Z_{ρ_1} e Z_{ρ_2} são livremente comparáveis.

Prova: A ordenação crescente dos subconjuntos P_N, P_0 e P_P é consequência imediata do comando de repetição " Para $t = 1, \ldots, N$ faça "(segunda linha de `Proc1`). Quanto aos produtos escalares serem livremente comparáveis, segue direto da propriedade 3.2. □

Exemplo 3.2 *Sejam $\rho_1 = (6\ \ 5\ \ 3\ \ 4\ \ 1\ \ 2)$ e $\rho_2 = (2\ \ 3\ \ 6\ \ 4\ \ 1\ \ 5)$.*

Tem-se os subconjuntos $P_N = \{3, 6\}$, $P_0 = \{4, 5\}$ e $P_P = \{1, 2\}$. Pode-se observar que $\forall\ t \in P_N$ e $t' \in P_P$, $t' < t \Rightarrow Z_{\rho_2} \leq Z_{\rho_1}$.

Se as hipóteses do lema 3.2 não forem satisfeitas, propomos ainda mais dois procedimentos, denotados por `Proc2` e `Proc3`, para continuar a construção da bijeção, que consideramos como "adequada" ao teorema 3.1 (TOPL).

O procedimento `Proc2` gera duas listas, $ListaN$ e $ListaP$, que armazenam os índices das parcelas não-positivas $N_t^i = f_t^-(d^+_{\rho_1(t)+i} - d^+_{\rho_1(t)+i+1})$ e das não-negativas $P_{t'}^j = f_t^-(d^+_{\rho_1(t')-j} - d^+_{\rho_1(t')-j-1})$, utilizando os conjuntos P_N, P_P. Finalmente o `Proc3`, gera o par ordenado (t, t'), induzido pela bijeção que desejamos. Ambos os procedimentos são apresentados a seguir.

Proc2(P_N,P_P);

```
1   ListaN = ListaP = ∅;
2   Para t ∈ P_N faça
3       Para i = 0, ..., ρ2(t) − ρ1(t) − 1 faça
4           ListaN ← (t, ρ1(t) + i, ρ1(t) + i + 1);
5       FimPara;
6   Para t' ∈ P_P faça
7       Para j = 0, ..., ρ1(t') − ρ2(t') − 1 faça
8           ListaP ← (t', ρ1(t') − j, ρ1(t') − j − 1);
9       FimPara;
10   FimPara;
```

FimProc2;

Pode-se perceber que as listas advindas do `Proc2` obedecem a ordem crescente nos seus índices. Definimos como **simétricos** os elementos das listas $ListaN$ e

$ListaP$ tais que, para $t \in P_N$ e $t' \in P_P$ tem-se

$$\rho_1(t) + i = \rho_1(t') - j - 1 \quad \text{e} \quad \rho_1(t) + i + 1 = \rho_1(t') - j.$$

O procedimento `Proc3` encontra os elementos **simétricos**.

Proc3($ListaN$**,**$ListaP$**,**P_N**);**

```
1   Para k = 1, ..., |ListaN| faça
2      buscar em ListaP dentre os elementos não marcados o
         primeiro simétrico de ListaN[k];
3      marcar elemento encontrado;
4      formar o par (t, t');
5      ListaPares ← (t, t')
6   FimPara;
```

FimProc3;

Lema 3.3 *Seja a lista ListaPares resultante da aplicação do* `Proc3`. *Se para todo par* $(t, t') \in ListaPares$, *tivermos exclusivamente* $t' < t$ *ou* $t < t'$ *então* Z_{ρ_1} *e* Z_{ρ_2} *são livremente comparáveis. Para* $t' < t \Rightarrow Z_{\rho_2} \leq Z_{\rho_1}$ *e para* $t < t' \Rightarrow Z_{\rho_1} \leq Z_{\rho_2}$.

Prova: Segue direto da propriedade 3.2. □

Um exemplo da aplicação do lema 3.3.

Exemplo 3.3 *Sejam* $\rho_1 =$ *(2 1 3 6 4 5) e* $\rho_2 =$ *(2 6 3 1 5 4) permutações em* Π_6.

O `Proc1` gera os subconjuntos $P_N = \{2, 5\}$, $P_0 = \{1, 3\}$ e $P_P = \{4, 6\}$. Na construção das $ListaN$ e $ListaP$ não há necessidade do cálculo dos valores das parcelas referentes ao subconjunto P_0. A tabela 3.2 auxilia na construção da $ListaPares$.

Na tabela 3.2 a $ListaPares = \{(2,4), (2,4), (2,4), (2,4), (2,4), (5,6)\}$ possui sempre a primeira coordenada dos pares (t, t') menor que a segunda e, após a aplicação do lema 3.3, concluímos que $Z_{\rho_1} \leq_l Z_{\rho_2}$.

De posse desses procedimentos e dos lemas 3.2 e 3.3, apresentamos o `AlgBijeção`.

$ListaN$	$ListaP$	simétricos	$ListaPares$
(2,1,2)	(4,6,5)	$(2,1,2) \leftrightarrow (4,2,1)$	(2,4)
(2,2,3)	(4,5,4)	$(2,2,3) \leftrightarrow (4,3,2)$	(2,4)
(2,3,4)	(4,4,3)	$(2,3,4) \leftrightarrow (4,4,3)$	(2,4)
(2,4,5)	(4,3,2)	$(2,4,5) \leftrightarrow (4,5,4)$	(2,4)
(2,5,6)	(4,2,1)	$(2,5,6) \leftrightarrow (4,6,5)$	(2,4)
(5,4,5)	(6,5,4)	$(5,4,5) \leftrightarrow (6,5,4)$	(5,6)

Tabela 3.2: $ListaN$, $ListaP$ e $ListaPares$

AlgBijeção(ρ_1, ρ_2);

```
1   Proc1(ρ1, ρ2);
2   Se Lema 3.2 não é satisfeito
      então
3         Proc2(PN,PP);
4         Proc3(ListaN,ListaP,PN);
5         Se Lema 3.3 é satisfeito
6            então ρ1 e ρ2 são livremente comparáveis
7            senão ρ1 e ρ2 não são livremente comparáveis
8         FimSe;
9      senão ρ1 e ρ2 são livremente comparáveis
10  FimSe;
```

FimProcBijeção;

Lema 3.4 *O* `AlgBijeção` *tem complexidade* $O(N^4)$*, onde* $N = C_{n,2}$.

Prova: O pior caso ocorre no procedimento `Proc3` quando este percorre a $ListaP$ em busca de um simétrico para cada elemento da $ListaN$. Ambas as listas possuem, no pior caso, cardinalidade $N^2/4$ para N par e $(N^2-1)/4$ para N ímpar. □

A bijeção β gerada pelo algoritmo proposto é suficiente para afirmar que Z_{ρ_1} e Z_{ρ_2} **são** ou **não** livremente comparáveis, como prova o próximo teorema. Por essa razão a chamamos de **Bijeção Canônica** β_c.

Teorema 3.2 *Seja* β_c *a Bijeção Canônica resultante do algoritmo* `AlgBijeção`*. Se esta* β_c *não satisfizer o TOPL, caso exista outra bijeção* β' *que satisfaça a propriedade 3.2,* β' *também não satisfará o TOPL.*

Prova: Considere T o conjunto dos índices t em ordem crescente das parcelas não-negativas, cujos fatores-diferença sejam iguais e T' o conjunto dos índices t' em ordem crescente relativos às respectivas parcelas simétricas não-positivas. Pelo lema 3.1, temos $|T| = |T'|$.

Considere a restrição de β_c ao conjunto T, tal que $\beta_c|_T(N_t^i) = P_{t'}^j$ com $t \in T$ e $t' \in T'$.

Suponhamos que todos os pares induzidos $(t, t') \in ListaPares$ construídos pelo procedimento `Proc3` tenham $t' < t$ exceto para um par (t_m, t'_m) com $t_m \in T$ e $t'_m \in T'$. Neste caso, $t_m < t'_m$. Pode-se fazer uma nova combinação na tentativa de construir outra bijeção β', cuja restrição ao conjunto T seja tal que $t' < t$ para todos os pares induzidos (t, t').

Estando T e T' ordenados temos,

$$t_1 < t_2 < \ldots < t_{m-1} < t_m < t_{m+1} < \ldots < t_{|T|} \text{ e}$$

$$t'_1 < t'_2 < \ldots < t'_{m-1} < t'_m < t'_{m+1} < \ldots < t'_{|T'|}.$$

Se fizermos a troca de t'_m por t'_{m-1} teremos os novos pares induzidos (t_m, t'_{m-1}) e (t_{m-1}, t'_m). Dado que os elementos estão ordenados, tem-se $t_{m-1} < t_m$. Por hipótese, $t_m < t'_m$ e por transitividade, o novo par (t_{m-1}, t'_m) possui $t_{m-1} < t'_m$. O mesmo ocorreria se t'_m fosse trocado com algum elemento t'_s, para $s = 1, \ldots, m-1$, ou com t'_s para $s = m+1, \ldots, |T'|$.

Portanto, não é possível construir bijeção β' que induza pares tal que $t' < t$. □

As permutações que estão no exemplo 3.1 **não formam Z_{ρ_1} e Z_{ρ_2} livremente comparáveis**. Pode-se chegar a essa conclusão, observando-se a quarta coluna da tabela 3.1. A $ListaPares$ dos pares (t, t') induzidos pela β é {(1,2), (1,2), (1,3), (1,3), (4,5), (4,5), **(4,3)**}. Note que o par em negrito não permite que o lema 3.3 seja satisfeito.

A utilização das listas $ListaN$, $ListaP$ e $ListaPares$ foi um recurso computacional para se evitar os cálculos das parcelas não-positivas e não-negativas. A propriedade 3.2 facilita a verificação do sinal da expressão (3.5) da partição canônica de $Z_{\rho_1} - Z_{\rho_2}$ pelos fatores $(f_t^- - f_{t'}^-)$, observando-se os índices do vetor ordenado $\mathbf{F}^-$.

Capítulo 4

O Teorema das Inversões

Introduzimos os grafos direcionados G_{Inv} e G'_{Inv} que obedecem a ordem de inclusão $G_{Inv} \subset G'_{Inv} \subset G_{Liv}$, onde o maior deles caracteriza todas as permutações livremente comparáveis. Na seção 4.1 determinamos propriedades em G_{Inv} válidas também para G'_{Inv}, que nos permite provar a Conjectura das Inversões, enunciada desde 1984 em [Ab84]. Tal conjectura sugere uma relação entre número de inversões de permutações e seus custos associados, ambos representando vértices desses grafos. A prova deste resultado abre a possibilidade de estudar as soluções do Problema Quadrático de Alocação, não somente pela qualidade dos seus custos, como também pelo número de inversões das suas permutações a eles associadas.

4.1 O grafo G_{Inv}

Considere a relação de ordem "$\rho_s < \rho_r$, se ρ_r é obtida de ρ_s por troca de um par de elementos adjacentes e $\rho_s(i) < \rho_r(i)$", lembrando que $\rho_s(i+1) = \rho_r(i)$. A relação transitiva resultante de " $<$ " estabelece o *poset* $(\Pi_N, <)$, representado pelo grafo de comparabilidade G_{Inv}, que aqui chamamos de Grafo das Inversões. Nesta seção provamos que G_{Inv} é um grafo parcial de G_{Liv}, que caracteriza a ordenação parcial livre das permutações, definido na seção 3.1.3.

Uma *inversão* em $\rho \in \Pi_N$ se caracteriza pelo par $(\rho(i), \rho(j))$ tal que $\rho(j) < \rho(i)$, com $i < j,\ i, j = 1, \ldots, N$, que chamamos de par de inversões da permutação ρ. Considere $\Im(\rho)$ o conjunto dos pares das inversões da permutação ρ. O *número de inversões* de ρ, notado por $|\Im(\rho)|$, é dado pela cardinalidade de $\Im(\rho)$.

Exemplo 4.1 *Considere a permutação em* Π_6 *dada por* $\rho = (2\ \ 4\ \ 3\ \ 5\ \ 1\ \ 6) \in \Pi_6$.

O conjunto dos pares de inversões é $\Im(\rho) = \{(2,1),(4,3),(4,1),(3,1),(5,1)\}$, cuja cardinalidade $|\Im(\rho)| = 5$, que é o número de inversões de ρ.

O grafo $G_{Inv} = (\Pi_N, W)$, onde $W = \{(\rho_r, \rho_s) \in \Pi_N \times \Pi_N / \rho_s$ é obtida de ρ_r por troca de pares de elementos adjacentes com $|\Im(\rho_s)| - |\Im(\rho_r)| = 1\}$ é dito ser o **Grafo das Inversões**. Seu conjunto de vértices pode ser particionado em níveis pelo número de inversões de suas permutações, de modo que aquelas com o mesmo número de inversões pertençam ao mesmo nível. O número de nós (permutações) de G_{Inv} é $N!$.

Cabe observar que o grafo G_{Inv} aqui definido é o diagrama de Hasse do Reticulado das Permutações, conhecido na literatura por Permutaedro, Vernet et al. [VRA95], e que notamos por $\mathcal{P}_N$

Inicia-se a construção do grafo direcionado G_{Inv} pela permutação identidade $id = (1, 2, 3, \ldots, N)$ cujo nível é 0. Fazendo-se as trocas dos elementos adjacentes, nível a nível, chega-se à permutação reversa, $rev = (N, N-1, N-2, \ldots, 1)$, última gerada e pertencente ao último nível do grafo $\frac{N(N-1)}{2} + 1$.

O procedimento `ConstGNi`, a seguir apresentado, constrói o grafo G_{Inv} por níveis.

ConstGNi(N,id**);**

```
1   filho[0] = id;
2   ListaFilhos ← filho[0];
3   ultfilho = N!;
4   k = 1;
5   Enquanto k ≤ (ultfilho-1) faça
6      Para i = 1,...,(N-1) faça
7         gerar filho[k] trocando os elementos adjacentes i e i+1
            do filho[k-1];
8         Se filho[k] não está em ListaFilhos então
9            ListaFilhos ← filho[k];
10            calcular numinv[k];
11            k = k + 1;
12         FimSe;
13      FimPara;
14   FimEnquanto;
```

FimConstGNi;

Este procedimento tem complexiade $O(N!)$, o que torna inviável a construção de G_{Inv} para N grande.

Lema 4.1 *Sejam ρ_1 e ρ_2 em Π_N. Se $(\rho_1, \rho_2) \in W$ então $Z_{\rho_1} \leq_l Z_{\rho_2}$.*

Prova: Considere ρ_1 e ρ_2 em Π_N tal que $(\rho_1, \rho_2) \in W$. Tem-se então que $\rho_1 = (\rho_1(1), \ldots, \rho_1(i), \rho_1(i+1), \ldots, \rho_1(N))$ e $\rho_2 = (\rho_1(1), \ldots, \rho_1(i+1), \rho_1(i), \ldots, \rho_1(N))$, $i = 1, \ldots, (N-1)$ e $\rho_1(i) < \rho_1(i+1)$.

Aplicando-se o `Proc1` da seção 3.2 em ρ_1 e ρ_2, temos como resultado os conjuntos $P_N = \{i\}$, $P_P = \{i+1\}$ e $P_0 = \{1, \ldots, i-1, i+2, \ldots, N\}$. Da propriedade 3.2, a partição canônica é dada por

$$Z_{\rho_1} - Z_{\rho_2} = (f_i^- - f_{i+1}^-)(d^+_{\rho_1(i)} - d^+_{\rho_1(i+1)})$$

Com $\mathbf{D}^+$ em ordem não-decrescente $(d^+_{\rho_1(i)} - d^+_{\rho_1(i+1)}) \leq 0$ e com $\mathbf{F}^-$ não-crescente, $(f_i^- - f_{i+1}^-) \geq 0$. Assim, temos $Z_{\rho_1} \leq_l Z_{\rho_2}$. □

A figura (4.1) mostra o grafo G_{Inv} para $N = 4$. Observe que neste caso, o grafo não representa uma instância do problema quadrático PQA pois não existe $n \in \mathbb{N}$ tal que $C_{n,2} = 4$. Se considerássemos $n = 4$ teríamos $N = 6$ e conseqüentemente, o grafo G_{Inv} contém 720 nós, tornando difícil a representação gráfica.

Lema 4.2 *O grafo de comparabilidade do poset $(\Pi_N, <)$ é G_{Inv}, um grafo parcial do grafo de comparabilidade G_{Liv}, que define o poset $(\Pi_N, \leq_l)$ e representa as permutações livremente comparáveis.*

Prova: Segue direto do lema 4.1. □

O teorema que será enunciado é bastante importante pois aumenta o número de permutações livremente comparáveis representadas até o momento por G_{Inv}, gerando um grafo denotado $\widehat{G_{Inv}}$, que caminha para o grafo G_{Liv}, na sequência de inclusão $G_{Inv} \subset \widehat{G_{Inv}} \subset G_{Liv}$.

Teorema 4.1 *Sejam ρ_i e ρ_j vértices de G_{Inv} ligados por um caminho composto de arcos em W. Tem-se $|\Im(\rho_i)| < |\Im(\rho_j)|$, se e somente se, $Z_{\rho_i} \leq_l Z_{\rho_j}$.*

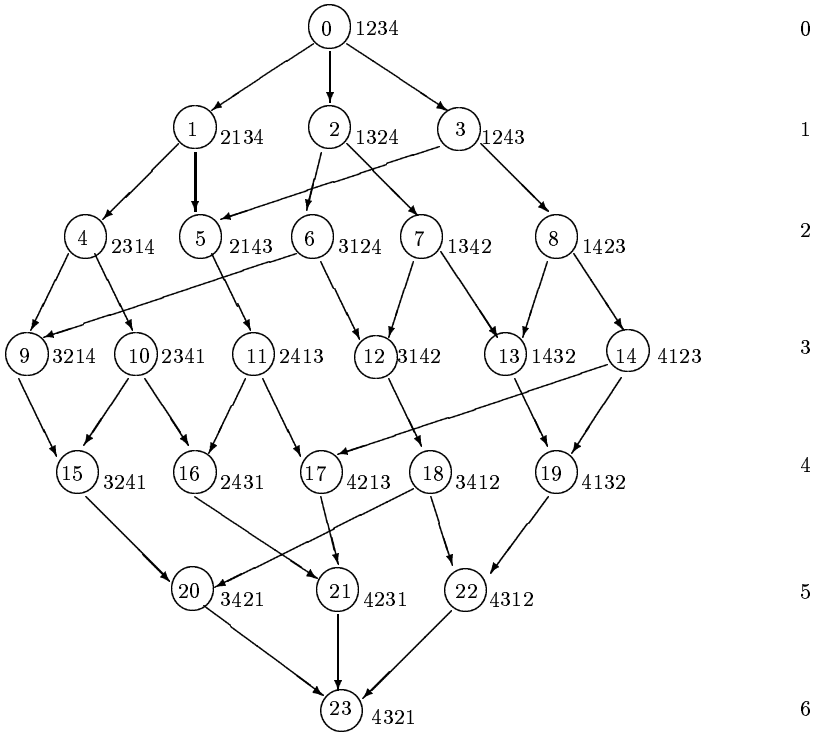


Figura 4.1: Grafo das Inversões G_{Inv}, $N = 4$ e níveis de 0 a 6

Prova: A demonstração é feita por indução sobre número de arcos do caminho μ entre ρ_i e ρ_j.

Vamos supor, inicialmente, que o caminho entre ρ_i e ρ_j seja composto por apenas um arco. Desta forma, $(\rho_i, \rho_j) \in W$ e da construção de G_{Inv} tem-se $|\Im(\rho_j)| - |\Im(\rho_i)| = 1$, implicando diretamente que $|\Im(\rho_i)| < |\Im(\rho_j)|$. Do lema 4.1, segue que $Z_{\rho_i} \leq_l Z_{\rho_j}$. Logo,

$$|\Im(\rho_i)| < |\Im(\rho_j)|, \quad \text{se e somente se,} \quad Z_{\rho_i} \leq_l Z_{\rho_j}. \tag{4.1}$$

Suponhamos que a hipótese seja válida para todos os caminhos de comprimento $(m-1)$ de ρ_i até ρ_j, composto de arcos em W.

Para facilitar a notação, tomemos $\rho_i = \rho_1$ e $\rho_j = \rho_{m+1}$, dado que desejamos que o caminho contenha m arcos. Então

$$\mu = \{(\rho_1, \rho_2), (\rho_2, \rho_3), \ldots, (\rho_m, \rho_{m+1})\}.$$

O caminho de ρ_1 até ρ_m tem comprimento $(m-1)$. Da hipótese de indução,

$$|\Im(\rho_1)| < |\Im(\rho_m)|, \text{ se e somente se, } Z_{\rho_1} \leq_l Z_{\rho_m}. \tag{4.2}$$

Tomemos o *m-ésimo* arco do caminho de W. Sendo $(\rho_m, \rho_{m+1}) \in W$ por construção de G_{Inv} tem-se $|\Im(\rho_m)| < |\Im(\rho_{m+1})|$. Do lema 4.1, segue que $Z_{\rho_1} \leq_l Z_{\rho_{m+1}}$. Logo,

$$|\Im(\rho_m)| < |\Im(\rho_{m+1})|, \text{ se e somente se, } Z_{\rho_m} \leq_l Z_{\rho_{m+1}}. \tag{4.3}$$

De (4.2) e (4.3) e por transitividade, segue-se que $|\Im(\rho_1)| < |\Im(\rho_{m+1})|$, se e somente se, $Z_{\rho_1} \leq_l Z_{\rho_{m+1}}$. Voltando a notação inicial, $|\Im(\rho_i)| < |\Im(\rho_j)|$, se e somente se, $Z_{\rho_1} \leq_l Z_{\rho_j}$. □

Notamos por $\widehat{G_{Inv}}$ o fecho transitivo de $G_{Inv} = (\Pi_N, W)$ cujo conjunto de arcos é determinado pela união de W com os demais arcos que resultam da transitividade, isto é, se existe um caminho entre ρ_i e ρ_j então existirá o arco (ρ_i, ρ_j). Temos que $\widehat{G_{Inv}}$ é o Reticulado das Permutações em Π_N cuja redução transitiva, G_{Inv}, é o permutaedro $\mathcal{P}_N$ de ordem N.

Corolário 4.1 *Sejam ρ_i e ρ_j permutações do grafo $\widehat{G_{Inv}}$. Se $Z_{\rho_i} \leq_l Z_{\rho_j}$ então $|\Im(\rho_i)| < |\Im(\rho_j)|$.*

Prova: Imediata do teorema 4.1. □

4.2 O grafo G'_{Inv}

Pretende-se estender o teorema 4.1 ao grafo G_{Liv}, grafo de comparabilidade do *poset* $(\Pi_N, \leq_l)$, pois este relaciona todas as permutações livremente comparáveis. Para isso, é necessário definir G'_{Inv}, grafo compreendido entre G_{Inv} e G_{Liv} ou seja, $G_{Inv} \subset G'_{Inv} \subset G_{Liv}$.

Considere o conjunto $W' = \{(\rho_r, \rho_s) \in \Pi_N \times \Pi_N / Z_{\rho_r} \leq_l Z_{\rho_s}$ ou $Z_{\rho_s} \leq_l Z_{\rho_r}$ e $|\Im(\rho_s)| - |\Im(\rho_r)| = 1\}$, cujos arcos resultam da relação advinda do Teorema da Ordenação Parcial Livre. Este conjunto é uma extensão do conjunto de arcos de G_{Inv}. Assim temos o grafo $G'_{Inv} = (\Pi_N, W')$, que como consequência, tem o grafo G_{Inv} como seu grafo parcial, dado que $W \subset W'$ e ambos possuem o mesmo conjunto de nós.

O lema a seguir é muito importante pois caracteriza duas permutações livremente comparáveis situadas em níveis consecutivos.

Lema 4.3 *Sejam ρ_1 e ρ_2 duas permutações tais que $||\Im(\rho_1)| - |\Im(\rho_2)|| = 1$ então Z_{ρ_1} e Z_{ρ_2} são livremente comparáveis, se e somente se, ρ_1 e ρ_2 diferem entre si de 2 elementos.*

Prova: ($\Rightarrow$) Por contradição, suponhamos que as permutações não diferem de apenas 2 elementos. Para tanto, pode-se considerar os conjuntos $P_N = \{t_1, t_2\}$, $P_P = \{t'\}$ e $P_0 = \Omega^N - \{P_N \cup P_P\}$. Pelo lema 3.2 tem-se $t_1 < t_2$.

Sendo ρ_i e ρ_j permutações em Ω^N tem-se

$$\sum_{t \in \Omega^N} \rho_i(t) - \rho_j(t) = 0. \tag{4.4}$$

Por definição os elementos $t \in P_0$ obedecem à condição $\rho_i(t) - \rho_j(t) = 0$. A expressão (4.4) torna-se

$$\rho_i(t_1) - \rho_j(t_1) + \rho_i(t_2) - \rho_j(t_2) + \rho_i(t') - \rho_j(t') = 0. \tag{4.5}$$

Quanto a ordenação entre os elementos de P_N e P_P há três possibilidades: (a) $t_1 < t_2 < t'$; (b) $t' < t_1 < t_2$; e (c) $t_1 < t' < t_2$.

Considerando para análise o item (a), as combinações de valores que se anulam, aliadas às características dos conjuntos P_N e P_P, resultam dois casos, enumerados a seguir.

Caso 1:

$$\begin{array}{l} \rho_i(t_1) = \rho_j(t_2),\ \text{como } \rho_i(t_1) < \rho_j(t_1) \Rightarrow \rho_j(t_2) < \rho_j(t_1); \\ \rho_i(t_2) = \rho_j(t'),\ \text{como } \rho_i(t_2) < \rho_j(t_2) \Rightarrow \rho_j(t') < \rho_j(t_2); \\ \rho_i(t') = \rho_j(t_1),\ \text{como } \rho_i(t') > \rho_j(t') \Rightarrow \rho_j(t') < \rho_j(t_1). \end{array} \tag{4.6}$$

Caso 2:

$$\begin{array}{l} \rho_i(t_1) = \rho_j(t'),\ \text{como } \rho_i(t_1) < \rho_j(t_1) \Rightarrow \rho_j(t') < \rho_j(t_1); \\ \rho_i(t_2) = \rho_j(t_1),\ \text{como } \rho_i(t_2) < \rho_j(t_2) \Rightarrow \rho_j(t_1) < \rho_j(t_2); \\ \rho_i(t') = \rho_j(t_2),\ \text{como } \rho_i(t') > \rho_j(t') \Rightarrow \rho_j(t') < \rho_j(t_2). \end{array} \tag{4.7}$$

Para ilustrar, sejam as permutações ρ_i e ρ_j

$$\begin{array}{l} \ldots \rho_i(t_1) \ldots \rho_i(t_2) \ldots \rho_i(t') \ldots \\ \ldots \rho_j(t_1) \ldots \rho_j(t_2) \ldots \rho_j(t') \ldots \end{array} \tag{4.8}$$

Utilizando as relações em (4.6) nas permutações ilustradas em (4.8) e, em seguida, (4.7) nas mesmas permutações, tem-se $|\Im(\rho_j)| - |\Im(\rho_i)| \leq 2$. Contradição, pois a diferença, por hipótese, deve ser de uma unidade.

Para o item (b) o raciocínio é análogo e, novamente, tem-se uma contradição com relação ao número de inversões, isto é, $|\Im(\rho_i)| - |\Im(\rho_j)| \leq 2$, porém em sentido contrário pois $t' < t_1 < t_2$.

Tomando agora o último caso (c), onde $t_1 < t' < t_2$, seja a diferença entre os produtos escalares $Z_{\rho_i} - Z_{\rho_j}$ expressa por

$$Z_{\rho_i} - Z_{\rho_j} = f^-_{t_1}(d^+_{\rho_i(t_1)} - d^+_{\rho_j(t_1)}) + f^-_{t_2}(d^+_{\rho_i(t_2)} - d^+_{\rho_j(t_2)}) + f^-_{t'}(d^+_{\rho_i(t')} - d^+_{\rho_j(t')}). \quad (4.9)$$

Substituindo as igualdades (4.6) em (4.9), tem-se:

$$\begin{aligned} Z_{\rho_1} - Z_{\rho_2} &= f^-_{t_1}(d^+_{\rho_i(t_1)} - d^+_{\rho_i(t')}) + f^-_{t_2}(d^+_{\rho_i(t_2)} - d^+_{\rho_i(t_1)}) + f^-_{t'}(d^+_{\rho_i(t')} - d^+_{\rho_i(t_2)}) \\ &= d^+_{\rho_i(t_1)}(f^-_{t_1} - f^-_{t_2}) + d^+_{\rho_i(t_2)}(f^-_{t_2} - f^-_{t'}) + d^+_{\rho_i(t')}(f^-_{t'} - f^-_{t_1}). \end{aligned} \quad (4.10)$$

Apesar do vetor $\mathbf{F}^-$ estar em ordem não-crescente, não há como afirmar que $Z_{\rho_i} - Z_{\rho_j} \leq 0$ ou $Z_{\rho_j} - Z_{\rho_i} \leq 0$, sem o conhecimento prévio dos valores das componentes de $\mathbf{F}^-$.

Quando substituimos as igualdades (4.7) na equação (4.10), o raciocínio é análogo e chegamos a mesma contradição. Sendo assim, quando $t_1 < t' < t_2$ os produtos não são livremente comparáveis.

Portanto, os conjuntos P_N e P_P devem ser unitários.

($\Leftarrow$) Tem-se a expressão (3.2) do capítulo 3

$$Z_{\rho_1} - Z_{\rho_2} = \sum_{t \in P_N} f^-_t(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) + \sum_{t' \in P_P} f^-_{t'}(d^+_{\rho_1(t')} - d^+_{\rho_2(t')}).$$

As permutações diferem de 2 elementos resultando os conjuntos P_N e P_P unitários e imediatamente tem-se $\rho_1(t) = \rho_2(t')$ e $\rho_2(t) = \rho_1(t')$. Aplicando a propriedade 3.2, a expressão pode ser reescrita como

$$Z_{\rho_1} - Z_{\rho_2} = (f^-_t - f^-_{t'})(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}). \quad (4.11)$$

Sabendo-se que os vetores $\mathbf{D}^+$ e $\mathbf{F}^-$ são ordenados, temos $(d^+_{\rho_1(t)} - d^+_{\rho_2(t)}) \leq 0$ e para $t < t' \Rightarrow Z_{\rho_1} \leq_l Z_{\rho_2}$. Analogamente, $t' < t \Rightarrow Z_{\rho_2} \leq_l Z_{\rho_1}$. Conclui-se que ρ_1 e ρ_2 são livremente comparáveis. □

Para exemplificar um dos itens da demonstração, sejam $\rho_i = (2\ 3\ 4\ 1\ 6\ 5)$ e $\rho_j = (2\ 5\ 4\ 3\ 6\ 1)$. Estas permutações possuem $t_1 < t_2 < t'$ e obedecem as relações em (4.6). Tem-se $|\Im(\rho_j)| - |\Im(\rho_i)| = 3$.

O lema 4.4 a seguir, permite definir o grafo G'_{Inv} desconsiderando-se uma das desigualdades em W', ou seja, $Z_{\rho_1} \geq_l Z_{\rho_2}$.

Lema 4.4 *Sejam ρ_1 e ρ_2 em Π_N. Se $(\rho_1, \rho_2) \in W'$ então $Z_{\rho_1} \leq_l Z_{\rho_2}$.*

Prova: Se $(\rho_1, \rho_2) \in W \subset W'$, do lema 4.1, $Z_{\rho_1} \leq_l Z_{\rho_2}$.

Consideremos $(\rho_1, \rho_2) \in (W' - W) \Rightarrow |\Im(\rho_2)| - |\Im(\rho_1)| = 1$.

Suponhamos por contradição que $Z_{\rho_2} \leq_l Z_{\rho_1}$. Daí, $\exists\ \beta_c$ tal que $\beta_c(N_t^i) = P_{t'}^j$ com $t \in P_N$; $t' \in P_P$; $i = 0, \ldots, \rho_2(t) - \rho_1(t) - 1$ e $j = 0, \ldots, \rho_1(t') - \rho_2(t') - 1$, que induz pares ordenados (t, t') com $t' < t\ \forall t \in P_N$ e $\forall t' \in P_P$. Pelo lema 4.3, os conjuntos $P_N = \{t\}$ e $P_P = \{t'\}$ são unitários, em outras palavras, as permutações diferem de apenas 2 elementos. Sejam tais permutações P_N e P_P

$$\begin{array}{l} \ldots \rho_1(t') \ldots \rho_1(t) \ldots \\ \ldots \rho_2(t') \ldots \rho_2(t) \ldots \end{array} \tag{4.12}$$

e por definição dos conjuntos segue-se

$$\rho_1(t') > \rho_2(t') \text{ e } \rho_1(t) < \rho_2(t). \tag{4.13}$$

Como essas permutações diferem apenas de 2 elementos, é fácil ver que

$$\begin{array}{l} \rho_1(t') = \rho_2(t) > \rho_1(t) \\ \rho_2(t') = \rho_1(t) < \rho_2(t). \end{array} \tag{4.14}$$

Quando o número de inversões é calculado, tem-se que $|\Im(\rho_1)| > |\Im(\rho_2)|$. Contradição pois temos a hipótese $|\Im(\rho_2)| - |\Im(\rho_1)| = 1$. □

A figura (4.2) mostra o grafo G'_{Inv} para $N = 4$. Os arcos que pertencem ao conjunto $(W' - W)$ estão em negrito.

O lema 4.4 é a base da demonstração da Conjectura das Inversões proposta por Abreu [Ab84] em sua tese de doutorado em 1984. Sua conjectura sugere que a ordenação dos custos correspondentes às **permutações livremente comparáveis** é compatível com a ordenação do número de inversões dessas permutações. Isto significa que custos de permutações livremente comparáveis "crescem" ou "decrescem" no mesmo sentido que os números de inversões dessas permutações. Provamos essa conjectura no teorema 4.3 que chamamos de **Teorema das Inversões**. Para isso, precisamos enunciar o teorema 4.2 cuja prova é análoga ao teorema 4.1, acrescida

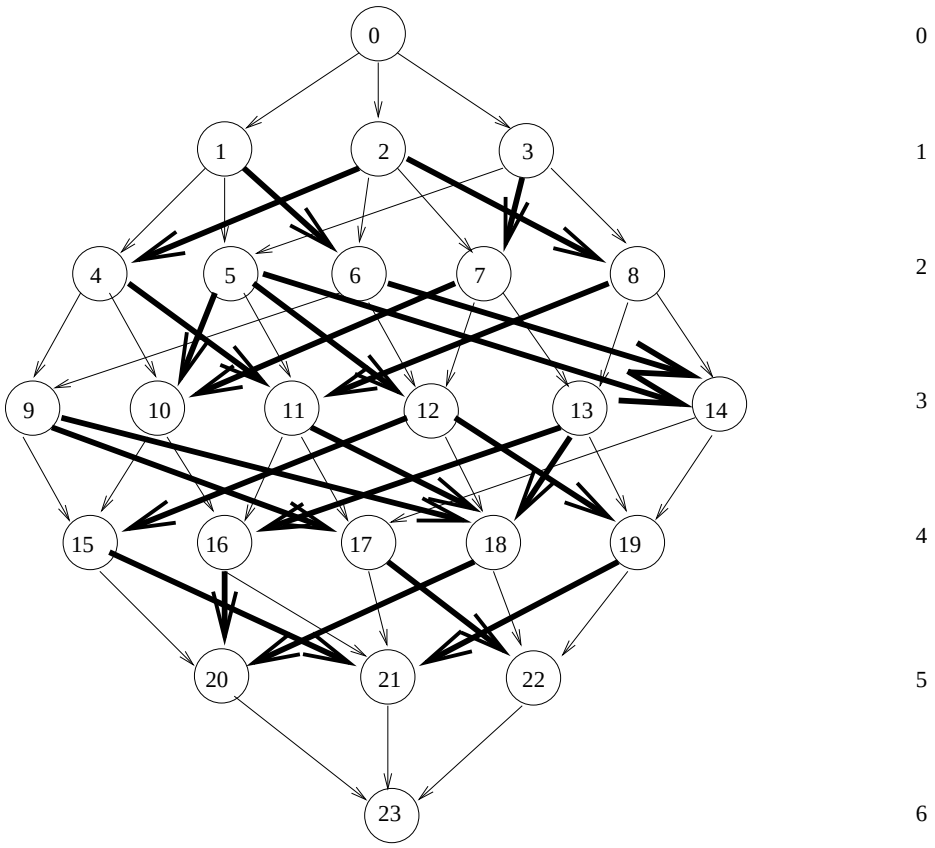


Figura 4.2: Grafo G'_{Inv}, $N = 4$ e destaque para os arcos de $(W' - W)$

apenas do lema 4.4. Isto ocorre pois o teorema 4.1 se refere ao grafo G_{Inv}, enquanto que o próximo se refere ao grafo G'_{Inv}, que neste caso tem seus arcos em W' que contém os arcos de W.

Teorema 4.2 *Sejam ρ_i e ρ_j vértices de G'_{Inv} ligados por um caminho de composto de arcos em W'. Tem-se $|\Im(\rho_i)| < |\Im(\rho_j)|$, se e somente se, $Z_{\rho_i} \leq_l Z_{\rho_j}$.*

Prova: A demonstração é feita por indução sobre número de arcos do caminho μ entre ρ_i e ρ_j.

Vamos supor, inicialmente, que o caminho entre ρ_i e ρ_j seja composto de apenas um arco. Logo $(\rho_i, \rho_j) \in W'$ e da construção de G'_{Inv} $|\Im(\rho_j)| - |\Im(\rho_i)| = 1$, implicando diretamente que $|\Im(\rho_i)| < |\Im(\rho_j)|$. Do lema 4.4, tem-se que $Z_{\rho_i} \leq_l Z_{\rho_j}$. Logo,

$$|\Im(\rho_i)| < |\Im(\rho_j)|, \text{ se e somente se, } Z_{\rho_i} \leq_l Z_{\rho_j}. \tag{4.15}$$

Suponhamos que a hipótese seja válida para todos os caminhos de comprimento $(m-1)$ de ρ_i até ρ_j, composto de arcos em W'.

Para facilitar a notação, tomemos $\rho_i = \rho_1$ e $\rho_j = \rho_{m+1}$, dado que desejamos que o caminho contenha m arcos. Então

$$\mu = \{(\rho_1, \rho_2), (\rho_2, \rho_3), \ldots, (\rho_m, \rho_{m+1})\}.$$

O caminho de ρ_1 até ρ_m tem comprimento $(m-1)$. Da hipótese de indução,

$$|\Im(\rho_1)| < |\Im(\rho_m)|, \text{ se e somente se, } Z_{\rho_1} \leq_l Z_{\rho_m}. \tag{4.16}$$

Tomemos o *m-ésimo* arco do caminho de W'. Sendo $(\rho_m, \rho_{m+1}) \in W'$ por construção de G'_{Inv} tem-se $|\Im(\rho_m)| < |\Im(\rho_{m+1})|$. Do lema 4.4, segue que $Z_{\rho_1} \leq_l Z_{\rho_{m+1}}$. Logo,

$$|\Im(\rho_m)| < |\Im(\rho_{m+1})|, \text{ se e somente se, } Z_{\rho_m} \leq_l Z_{\rho_{m+1}}. \tag{4.17}$$

De (4.16) e (4.17) e por transitividade, segue-se que $|\Im(\rho_1)| < |\Im(\rho_{m+1})|$, se e somente se, $Z_{\rho_1} \leq_l Z_{\rho_{m+1}}$. Voltando a notação inicial, $|\Im(\rho_i)| < |\Im(\rho_j)|$, se e somente se, $Z_{\rho_1} \leq_l Z_{\rho_j}$. □

Corolário 4.2 *Sejam ρ_i e ρ_j permutações do grafo $\widehat{G'_{Inv}}$. Se $Z_{\rho_i} \leq_l Z_{\rho_j}$ então $|\Im(\rho_i)| < |\Im(\rho_j)|$.*

Prova: Imediata do teorema 4.2. □

O grafo $\widehat{G'_{Inv}}$, fecho transitivo de $G'_{Inv} = (\Pi_N, W')$, tem seu conjunto de arcos determinado pela união de W' com aqueles que resultam da transitividade, isto é, se existe um caminho entre ρ_i e ρ_j então existirá o arco (ρ_i, ρ_j). O **Teorema das Inversões** a seguir apresentado, mostra que todo arco do grafo G_{Liv} é arco do fecho transitivo $\widehat{G'_{Inv}}$. O grafo G_{Liv} é então o grafo de comparabilidade do *poset* $(\Pi_N, \leq_l)$ que envolve todas a permutações livremente comparáveis. Desta forma, tem-se que pares de permutações livremente comparáveis, situadas em níveis distintos do grafo G_{Liv}, o custo da permutação que estiver no nível mais baixo, é obrigatoriamente menor, em outras palavras, *custos de permutações livremente comparáveis "crescem" ou "decrescem" quando os números de inversões dessas permutações também "crescerem" ou "decrescerem".*

Teorema 4.3 (Teorema das Inversões) *Sendo o grafo $\widehat{G'_{Inv}}$ o fecho transitivo de $G'_{Inv} = (\Pi_N, W')$ e $G_{Liv} = (\Pi_N, M)$, grafo resultante do TOPL, tem-se que $\widehat{G'_{Inv}} = G_{Liv}$.*

Prova: ($\Rightarrow$) $\widehat{G'_{Inv}}$ é um subgrafo de G_{Liv} pois possui o mesmo conjunto de nós e do teorema 4.2 tem-se que $W' \subseteq M$, onde $M = \{(\rho_i, \rho_j)/Z_{\rho_i} \leq_l Z_{\rho_j},\ i, j = 1, \ldots, N!\}$.

($\Leftarrow$) Queremos provar que para dois nós quaisquer ρ_i e $\rho_j \in \Pi_N$ com $Z_{\rho_i} \leq_l Z_{\rho_j}$ e $|\Im(\rho_j)| - |\Im(\rho_i)| = m$ deve existir um caminho entre eles com m arcos em W'.

Suponamos que não exista um caminho μ entre ρ_i e ρ_j, $(\rho_i, \rho_j) \in M$, composto de m arcos pertencentes ao conjunto W', ou seja, pelo menos um arco, $(\rho''_i, \rho''_j) \notin W'$, como mostra a figura 4.3.

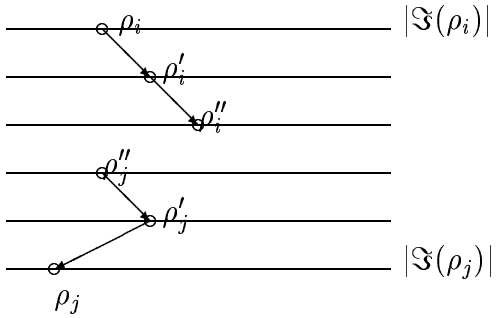


Figura 4.3: Caminho interrompido entre ρ''_i e ρ''_j.

Sejam os caminhos parciais dados por $\mu_1 = (\rho_i, \rho'_i, \rho''_i)$ e $\mu_2 = (\rho''_j, \rho'_j, \rho_j)$, tais que $(\rho_i, \rho'_i) \in W'$; $(\rho'_i, \rho''_i) \in W'$ e $(\rho''_j, \rho'_j) \in W'$; $(\rho'_j, \rho_j) \in W'$, de acordo com a figura 4.3. Segue-se do teorema 4.2, que

$$Z_{\rho_i} \leq_l Z_{\rho'_i} \leq_l Z_{\rho''_i} \Rightarrow Z_{\rho_i} - Z_{\rho''_i} \leq 0 \tag{4.18}$$

e

$$Z_{\rho''_j} \leq_l Z_{\rho'_j} \leq_l Z_{\rho_j} \Rightarrow Z_{\rho''_j} - Z_{\rho_j} \leq 0. \tag{4.19}$$

Somando-se ambos os membros das desigualdades provenientes de (4.18) e (4.19) chegamos a

$$(Z_{\rho_i} - Z_{\rho''_i}) + (Z_{\rho''_j} - Z_{\rho_j}) \leq 0. \tag{4.20}$$

Como $(\rho''_i, \rho''_j) \notin W'$, tomemos uma instância PQA tal que $Z_{\rho''_i} > Z_{\rho''_j}$ e então é possível encontrar um $K > 0$, suficientemente grande, tal que $Z_{\rho''_i} - Z_{\rho'_j} = K$ e que somando este valor a (4.20), se tenha

$$(Z_{\rho_i} - Z_{\rho''_i}) + (Z_{\rho''_i} - Z_{\rho''_j}) + (Z_{\rho''_j} - Z_{\rho_j}) \geq 0. \tag{4.21}$$

Mas (4.21) é igual a $Z_{\rho_i} - Z_{\rho_j} \geq 0$, contrariando o fato de $(\rho_i, \rho_j) \in M$. □

4.2.1 Algoritmo para um caminho entre permutações livremente comparáveis

O algoritmo `AlgPath` tem por objetivo encontrar um caminho μ entre duas permutações livremente comparáveis. Para isso, necessitamos de algumas definições, apresentadas a seguir.

Chamamos de uma **troca admissível** de dois elementos de uma permutação ρ^p aquela que gera uma nova permutação ρ^{p+1}, tal que $|\Im(\rho^{p+1})| - |\Im(\rho^p)| = 1$. Neste caso, tem-se $(\rho^p, \rho^{p+1}) \in W'$.

Seja $m = |\Im(\rho_j)| - |\Im(\rho_i)|$ o número de arcos do caminho μ. Defina uma lista $L = (\rho^p(k), \rho_j^{-1}(\rho_i(k)))$ com $k = 1, \ldots, N$ e $p = 0, \ldots, m$, onde $\rho_j^{-1}(\rho_i(k))$ representa a posição que o elemento $\rho_i(k)$ ocupa na permutação ρ_j. O objetivo do procedimento `AlgPath` é ordenar as posições $\rho_j^{-1}(\rho_i(k))$, fazendo m trocas admissíveis, a partir da permutação $\rho^0(k)$, lembrando que $\rho^0 = \rho_i$.

AlgPath(ρ_i, ρ_j);

1 $p = 0$;
2 $m = |\Im(\rho_j)| - |\Im(\rho_i)|$;
3 $\rho^0 = \rho_i$;
4 `Para` $k = 1, \ldots, N$ `faça`
5 $L \leftarrow (\rho^p(k), \rho_j^{-1}(\rho^0(k)))$;
6 `FimPara;`
7 `Para` $k = 1, \ldots, N-1$ `faça`
8 `t = k+1;`
9 `Repetir`
10 `Se` $(k \neq \rho_j^{-1}(\rho^p(k))$ `e` $(t \neq \rho_j^{-1}(\rho^p(t)))$
11 `verificar se posição` t `permite troca admissível;`
12 `Se` *`a troca é admissível`* `então`
13 $aux = L(k)$;
14 $L(k) = L(t)$;
15 $L(t) = aux$;
16 $p = p + 1$;
17 `FimSe;`
18 $t = t + 1$;
19 `FimSe;`

20 Até que $t = N$;

21 Se *p não alterou seu valor* então

22 voltar à linha 9 relaxando a condição $t \neq \rho_j^{-1}(\rho^p(t))$

23 FimPara;

FimAlgPath;

A linha 10 do `AlgPath` evita que as posições de ρ^p que já alcançaram a posição desejada na permutação ρ_j sejam alteradas. A linha 11 verifica se a posição $k < t$ permite uma troca admissível. Nesse passo, é prioridade não alterar posições que, inicialmente possuam $t = \rho_j^{-1}(\rho^p(t))$. Porém, há casos em que essa condição deve ser violada senão o algoritmo não consegue estabelecer uma ligação entre dois níveis consecutivos. O algoritmo verifica esse fato na linha 20. Quando isso acontece, deve-se voltar a linha 9 porém, com a segunda condição na linha 10, $t \neq \rho_j^{-1}(\rho^p(t))$, relaxada. As linhas 13 e 14 do algoritmo fazem a **troca admissível** nas posições da permutação $\rho^p(k)$ e $\rho^p(t)$ e consequentemente, na ordenação de $\rho_j^{-1}(\rho^p)$. Portanto, se ρ_i e ρ_j forem livremente comparáveis, alcança-se a ordenação natural $(1, 2, \ldots, N)$ em $\rho_j^{-1}(\rho^m)$.

A seguir, faremos o exemplo 4.2 da aplicação do `AlgPath`. Ao final devemos obter $\rho^m = \rho_2$ e $\rho_j^{-1}(\rho^m(k))$ em ordem crescente.

Exemplo 4.2 *Considere* $\rho_i = (1\ 2\ 6\ 5\ 3\ 4)$ *e* $\rho_j = (1\ 5\ 6\ 4\ 2\ 3)$ *permutações em* Π_6.

k	$\rho^0(k)$	$\rho_j^{-1}(\rho^0(k))$	$\rho^1(k)$	$\rho_j^{-1}(\rho^1(k))$	$\rho^2(k)$	$\rho_j^{-1}(\rho^2(k))$	$\rho^3(k)$	$\rho_j^{-1}(\rho^3(k))$
1	1	1	1	1	1	1	1	1
2	2	5	5	2	5	2	5	2
3	6	3	6	3	6	3	6	3
4	5	2	2	5	3	6	4	4
5	3	6	3	6	2	5	2	5
6	4	4	4	4	4	4	3	6

Tabela 4.1: Construção de $\mu = (\rho^0, \rho^1, \rho^2, \rho^3)$.

A figura 4.4 mostra o caminho percorrido pelo algoritmo. O nó 166 é o 166^o gerado pelo algoritmo **ConstGNi**, apresentado na seção 4.1, pertence ao nível 5, analogamente tem-se a rotulação dos demais nós do caminho.

O lema 4.3, garante que o algoritmo proposto `AlgPath` sempre encontra um caminho entre duas permutações livremente comparáveis. No caso das permutações

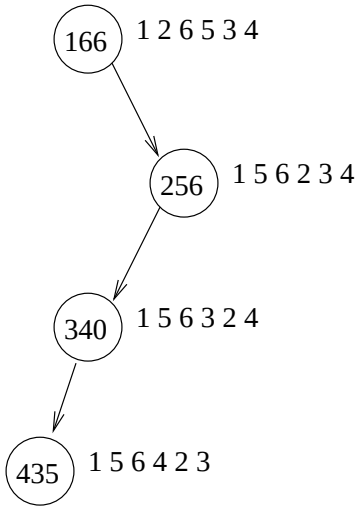


Figura 4.4: Caminho entre as permutações

não serem livremente comparáveis, o algoritmo não entra em *looping*, visto que as condições para terminar os comandos de repetição são sempre satisfeitas. Nestes casos, ao final, $\rho_j^{-1}(\rho^m)$ não possuirá seus elementos em ordem crescente.

O teorema 4.3 garante a existência de um caminho entre duas permutações livremente comparáveis, composto somente por arcos do conjunto W'. O lema 4.3 mostra que se o arco $(\rho_i, \rho_j) \in W'$, os conjuntos P_N e P_P são unitários. O algoritmo `AlgPath` trabalha com esses conjuntos unitários (linhas 7 e 8) e com trocas admissíveis de elementos $\rho(t)$ e $\rho(t')$ (linhas 13 e 14).

Capítulo 5

Heurísticas para o PQA

O Problema Quadrático de Alocação é considerado um dos problemas combinatórios mais difíceis, tanto do ponto de vista teórico, quanto do computacional. Ele pertence à classe *NP-Hard* e para tentar resolvê-lo muitos pesquisadores se dedicam a estudar heurísticas e técnicas meta-heurísticas que possam explorar características inerentes à sua natureza. Neste capítulo serão descritas algumas delas, tais como *GRASP*, Algoritmos Genéticos e *Simulated Annealing*, por terem sido muito aplicadas ao *PQA*. Com a introdução de informações algébricas determinadas nos capítulos anteriores nas heurísticas aqui apresentadas, desenvolvemos procedimentos dedicados a este problema cujas descrições se encontram no decorrer deste capítulo.

5.1 GRASP - Greedy Randomized Adaptive Search Procedures

Em linhas gerais, o GRASP - *Greedy Randomized Adaptive Search Procedures* 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 primeira, a construtiva, que determina a solução que será submetida à busca local e a segunda fase, cujo objetivo é obter alguma melhoria na solução corrente. A seguir, o algoritmo GRASP em pseudo-código é apresentado.

ProcGRASP( );

```
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);
```
FimProcGRASP;

Na maioria das aplicações, o critério de parada é baseado no número máximo de iterações previamente estabelecido. Outros critérios podem ser definidos, como parar quando a solução procurada for encontrada ou estabelecer um tempo máximo de execução.

Na primeira fase, 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 adicionado 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.

ConstSolInicGulosaAleatória($sol = \emptyset$);
```
1   Enquanto ''solução não estiver completa'' faça
2       ConsLRC(LRC);
3       s = SelecAleatElem(LRC);
4       sol ← sol ∪ {s};
5       FuncAdapGul(s);
6   FimEnquanto;
```
FimConstSolInicGulosaAleatória;

Uma vez obtida uma solução s, consulta-se a estrutura de vizinhança $Viz(s)$ relativa a 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 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.

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;
5   Retorna(s como localmente ótima);
```

FimBuscaLocal;

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, 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.

O algoritmo GRASP foi aplicado a diversos problemas: planejamento e escalonamento de produção, Laguna e González [LG91], problemas de partição em grafos, Laguna et al. [LFE93], de localização, Klincewicz [Kli92] e o Problema Quadrático de Alocação, Li et al. [LPR94],

5.2 Aplicação do GRASP ao PQA

Li, Pardalos e Resende [LPR94] submeteram 88 instâncias da biblioteca $QAPLIB$ [BKR97] ao GRASP. Esta metodologia encontrou as melhores soluções até então conhecidas, superando-as em alguns casos.

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

Quando aplicada ao PQA, a fase de construção do GRASP pode ser dividida em dois estágios: o primeiro, cria os dois elementos da permutação a ser gerada e o

segundo estágio constrói, passo a passo, os $(n-2)$ elementos restantes.

5.2.1 Fase de construção

O estágio 1 constrói a lista restrita de candidatos, LRC, para escolher os dois primeiros elementos de uma permutação. No estágio 2, completa-se as $(n-2)$ atribuições restantes, sempre escolhendo pares de facilidade-localidade, (i,k), visando uma menor contribuição para o custo desta solução inicial.

No estágio 1 são determinados os primeiros dois pares localidade-facilidade da solução inicial. A LRC é construída, considerando-se dois parâmetros α e β, $0 < \alpha < 1$ e $0 < \beta < 1$. As $(n^2 - n)$ entradas da matriz de distâncias D são ordenadas, em ordem não-decrescente, 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. Finalmente, os produtos das distâncias pelos fluxos são calculados, ordenados em ordem crescente, e considerados apenas os $\lfloor \alpha\beta(n^2 - n) \rfloor$ menores.

O procedimento **Estágio1** mostra os passos que devem ser seguidos. Chamaremos de Ω o conjunto de pares de localidade-facilidade correspondentes à solução inicial do problema.

Estágio1(D, F, α, β);
1 $\Omega = \emptyset$;
2 `ordenar em ordem não-decrescente as entradas de` D;
3 `ordenar em ordem não-crescente as entradas de` F;
4 `multiplicar os primeiros` $\lfloor \beta(n^2 - n) \rfloor$ `de` F `e` D:
5 `ordenar em ordem não-decrescente os produtos;`
6 $LRC \leftarrow \lfloor \alpha\beta(n^2 - n) \rfloor$ `primeiros produtos;`
7 `escolher aleatoriamente` $f_{ij}d_{kl}$ `em` LRC;
8 $\Omega \leftarrow \{(i,j),(k,l)\}$;
9 `Retorna(`Ω`);`
FimEstágio1;

Sendo constantes os dados de entrada, todo o processo de ordenação é realizado apenas uma vez e a ordem dos custos na LRC permanece inalterada. 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, acessando então, os índices do custo $f_{ij}d_{kl}$ escolhido, correspondentes aos pares de associações $\{(i,k),(j,l)\}$. Esses pares constituem parâmetros de entrada para o segundo estágio na construção da solução inicial.

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

$$c_{ik} = \sum_{(j,l)\in\Omega} f_{ij}d_{kl}. \tag{5.1}$$

Cada coeficiente c_{ik} corresponde ao custo de instalação da facilidade i na localidade k, considerando-se as últimas associações feitas. Seja m o número de possíveis pares (i,k) a serem ainda escolhidos e considere α o parâmetro citado anteriormente. A lista restrita de candidatos limita a busca de (i,k) aos $\lfloor \alpha m \rfloor$ pares de localidade-facilidade com menores custos c_{ik}. Uma vez determinado aleatoriamente o par (i,k), o conjunto Ω é atualizado para $\Omega \leftarrow \Omega \cup \{(i,k)\}$. Este estágio determina os $(n-2)$ pares de localidade-facilidade a partir dos obtidos no estágio 1. Esta solução construída 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:

Estágio2 $(\alpha,(j_1,l_1),(j_2,l_2))$;

1 $\Omega = \{(j_1,l_1),(j_2,l_2)\}$;
2 $Vetcik = \emptyset$;
3 Para associações = 3,...,n faça
4 $m = 0$;
5 Para $i = 1,\ldots,n$ faça
6 Para $k = 1,\ldots,n$ faça
7 Se $(i,k) \notin \Omega$ então
8 $c_{ik} = \sum_{(j,l)\in\Omega} f_{ij}d_{kl}$;
9 $Vetcik \leftarrow c_{ik}$;
10 $m = m + 1$;
11 FimSe;
12 FimPara;
13 FimPara;
14 ordenar em ordem não-decrescente as componentes de $Vetcik$;
15 escolher aleatoriamente s no intervalo $[1,\lfloor \alpha m \rfloor]$;
16 determinar o par (i,k) correspondente à posição s do vetor;

17 $\Omega \leftarrow \Omega \cup \{(i,k)\}$;

18 FimPara;

FimEstágio2;

5.2.2 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

$$\delta(\varphi_1, \varphi_2) = \{i | \varphi_1(i) \neq \varphi_2(j)\} \tag{5.2}$$

e a distância entre φ_1 e φ_2 é definida por

$$d(\varphi_1, \varphi_2) = |\delta(\varphi_1, \varphi_2)|. \tag{5.3}$$

A estrutura de vizinhança mais comumente usada chama-se **k-troca**. Uma vizinhança k-troca para uma permutação $\varphi \in \Pi_n$ é definida por:

$$Viz_k(\varphi) = \{\varphi' | d(\varphi, \varphi') \leq k\}, \ onde \ 2 \leq k \leq n. \tag{5.4}$$

Neste trabalho, os pesquisadores utilizaram a vizinhança 2-trocas. No exemplo, para $n = 4$ tem-se $|Viz(\varphi)| = C_{4,2} = 6$. Seja $\varphi = (3 \ \ 1 \ \ 4 \ \ 2)$, temos

$$Viz(\varphi) = \{(1\ 3\ 4\ 2), (4\ 1\ 3\ 2), (2\ 1\ 4\ 3), (3\ 4\ 1\ 2), (3\ 2\ 4\ 1), (3\ 1\ 2\ 4\ )\}.$$

Na estratégia de Busca em Largura e Profundidade constrói-se uma árvore por níveis. A partir de φ_0, gerada na primeira fase, é construída uma vizinhança com estrutura 2-trocas. 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-trocas sobre a melhor solução do nível 1) e novamente é escolhida 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 (Figura 5.1). Esta estratégia tem sido bastante utilizada e fornece resultados satisfatórios encontrados [LPR94].

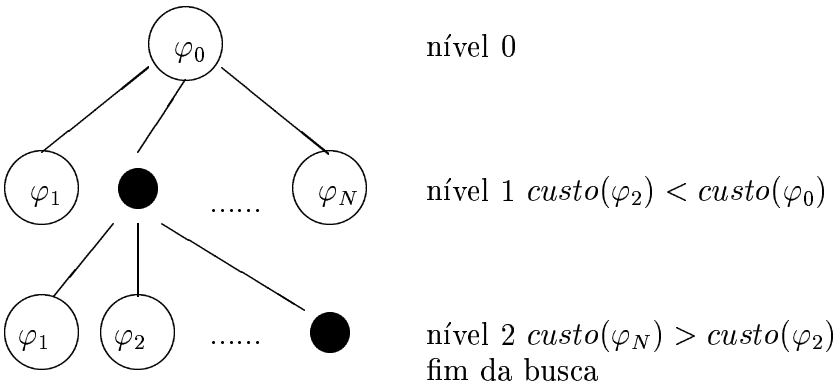


Figura 5.1: Busca em largura e profundidade

Nas seções seguintes apresentamos duas propostas de modificações GRASP que implementamos, na esperança de encontrarmos melhores soluções ou melhores tempos computacionais. Os resultados estão apresentados em 5.3.1 e 5.4.1.

5.3 Um Espaço de Busca Local Ampliado

Nesta seção é realizada uma modificação na heurística GRASP aplicada ao Problema Quadrático de Alocação desenvolvida por Li et al. [LPR94], com o objetivo de expandir o espaço de busca local. Para isso, nos baseamos nos resultados teóricos obtidos nos capítulos 3 e 4. Os principais resultados utilizados para desenvolver tais modificações no GRASP foram o Teorema da Ordenação Parcial Livre, teorema 3.1, e o Teorema das Inversões, teorema 4.3. O primeiro diz que pares de permutações $(\rho_1, \rho_2) \in G_{Liv}$ podem ter seus custos comparáveis independentemente dos valores dos fluxos e distâncias da instância considerada (seção 3.1.3), e o segundo, diz que em pares de permutações (ρ_1, ρ_2) livremente comparáveis, se ρ_1 possui o número de inversões maior que ρ_2 então os respectivos custos preservam a mesma relação de ordem.

No GRASP original escolhe-se apenas a permutação de menor custo para que seja efetuada a busca local. A alteração proposta é escolher também a permutação que possui o menor número de inversões e gerar permutações da vizinhança, aplicando-se a estrutura 2-trocas em ambas as permutações. Nesta nova proposta, a busca local termina quando não houver mais chances de melhorar nem o custo da permutação e nem o número de inversões.

No nível 1 da árvore de busca tem-se $C_{n,2}$ permutações. A partir do nível 2, este número pode dobrar pois são escolhidas duas permutações diferentes para atender ao duplo critério. Nada impede porém que sejam iguais, diminuindo o custo computacional nestes casos. A figura 5.2 mostra a modificação proposta.

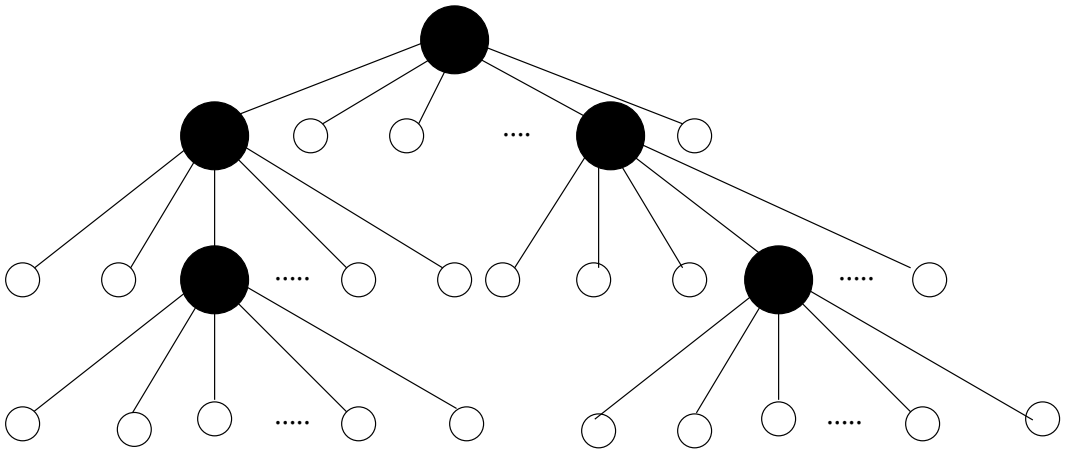


Figura 5.2: Busca local ampliada

É claro que o novo processo de busca é mais custoso, entretanto, em nossos testes, foi possível obter o ótimo de várias instâncias da $QAPLIB$ em um número menor de iterações do que o GRASP original em Li et al. [LPR94], como vemos a seguir.

5.3.1 Análise dos Resultados Computacionais

Este algoritmo foi implementado em linguagem C e executado em estações de trabalho Digital DEC-3000, OSF1, 125 Mhz e 32 MRAM. Para comparação dos resultados, foi implementado o GRASP de Li et al. [LPR94] e o Grasp Ampliado, utilizando-se a mesma linguagem, estrutura de dados, dados de entrada e levando-se em consideração, os tipos diferentes de busca de cada algoritmo. Ambos os algoritmos terminam quando a solução ótima é encontrada.

A tabela 5.1 mostra os resultados na seguinte ordem: solução ótima encontrada na $QAPLIB$, melhor solução alcançada pelo Grasp Ampliado, número de iterações necessárias, tempo real (em segundos). Na sequência, a ordem se repete para o GRASP.

É claro que a busca local proposta é mais custosa pois a partir das duas permutações escolhidas (a de menor custo e a de menor número de inversões), a cardinalidade da vizinhança é dobrada. Entretanto, foi possível obter o ótimo das instâncias

da $QAPLIB$ em média um número 2.14 vezes menor de iterações que o do GRASP original, como registra a tabela 5.1. Com respeito ao tempo computacional, em média o GRASP é 2.29 vezes mais rápido. O ganho no número de iterações parece não ser suficiente para compensar o tempo gasto na duplicação de permutações que devem ser pesquisadas na vizinhança, principalmente porque o cálculo do número de inversões é dispendioso.

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

Tabela 5.1: Grasp Ampliado(GAmpl) x GRASP original (G)

Para a instância Nug30 o número máximo de iteração que utilizamos foi 1000. O Grasp Ampliado obteve a solução ótima, cujo valor é de 6124, em 364 iterações com tempo computacional de 102505 segundos e o GRASP atingiu o valor de 6136 na iteração 955, em 8030 segundos. Pode-se notar a redução no número de iterações e, além disso, alcançamos soluções melhores com o Grasp Ampliado, mas concluímos que não é recomendado para instâncias com dimensões superiores a 20, pois o tempo computacional é excessivamente alto.

5.4 GRASP Restrito

Nesta seção é apresentada uma proposta para descartar soluções iniciais supostamente ruins, com base na normalização de custos calculados num intervalo entre limites de solução. No 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.

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

Os limites inferior e superior para o Problema Quadrático de Alocação, aqui utilizados, são respectivamente os limite inferior universal e superior universal de todas as instâncias pertencentes à $RelClass(Q^*)$, determinados no capítulo 2.

Define-se o custo normalizado de uma solução $\varphi \in \Pi_n$ do PQA por:

$$|Custo(\varphi)| = \frac{Custo(\varphi) - LimInf}{LimSup - LimInf}. \tag{5.5}$$

Para a instância GP66 tem-se $LimInf = 389$, a soma da diagonal principal da matriz Q^*, e $LimSup = 587$, a soma da diagonal secundária da mesma matriz. Consideremos a solução ótima $\varphi^* = (2\ 4\ 3\ 1)$ e seu custo ótimo $Custo(\varphi^*) = 403$. Assim, pela equação (5.5) temos $|Custo(\varphi^*)| = 0.0707$. Na figura 5.3, podemos verificar que $|Custo(\varphi^*)|$ está bem próximo do $LimInf$. Para comparação, seja uma solução viável, para a mesma instância, φ de $Custo(\varphi) = 479$, cujo custo normalizado é $|Custo(\varphi^*)| = 0.4545$. Neste caso, parece que φ está "relativamente" longe do $LimInf$.

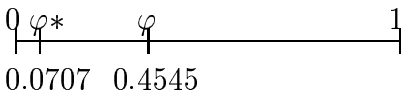


Figura 5.3: Distâncias normalizadas

Os limites usados poderiam ser substituídos por outros mais precisos que podem ser encontrados em Li et al. [LPRR94], Carraresi e Malucelli [CM92], Gilmore [Gi62], Lawler [La63] e Karisch et al. [KCCE98]. O custo da determinação de melhores limites, todos de ordem igual ou superior a $O(n^3)$, poderia reduzir a vantagem obtida

pelo algoritmo proposto, dada a baixa complexidade da determinação do limite aqui utilizado.

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 5.2.1 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 α e β $0 < \alpha, \beta < 1$ são importantes neste processo pois "filtram" as melhores opções de escolha.

O ajuste destes parâmetros não é uma tarefa fácil. A 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 parâmetros, aumenta-se o espaço de busca, mas deteriora-se a média das soluções iniciais. Na tabela 5.2 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 de acordo com a equação (5.5) e os cálculos feitos para $\alpha = \beta = 0.1, 0.25, 0.5, 0.75, 1.0$.

instâncias	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
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
Els19	0.57	0.49	0.45	0.46	0.52

Tabela 5.2: 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, com $\alpha = \beta = 0.75$, resultando média normalizada de 0.46. O custo normalizado da solução ótima é igual a 0.048. Propomos aceitar as soluções geradas com custo abaixo de um determinado limite, por exemplo Lim = 0.45. A figura 5.4 ilustra as soluções aceitas, cujo custo normalizado é inferior ou igual a $Lim = 0.45$.

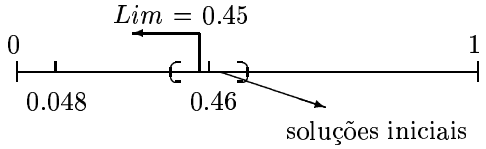


Figura 5.4: Limite de aceitação para soluções iniciais

Com esta poda no espaço das soluções iniciais tem-se um menor tempo de execução do algoritmo pois várias tentativas de busca, que não seriam promissoras, são abortadas.

5.4.1 Análise dos resultados computacionais

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, Pardalos e Resende [LPR94] e o GRASP Restrito, ambos utilizando a mesma linguagem, os mesmos dados de entrada e a mesma estrutura de dados. A restrição aqui proposta foi testada para algumas instâncias da $QAPLIB$, com dimensões $n = 12$, 15, 18 e 19, com valores de limites dados por $Lim = 0.3$, 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. O comportamento do algoritmo proposto foi observado a partir dos testes realizados com um conjunto de 18 instâncias, enumeradas na tabela 5.2, com os parâmetros $\alpha = \beta = 0.1$, 0.25, 0.5, 0.75 e 1.0 e $Lim = 0.3$, 0.45 e 0.5, totalizando 270 execuções. O GRASP Restrito apresentou melhor desempemho com os valores de α e β iguais a 0.5 e 0.75 e os valores de $Lim = 0.45$ e 0.5. Quando $Lim = 0.3$ o algoritmo descartou muitas soluções, o que prejudicou seu desempenho.

A tabela 5.3 mostra os resultados para instâncias de dimensões maiores, apresentando a média normalizada das soluções iniciais geradas pelos parâmetros $\alpha = \beta =$

0.5, o número de soluções descartadas pelos limites de aceitação com valores $Lim = 0.45$ e 0.5 e a economia no tempo computacional, quando comparado ao GRASP da seção 5.1.

instâncias	Média normalizada ($\alpha = \beta = 0.5$)	Num. de sol. descartadas ($Lim = 0.45$)	Num. de sol descartadas ($Lim = 0.5$)	Econ.tmp computacional. com $Lim = 0.45$
Nug30	0.45	229	21	38.21%
Kra30a	0.45	276	31	44.91%
Kra30b	0.45	253	27	42.05%
Ste36a	0.37	45	7	7.62%
Ste36b	0.25	1	0	0%
Sko42	0.42	307	2	52.29%
Sko49	0.46	338	3	57.77%
Sko64	0.46	408	0	69.7%

Tabela 5.3: Comparação GRASP e GRASP Restrito

Com relação ao tempo computacional, o algoritmo proposto neste trabalho apresentou um desempenho melhor. As instâncias Ste36a e Ste36b, cujas médias normalizadas das soluções iniciais foram baixas não apresentaram bons resultados, como visto na tabela 5.3.

A qualidade da solução encontrada pelo GRASP Restrito é tão boa quanto a do GRASP estando, em média, a 98% de proximidade das melhores soluções conhecidas da $QAPLIB$.

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 na tabela 5.3 que o número de soluções descartadas é pequeno quando considerado $Lim = 0.5$.

Para melhor escolher o valor do limite, deve-se levar em conta a média normalizada das soluções iniciais geradas na fase de construção. No caso das instâncias Ste36a e Ste36b, um bom limite seria $Lim = 0.37$ para a primeira, e $Lim = 0.25$ para a segunda.

Utilizando o novo algoritmo em instâncias de dimensões $n = 100$, os resultados obtidos foram bastante satisfatórios. As instâncias escolhidas para os testes desta ordem são as de Skorin-Kapov, sko100a-f, disponíveis na $QAPLIB$. Para cada uma, foram executadas 100 iterações. As médias das soluções iniciais estiveram um pouco

acima de 0.46, quando $\alpha = \beta = 0.5$ e o limite escolhido foi $Lim = 0.47$. Isto permitiu economizar o tempo computacional gasto sem qualquer prejuízo à qualidade das soluções alcançadas pelo GRASP Restrito. A melhor solução viável alcançada em cada instância foi a mesma atingida pelo GRASP original. A proximidade com a melhor solução viável conhecida ($QAPLIB$) foi em média 99%. Os resultados encontram-se na tabela 5.4.

	Média Normalizada $\alpha = \beta = 0.5$	Num. Soluções descartadas ($Lim = 0.47$)	Proximidade melhor solução viável	Economia 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 5.4: Resultados das instâncias Skorin-Kapov

Nota-se que a escolha do limite, considerando a média normalizada das soluções iniciais, é um ponto importante para a eficiência do algoritmo proposto. Em escolhendo o valor Lim = 0.5, nessas instâncias Skorin-Kapov, não haveria economia alguma, pois muitas soluções seriam aceitas, e no caso de Lim = 0.45, poucas soluções seriam submetidas à busca local, prejudicando a qualidade das soluções.

5.5 Algoritmos Genéticos

Desde os anos 60 e 70, os Algoritmos Genéticos (AG's) vêm tomando um grande impulso com os estudos de John Holand e sua equipe da Universidade de Michigan. Seus trabalhos resultaram na publicação do livro entitulado *Adaptation in Natural and Artificial Systems* [Hol75], citação obrigatória nos artigos sobre esse assunto.

A evolução das espécies, segundo Darwin [Dar1859], é uma consequência da seleção dos indivíduos mais aptos ao ambiente em que vivem. Estes têm mais chance de sobreviver e de gerar filhos que herdam as características dos pais. AG's procuram imitar esta forma de "otimização" das espécies. O ambiente corresponde ao problema a ser otimizado e os indivíduos, às soluções deste problema.

Todo indivíduo possui um genótipo que é a representação de uma solução do

problema. A expressão deste genótipo constitui o fenótipo do indivíduo, que será avaliado, produzindo um dado numérico que corresponde sua adequação ao ambiente (ou problema). Com os indivíduos da população avaliados pode-se determinar quais terão chance de sobreviver e transmitir suas características para a próxima geração.

Os AG's têm sido usados para resolver problemas em várias áreas científicas, principalmente na área de Otimização Combinatória. Vamos destacar os trabalhos de Tate e Smith [TS94] e Fleurent e Ferland [FF94] que aplicaram os AG's ao problema que estamos estudando, ou seja, Problema Quadrático de Alocação.

5.5.1 Noções básicas do funcionamento dos AG's

Os AG's começam sua evolução a partir da geração aleatória de uma população de m indivíduos, onde cada um deles representa uma solução do problema proposto. Tal população é submetida a um processo iterativo composto de três passos principais: (1) avaliação, (2) seleção e (3) aplicação dos operadores genéticos, *crossover* - recombinação dos genes e mutação - alteração aleatória de genes. Este ciclo é repetido até que um critério de parada estipulado seja alcançado, por exemplo, um número pré-estabelecido de gerações (iterações) que devem ser processadas.

A figura (5.5) ilustra uma iteração dos AG's.

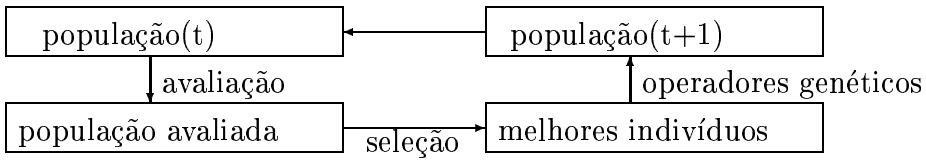


Figura 5.5: Uma iteração dos Algoritmos Genéticos

A abordagem básica dos Algoritmos Genéticos é apresentada a seguir.

AG's;

```
1   gerar população inicial;
2   avaliar população;
3   Enquanto o critério de parada não for satisfeito
4      selecionar indivíduos para a próxima população;
5      aplicar crossover e mutação;
6      avaliar população;
7   FimEnquanto;
```

FimAG's;

É interessante ressaltar a relação dos termos empregados nos Algoritmos Genéticos com os termos familiares da matemática. A tabela (5.5) mostra essa relação.

Algoritmos Genéticos	**Matemática**
população de cromossomos	conjunto de soluções
cromossomo	solução
gene	componente da solução

Tabela 5.5: Relação dos termos

5.5.2 Inicialização e representação dos indivíduos

Para a geração de uma população inicial de m indivíduos, normalmente utilizam-se procedimentos aleatórios ou heurísticas elaboradas para este fim.

Na **representação binária**, os indivíduos são vetores binários cuja construção depende do problema que está sendo tratado. Por exemplo, no caso de encontrar o valor que maximiza a função $f(x) = x^2$, no intervalo [10,140], a dimensão do vetor binário será determinada de acordo com a precisão exigida (por exemplo, $prec = 10^{-3}$). Uma maneira fácil de descobrir o número de componentes do vetor binário é através da expressão (5.6), considerando um intervalo [a,b]:

$$2^n = \ prec \ \times (b - a), \tag{5.6}$$

onde n é o número de componentes do vetor.

Para transformar o vetor binário, $(b_1 \dots b_n)$, em número real, temos que levar em consideração a precisão exigida e o intervalo [a,b] que está sendo considerado, através das seguintes expressões:

$$x' = \sum_{i=0}^{n-1} b_i 2^i \quad \text{e} \quad x = a + \tfrac{x'(b-a)}{2^n - 1}. \tag{5.7}$$

A representação binária para as permutações viáveis do Problema Quadrático de Alocação não tem relação com o intervalo [a,b] ou a precisão. Neste caso, o número de componentes está relacionado com a dimensão da instância do PQA e será $n \times n$. Se o bit j do segmento i for igual a 1, tem-se $\varphi(i) = j$, como mostra a figura (5.6).

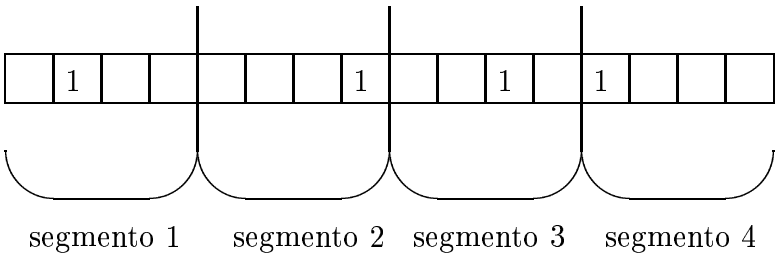


Figura 5.6: Representação binária da permutação $\varphi = (2\ \ 4\ \ 3\ \ 1)$

5.5.3 Avaliação da população e seleção dos indivíduos

Citando novamente a evolução das espécies segundo Darwin, os indivíduos mais aptos ao meio ambiente em que vivem possuem maiores chances de sobrevivência e assim, de gerar filhos que herdam suas características. Os Algoritmos Genéticos tentam simular esse desafio da natureza avaliando os indivíduos de uma população "medindo" tal aptidão. A definição da função *aval* de avaliação está intimamente associada às características do problema.

Considerando a população de m indivíduos como sendo $(v_1, \ldots, v_m)$, deve-se calcular a probabilidade de cada indivíduo dada por

$$p_i = \frac{aval(v_i)}{\sum_{j=1}^{m} aval(v_j)} \tag{5.8}$$

e a probabilidade acumulada é então

$$q_i = \sum_{j=1}^{i} p_j, \quad i = 1, 2, ..., m. \tag{5.9}$$

Como uma roleta, gera-se aleatoriamente um número $r \in (0, 1]$. Se $r \leq q_1$ então v_1 é selecionado senão, o indivíduo v_i $(2 \leq i \leq m)$ onde $q_{i-1} < r \leq q_i$ é escolhido. Este sorteio é repetido m vezes (tamanho da população). Desta forma, tem-se a população selecionada com mesmo número de indivíduos, que supostamente possui os indivíduos com as maiores probabilidades. A tabela 5.6 e a figura 5.7 mostram um exemplo fictício de uma população (v_1, v_2, v_3, v_4) cujas probabiliades p_i e probabilidades acumuladas q_i foram calculadas a partir das equações (5.8) e (5.9). Os valores para a função de avaliação $aval(v_i)$ dos indivíduos foram escolhidos sem a preocupação de uma definição formal, pois o objetivo do exemplo é apenas o de ilustrar a construção da roleta para o sorteio dos exemplares. Os indivíduos v_2 e v_4 estão nas faixas mais largas, onde o número $r \in (0, 1]$ pode cair com mais facilidade,

nada impedindo que caia nas faixas 1 e 3, selecionando desta forma, exemplares menos aptos ao ambiente. Sobre essa população selecionada serão aplicados os operadores *crossover* e mutação.

v_i	$aval(v_i)$	p_i	q_i	faixas
v_1	867.30	0.0198	0.0198	(0.0000; 0.0198]
v_2	17918.50	0.4097	0.4295	(0.0198; 0.4295]
v_3	8632.27	0.1979	0.6274	(0.4295; 0.6274]
v_4	16312.40	0.3730	1.0004	(0.6274; 1.0004]

Tabela 5.6: Exemplo fictício para ilustrar o sorteio de indivíduos

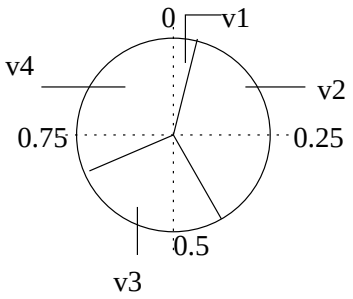


Figura 5.7: Roleta com as faixas definidas pela tabela 5.6.

5.5.4 Operadores genéticos: Crossover e Mutação

O operador genético *crossover* simula a recombinação de genes entre dois indivíduos e a mutação é uma alteração em um determinado gene. A figura (5.8) ilustra o comportamento desses operadores sobre os indivíduos. O ponto de crossover e os genes que participarão da mutação são escolhidos aleatoriamente.

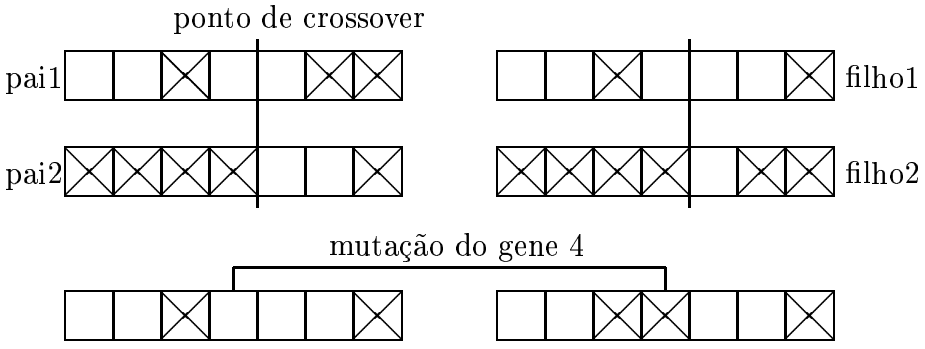


Figura 5.8: *crossover* e mutação

Os AG's trabalham com os seguintes parâmetros:

- número de iterações;
- tamanho da população m;
- probabilidade de *crossover* p_c;
- probabilidade de mutação p_m.

A escolha dos indivíduos que participarão do *crossover* depende de uma probabilidade de *crossover*, p_c, que é um dos parâmetros dos AG's. A cada iteração, para todos os indivíduos da população, deve-se gerar aleatoriamente um número $r_c \in (0, 1]$. Se $r_c \leq p_c$ então o indivíduo é escolhido. O procedimento para selecionar dois indivíduos que farão a mistura dos genes fica a cargo do pesquisador. Para o operador mutação, o sorteio de um número aleatório $r_m \in (0, 1]$ é feito para cada gene dos indivíduos. Se $r_m \leq p_m$ então o gene sofrerá mutação, onde p_m é a probabilidade de mutação. O mecanismo dos operadores são diferentes para cada aplicação dos AG's. Beasley et al. [BBM93b] e Michalewicz [Mic92] discutem vários tipos de operadores, além de representações mais apropriadas dos indivíduos explorando as características de cada tipo de problema. Tate e Smith [TS94] e Fleurent e Ferland [FF94] fazem uso de operadores mais adequados ao PQA.

A seguir, é apresentado uma proposta de um Algoritmo Genético Híbrido, baseando-se no trabalho desenvolvido por Garcia et al. [GMBR95] e por Li et al. [LPR94].

5.6 Algoritmo Genético Híbrido Aplicado ao PQA

No desejo de se obter melhores resultados para os problemas de otimização *NP-Hard*, os pesquisadores misturam diversas heurísticas dando origem aos chamados Algoritmos Genéticos Híbridos. Fleurent e Ferland [FF94] utilizam a busca tabu ou um procedimento de busca local, como operadores de mutação, na aplicação dos AG's no Problema Quadrático de Alocação. Nesse caso, a idéia original da mutação foi alterada pois tomam-se os indivíduos para aplicar a mutação, e não apenas os genes que os compõem.

Nesta seção são apresentadas duas versões para o Algoritmo Genético Híbrido, desenvolvidas neste trabalho. A primeira trata de algoritmo genético com população

inicial aleatória e a segunda, algoritmo genético com população inicial GRASP. Em pseudo-linguagem, apresentada a seguir, pode-se observar as diferenças entre as duas versões híbridas, nos passos de geração da população inicial e processo de seleção. A heurística utilizada para fazer parte dessas versões dos AG's Híbridos foi um procedimento simples de busca local, com vizinhança 2-trocas, desenvolvido na seção 5.2.2, caracterizando a mutação.

AG-Aleatório;

```
1   gerar população inicial com processo aleatório;
2   Enquanto o ''critério de parada não for satisfeito''
3      selecionar indivíduos para a próxima população;
4      aplicar crossover;
5      aplicar mutação = busca local;
6   FimEnquanto;
```

FimAG's-Aleatório;

AG-Grasp;

```
1   gerar população inicial com fase de construção do GRASP;
2   Enquanto o ''critério de parada não for satisfeito''
3      aplicar crossover;
4      aplicar mutação = busca local;
5   FimEnquanto;
```

FimAG's-Grasp;

O processo de seleção na versão *AG-Grasp* foi eliminado pois as probabilidades p_i dos indivíduos, $i = 1, \ldots, m$, calculadas pela expressão (5.8) variam pouco, acarretando uma população selecionada semelhante a que lhe deu origem. Isto acontece pois uma população inicial, onde cada elemento é gerado pela fase de construção do GRASP (seção 5.2.1), de uma certa forma, já é composta de indivíduos de boa qualidade. Os operadores genéticos e a forma como são escolhidos os participantes para o *crossover* e mutação não se alteram. Esses passos são apresentados, em detalhes, no decorrer desta seção.

5.6.1 População Inicial e Representação dos indivíduos

Na primeira versão do algoritmo genético híbrido (*AG-Aleatório*), a geração de cada um dos m indivíduos da população é feita de forma aleatória, sem a preocupação com a qualidade dos exemplares.

Na tentativa de diminuir o número de iterações (gerações) necessárias para alcançar uma solução ótima ou viável aceitável, utilizou-se a fase de construção do GRASP (seção 5.2.1) para gerar indivíduos de boa qualidade para a população inicial, dando origem a versão *AG-Grasp*.

A representação binária da seção 5.5.2 não é aconselhável pois exige uma memória excessiva. Para contornar esse problema, uma representação com números inteiros, denominada de representação inteira, de fácil compreensão e tratamento, foi escolhida para ser utilizada nas duas versões dos algoritmos genéticos híbridos propostas neste trabalho. O indivíduo é exatamente a imagem da permutação, consequentemente, a economia de memória é excelente. A figura (5.9) ilustra a permutação de dimensão 4, $\varphi = (2\ \ 4\ \ 3\ \ 1)$.

2	4	3	1

Figura 5.9: Representação inteira da permutação $\varphi = (2\ \ 4\ \ 3\ \ 1)$

5.6.2 *Crossover* e Mutação

O operador *crossover* escolhido para estas versões dos AGH's preserva a viabilidade das soluções. A definição deste operador considera a quebra do par de indivíduos, v_1 e v_2, em dois pontos, p_1 e p_2, como discutido em [BBM93b]. O primeiro ponto p_1 é escolhido aleatoriamente e, além disso, é fornecido o parâmetro *alcance máximo*, que determina o segundo ponto de quebra $p_2 = p_1$+*alcance máximo*. Um operador semelhante foi utilizado por Garcia et al. em [GMBR95] para o problema de alocação de disciplinas a professores de um departamento.

Para a aplicação do *crossover* foi desenvolvido o seguinte algoritmo.

Crossover(v_1,v_2,p_1,***alcance máximo***);

1 `Para` $k = p_1, \ldots, (p_1$+`alcance máximo`$)$ `faça`

2 $\quad valor1 = v_1[k]$;

3 $valor2 = v_2[k]$;

4 buscar em v_1 o $valor2$ e fazer a troca;

5 buscar em v_2 o $valor1$ e fazer a troca;

6 FimPara;

FimCrossover;

A escolha dos indivíduos para o *crossover* foi implementada seguindo o algoritmo **EscolhaCros** que possui, como parâmetros de entrada, o tamanho da população m e a probabilidade de *crossover* p_c.

EscolhaCros(m, p_c);

1 Para $ind1 = 1, \ldots, (m/2)$ faça

2 gerar $r_c \in (0, 1]$;

3 Se $r_c \leq p_c$ então

4 gerar $(m/2) + 1 \leq ind2 \leq m$;

5 o par ind1 e ind2 participarão do *crossover*;

6 FimSe;

7 FimPara;

FimEscolhaCros;

A sequência dos passos da aplicação do operador *crossover* sobre os indivíduos escolhidos pelo procedimento **Crossover**, podem ser acompanhadas na figura (5.10).

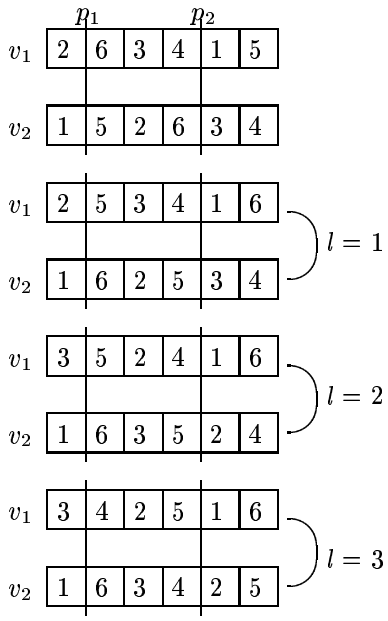


Figura 5.10: Sequência dos passos para *crossover*

A mutação é caracterizada por uma busca local de vizinhança 2-trocas, como apresentada na seção 5.2.2. Para todos os indivíduos da população, o algoritmo gera aleatoriamente um número $r_m \in (0, 1]$. Se $r_m \leq p_m$ então este sofrerá mutação, sendo submetido ao procedimento de busca local. Naturalmente, o indivíduo que retorna de um procedimento de busca tem custo menor ou igual aquele tomado como solução inicial.

5.6.3 Análise dos Resultados Computacionais

As duas versões, AG-Aleatório e AG-GRASP, foram implementadas em liguagem C, em estações de trabalho Digital DEC-3000 com OSF1, 125 Mhz e 32 MRAM.

Foram utilizados os seguintes parâmetros:

- número máximo de gerações $numger = 1000$;
- número de indivíduos da população:
  - $n = 12 \rightarrow numind = 100$;
  - $n = 15 \rightarrow numind = 150$;
  - $n = 18, 19, 20 \rightarrow numind = 200$;
- probabilidade de *crossover* $p_c = 0.3$;
- probabilidade de mutação $p_m = 0.03$;
- critério de parada:
  - atingir $numger$;
  - ou não atualizou a melhor solução por $(numger/3)$ gerações;

As instâncias submetidas aos dois algoritmos foram retiradas da $QAPLIB$. A tabela 5.7 faz a comparação entre as duas versões, enfocando a solução alcançada.

Uma análise quanto à qualidade da solução alcançada pelas duas versões, mostra que o AG-Aleatório obtive um melhor desempenho comparado com a versão AG-GRASP. O fato da geração da população inicial na segunda versão levar vantagem na qualidade inicial dos indivíduos, perde no tempo de sua construção, e além disso, gera uma população homogêna. A consequência desta homogeneidade pode ser

observada na tabela 5.7, que mostra as instâncias que não alcançaram o ótimo pois caíram em ótimos locais, destaque na tabela com (**). As instâncias que estão destacadas por (*), falharam na busca do ótimo, pois não houve melhora na solução por ($numger/3$) gerações, um dos critérios de parada estabelecidos.

instância	$QAPLIB$	AG-Aleatório		AG-GRASP	
		solução alcançada	GAP	solução alcançada	GAP
chr12a	9552	9552	-	10214(**)	6.90%
chr12b	9742	9742	-	9742	-
chr12c	11156	11156	-	12528(**)	12.30%
chr15a	9896	9896	-	11938(**)	20.63%
chr15b	7990	7990	-	9854(**)	23.33%
chr15c	9504	9504	-	11900(**)	25.21%
chr18a	11098	11098	-	15134(**)	36.37%
chr18b	1534	1534	-	1556(**)	1.43%
nug12	578	578	-	578	-
nug15	1150	1150	-	1150	-
nug20	2570	2570	-	2570	-
rou12	235528	235852(*)	0.13%	235528	-
rou15	354210	354210	-	354210	-
rou20	725522	726100(*)	0.07%	729458(*)	0.54%
scr12	31410	31410	-	31410	-
scr15	51140	51140	-	51140	-
scr20	110030	110030	-	110352(**)	0.3%
els19	17212548	17212548	-	17212548	-

Tabela 5.7: AG-Aleatório e AG-GRASP: comparação na qualidade da solução

A tabela 5.8 fornece o número de gerações, o número de buscas locais (mutação), tempo para a construção da população inicial e tempo total para encontrar as soluções da tabela 5.7.

Quanto ao tempo computacional, o AG-Aleatório é mais rápido, mesmo que a instância necessite de um número maior de gerações para atingir a solução $QAPLIB$ e, consequentemente, um número maior de buscas. O AG-GRASP gasta um tempo computacional relativamente grande na construção da população inicial (exemplos, a instância nug12, rou15 e scr15).

Observamos que algumas modificações da idéia original dos AG's parecem eficientes, mas quando implementadas, os resultados obtidos são de baixa qualidade. Para os AG's, a variedade dos exemplares é um fator muito importante para que as combinações de genes forneçam indivíduos mais aptos ao ambiente.

instância	AG-Aleatório				AG-GRASP			
	número gerações	número buscas	tmp pop. inicial(s)	tempo total(s)	número gerações	número buscas	tmp pop. inicial(s)	tempo total(s)
chr12a	52	158	0.02	4.95	337	944	1.37	37.58
chr12b	36	95	0.02	3.88	1	4	1.30	1.57
chr12c	2	6	0.02	0.27	338	968	1.35	38.37
chr15a	230	1091	0.05	65.48	337	1503	5.50	108.75
chr15b	494	2355	0.05	133.45	336	1528	5.42	109.12
chr15c	143	682	0.03	43.58	368	1596	5.47	119.15
chr18a	254	1470	0.08	213.17	336	2052	16.28	262.62
chr18b	87	512	0.08	101.28	346	2087	16.40	276.07
nug12	7	20	0.02	0.73	2	6	1.42	1.77
nug15	55	260	0.03	26.27	2	16	5.98	8.07
nug20	67	402	0.08	182.42	120	720	32.00	380.32
rou12	337	1014	0.02	36.42	24	71	1.47	5.60
rou15	7	37	0.03	3.65	3	18	6.20	8.92
rou20	362	2199	0.10	871.23	348	2113	31.67	1041.82
scr12	5	21	0.03	0.98	1	6	1.45	1.75
scr15	5	28	0.05	3.33	2	9	6.05	7.70
scr20	464	2783	0.08	1338.37	367	2170	30.81	540.95
els19	4	28	0.08	13.53	9	57	25.03	46.77

Tabela 5.8: AG-Aleatório e AG-GRASP: comparação no esforço computacional

5.7 Simulated Annealing

A técnica Simulated Annealing (SA), como método para resolução de Problemas de Otimização Combinatória, foi proposta por Kirkpatrick et al. [KGV83]. O nome deste algoritmo deriva da simulação do resfriamento de sólidos, devido a Metropolis et al. [MRRTT53]. O termo Annealing refere-se ao processo de cozimento de material, lentamente, até que se atinja um estado de equilíbrio. Define-se uma função objetivo z que avalia a energia do sistema em cada estado do processo. Um estado inicial é estabelecido com energia z_0, e em seguida, provoca-se uma perturbação no sistema. Existe uma série de estados que o processo pode atingir, após esta perturbação, que constitue a vizinhança do estado anterior, cada um com sua respectiva energia z. Dentre esses candidatos, escolhe-se aleatoriamente um, e calcula-se a troca de energia $\Delta E = z - z_0$. Se $\Delta E < 0$ então o novo estado é aceito, caso contrário, aceita-se a transformação com uma certa probabilidade $p(\Delta E) = exp^{\frac{-\Delta E}{k_b T}}$, onde T é um parâmetro de controle correspondente a temperatura e k_b é a constante de Boltzmann. A troca ΔE na função objetivo corresponde a troca no nível de energia que ocorre quando a temperatura T decresce, de acordo com uma escala,

denominada *annealing*. O método fornece um mecanismo para a aceitação de pequenos acréscimos na função objetivo, controlado pela probabilidade $p(\Delta E)$ através da temperatura. O procedimento **SA**, mostra o funcionamento do método.

SA(instância,k_b);

```
1   gerar aleatoriamente um estado inicial;
2   inicializar a temperatura T;
3   Enquanto T > 0 faça
4      Enquanto ''estado de equilíbrio não é alcançado'' faça
5         gerar aleatoriamente um estado dentro da vizinhança;
6         avaliar a troca de energia ΔE;
7         Se ΔE < 0 então
8            atualiza-se o estado corrente;
9            Senão
10               atualiza-se o estado corrente com p(ΔE);
11            FimSe;
12         FimSe;
13      FimEnquanto;
14      decrescer a temperatura T de acordo com a escala annealing;
15   FimEnquanto;
16   Retornar o estado de menor energia;
```

FimSA;

Um movimento de busca com *estratégia descendente* pode levar a ótimos locais e, raramente, a uma solução de boa qualidade. A principal vantagem do método SA é a habilidade de escapar de ótimos locais. A figura 5.11 ilustra a armadilha de um ótimo local. As vizinhanças dos pontos P e R têm em comum o ponto Q. Portanto, uma estratégia de busca descendente, iniciada em P ou R, encontraria o mesmo ótimo local Q, sem chances de alcançar um outro mínimo local, ponto S, cujo valor para a função seja menor. Tão pouco teria chances de encontrar o mínimo global M.

Os principais componentes do método de Simulated Annealing são:

- a temperatura T, que é o parâmetro que controla a probabilidade $p(\Delta E)$ de aceitação de um acréscimo na função objetivo.
- a escala *annealing*, função decrescente pré-estabelecida, que determina quando

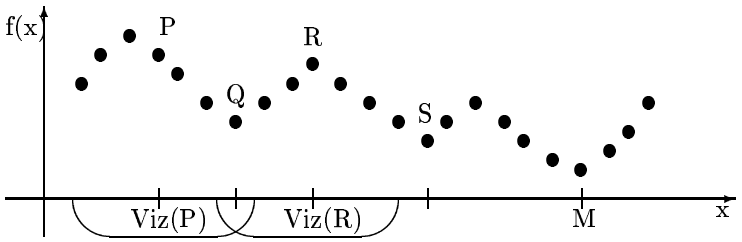


Figura 5.11: Ótimos locais: Q e S; Ótimo global: M

e como a temperatura é reduzida. Durante o processo a temperatura T diminui, através da escala *annealing*, para reduzir $p(\Delta E)$.

- o estado de equilíbrio, isto é, a condição em que uma aceitação de acréscimo na função objetivo dificilmente ocorre.

A analogia deste processo termodinâmico com a Otimização Combinatória é feita atribuindo-se um significado específico para os termos da Termodinâmica. A tabela 5.9 mostra esta correspondência.

Termodinâmica	**Otimização Combinatória**
estado do sistema	solução viável
troca de estado	solução vizinha
energia	custo
ΔE	variação no valor dos custos
temperatura	parâmetro de controle
estado de equilíbrio	melhor solução heurística

Tabela 5.9: Analogia

Existe uma flexibilidade para a utilização dos parâmetros, estruturas de vizinhanças e estratégias para o procedimento de busca. Mavridou e Pardalos [MP97], discutiram o Simulated Annealing quando aplicados a Problemas de Layout de Facilidades.

5.7.1 Simulated Annealing e o PQA

Alguns pesquisadores se dedicam a aplicar *Simulated Annealing* ao Problema Quadrático de Alocação, definindo seus parâmetros e funções com base em diferentes aspectos.

Na busca de uma solução em uma vizinhança, a estrutura adotada pela maioria dos pesquisadores baseia-se numa busca aleatória de vizinhos potenciais para efetuar a troca. Connolly, [Co90], utilizou a busca sequencial ou busca ordenada, sem reposição de elementos, fornecendo melhores resultados que a busca aleatória.

A temperatura e a escala *annealing* podem variar sem perder sua eficiência. Lundy e Mees [LM86] criaram uma escala *annealing* para redução de temperatura usando a função

$$T(t+1) = \frac{T(t)}{1+\beta T(t)}, \tag{5.10}$$

onde $\beta < T_0$ e $t \geq 0$, para ser aplicada após cada troca de vizinhança. Para isso, é necessário que a temperatura inicial T_0 seja suficientemente grande, de modo que quase todas as tentativas de trocas de solução sejam aceitas. Kirkpatrick et al. [KGV83] adotaram um esquema de redução de temperatura que utiliza um parâmetro α, $0 < \alpha < 1$. A escala *annealing* por eles usada é muito simples:

$$T(t+1) = \alpha T(t), \tag{5.11}$$

onde $t \geq 0$.

A cada temperatura, M iterações (tentativas de aceitar mudança de estado) são feitas, até que se atinja o "estado de equilíbrio". A dimensão de M também é um fator importante na convergência do algoritmo. Burkard e Rendl [BR83] fixaram $M = 2n$, onde n é a ordem do problema.

Como toda meta-heurística, os autores têm ampla liberdade de estabelecer o critério de parada que mais lhes parecer interessante. Os mais utilizados são: o baixo número de aceitação de soluções; o equilíbrio atingido pela função objetivo; ou quando a temperatura $T(t)$ alcança um valor pré-definido, suficientemente pequeno.

5.7.2 Algoritmo REDINV-SA

Nesta seção é apresentado o algoritmo REDINV-SA, aplicado ao PQA, desenvolvido por Querido, [Que94], em sua tese de doutorado, com base na linha de pesquisa que estamos trabalhando.

A função objetivo avaliada pelo algoritmo é dada por

$$z = \sum_{i,j=1}^{n} f_{ij} d_{\varphi(i)\varphi(j)}, \tag{5.12}$$

onde $F = [f_{ij}]$ e $D = [d_{kl}]$ são as matrizes simétricas, com elementos inteiros, que definem a instância $PQA(F, D)$ do Problema Quadrático de Alocação. A vizinhança $\varphi \in \Pi_n$, denotada por $Viz(\varphi)$, é constituída por permutações $\varphi' \in \Pi_n$ resultantes da troca de dois elementos em $\rho \in \Pi_N$, associada a φ no grafo G_{Liv}. Esta troca deve reduzir o número de inversões de $\rho \in \Pi_N$, razão do nome REDINV.

O procedimento **ConstViz**, em pseudo-código, mostra os passos necessários para a construção da $Viz(\varphi)$. Considere para tal a bijeção ψ_{ij} definida na seção 2.1 e as permutações ϕ_F e ϕ_D que armazenam as trocas feitas nas posições das coordenadas dos vetores **F** e **D**, que definem o $PAL(F, D)$, levando-os aos vetores $\mathbf{F}^-$ e $\mathbf{D}^+$, de coordenadas dispostas em ordem não-crescente e não-decrescente, respectivamente.

ConstViz(φ);
1 determinar ξ_φ através da expressão (2.4);
2 determinar $\rho_\xi = \phi_D \circ \xi_\varphi \circ \phi_F^{-1}$;
3 Para $i = 1, \ldots, N$ faça
4 determinar a posição s tal que $max|i - \rho_\xi(i)|$;
5 FimPara;
6 Para $i = 1, \ldots, N$ faça
7 determinar a posição r tal que $min\{|s - \rho_\xi(i)| + |i - \rho_\xi(s)|\}$;
8 FimPara;
9 retornar à φ e trocar os pares de vértices $\psi^{-1}(s)$ e $\psi^{-1}(r)$;
FimConstViz;

A seguir, será exibido um exemplo para acompanhar, em detalhes, o procedimento **ConstViz**.

Exemplo 5.1 *Considere a instância de Gavett e Plyter (GP66) já apresentada na seção 2.1.*

Os vetores são **F** = (28 25 13 15 4 23) e **D** = (6 7 2 6 5 1) e as permutações que os ordenam são ϕ_F = (1 2 5 4 6 3) e ϕ_D = (4 6 2 3 5 1). Dada a permutação dos vértices φ = (4 1 3 2) das cliques K_F e K_D vamos construir sua vizinhança de acordo com o procedimento.

Seguindo os passos de **ConstViz** tem-se:

- ξ_φ = (3 6 5 2 4 1) pela expressão (2.4);

- $\rho_\xi = (1\ 2\ 6\ 4\ 3\ 5) \circ (3\ 6\ 5\ 2\ 4\ 1) \circ (4\ 6\ 2\ 3\ 5\ 1)$, resultando em $\rho_\xi = (2\ 1\ 4\ 6\ 5\ 3)$;
- $s = 6$ pois temos $max|i - \rho_\xi(i)| = 3$, $i = 1, \ldots, N$;
- para $s = 6$ então $r = 4$, pois o $min\{|6 - \rho_\xi(i)| + |i - 3|\} = 1$, $i = 1, \ldots, N$;

De posse das posições $s = 4$ e $r = 6$, passemos a pesquisar, através de ψ^{-1}, os vértices que devem ser trocados na permutação φ inicial.

$$\begin{cases} \psi^{-1}(6) = (3,4) \\ \psi^{-1}(4) = (2,3) \end{cases} \text{ as combinações de vértices seriam: } \begin{cases} 3 \text{ e } 2 \\ 4 \text{ e } 2 \\ 3 \text{ e } 3 \text{ (sem efeito)} \\ 4 \text{ e } 3 \end{cases}.$$

Portanto, a $Viz(4\ 1\ 3\ 2) = \{(4\ 1\ 2\ 3), (2\ 1\ 3\ 4), (3\ 1\ 4\ 2)\}$. Note que a $|Viz(\varphi)| = 3$ ou 4. Se houver algum vértice em comum nos arcos resultantes da aplicação de ψ^{-1}, como no exemplo 1, esta troca não tem efeito. O REDINV-SA toma como solução eleita, o mínimo local. Basta calcular os custos de cada elemento da $Viz(\varphi)$ e tomar aquele que fornece menor valor, que corresponde a $\varphi = (4\ 1\ 2\ 3)$ com $z' = 452$. O procedimento de busca local retorna φ', com seu respectivo custo z'.

Para que o pseudo-código do algoritmo REDINV-SA seja melhor compreendido, vale a pena relembrar algumas definições apresentadas nas seções anteriores.

Sejam os vetores ordenados $\mathbf{F}^-$ e $\mathbf{D}^+$ que armazenam as entradas das matrizes F e D que definem o $PQA(F, D)$. Considere $Q^* = (F^-)^t(D^+)$ e a classe de problemas que compartilham Q^*, denotada por $RelClass(Q^*)$. O $LimInf = tr(Q^*)$ é o limite inferior universal das soluções viáveis das instâncias $\Im \in RelClass(Q^*)$ e, a soma da diagonal secundária de Q^* é limite superior universal, $LimSup$, para a mesma classe de problemas.

Os parâmetros adotados por Querido em [Que94] para o REDINV-SA são:

- a escala *annealing* é dada pela função usada em (5.11), $T(t+1) = \alpha T(t)$, onde $\alpha = 0.9$;
- $T_0 = LimSup$;
- $M = n$, número de iterações para cada $T(t)$;
- critério de parada é $T(t) < LimInf$;

Com base no procedimento SA e com os parâmetros definidos acima, o REDINV-SA é apresentado abaixo.

REDINV-SA(instância,n);

```
gerar aleatoriamente φ0 ∈ Π_n e calcular z0;
t = 0;
T(0) = LimSup;
Enquanto T(t) > LimInf faça
   Enquanto M ≤ n faça
      ConstViz(φ);
      avaliar Δz = z' − z0;
      Se Δz < 0 então
         φ0 ← φ';
         Se Δz ≥ 0 então
            Se random(0,1) < exp^(−Δz/T(t)) então
               φ0 ← φ';
            FimSe;
         FimSe;
      FimSe;
   FimEnquanto;
   T(t) = α T(t);
   t = t + 1;
FimEnquanto;
```

FimREDINV-SA;

Para que não haja ciclos, a implementação guarda, em uma lista, as soluções que já foram escolhidas. O tamanho desta lista depende da dimensão do problema. O REDINV-SA foi aplicado a uma série de instâncias PQA e os resultados foram satisfatórios e podem ser encontradas em [Que94].

Capítulo 6

Avaliação da qualidade de soluções

O **Teorema das Inversões** mostra que em pares de permutações livremente comparáveis, situadas em níveis distintos do grafo G_{Liv}, o custo da permutação que estiver no nível mais baixo, é obrigatoriamente menor. Em outras palavras, *custos de permutações livremente comparáveis crescem ou decrescem quando os números de inversões dessas permutações crescem ou decrescem, respectivamente.* Embora pares de permutações, representando soluções viáveis para o $PQA(F, D)$, em geral, não sejam livremente comparáveis, este teorema nos estimula a pesquisar se custos mais baixos para uma instância $PQA(F, D)$ possuem suas permutações $\rho \in \Pi_N$ correspondentes com "baixo" número de inversões.

Na tentativa de validar o número de inversões como parâmetro de avaliação da qualidade da solução da instância do Problema Quadrático de Alocação, apresentamos três tipos de testes empíricos: o primeiro teste determina em que altura do grafo G_{Liv} se encontra a permutação $\rho_{\varphi^*} \in \Pi_N$ que corresponde à solução ótima $\varphi^* \in \Pi_n$. Isto equivale a determinar a altura do grafo G_{Liv} em que se encontra a permutação correspondente à melhor solução viável conhecida da instância; o segundo compara a curva descrita pelo comportamento do custo, com a curva descrita pelo número de inversões, sabendo-se que as soluções do PQA são geradas aleatoriamente e fornecem o custo para a primeira curva, enquanto o número de inversões das permutações correspondentes às soluções geradas, fornecem os valores para a segunda curva; finalmente, o terceiro teste utiliza a técnica estatística, conhecida por análise de regressão, considerando como variáveis o custo e o número de inversões de permutações também geradas aleatoriamente, tal como no segundo teste.

6.1 Altura da melhor solução PQA em G_{Liv}

A biblioteca $QAPLIB$ [BKR97] apresenta uma série de instâncias do Problema Quadrático de Alocação exibindo a permutação ótima com seu respectivo custo. Se a instância ainda não foi resolvida otimamente, a $QAPLIB$ sempre fornece o valor do custo da melhor solução viável conhecida, alcançada por algum método heurístico lá especificado, e eventualmente, a respectiva solução.

Não se pode garantir a unicidade da solução ótima de instâncias PQA, pois as entradas das matrizes, em geral, são não distintas, assim como as somas de parcelas diferentes podem resultar em totais iguais. Sendo assim, soluções ótimas de uma mesma instância são representadas por permutações com número de inversões diferentes e portanto, se situam em níveis distintos do grafo G_{Liv}. Tomamos as soluções ótimas fornecidas pela $QAPLIB$, as alcançadas pelo GRASP, apresentadas por Li et al. [LPR94] e as soluções encontradas pelos algoritmos desenvolvidos no capítulo 5 (Grasp Ampliado, Grasp Restrito e o Algoritmo Genético). Verificamos para cada instância, se tais ótimos são distintos e calculamos o número de inversões das permutações associadas a esses ótimos. De acordo com Firmo em [Fi99], estimamos a altura h_ρ do vértice do grafo G_{Liv}, correspondente a ρ, segundo a fórmula

$$h_\rho = \frac{|\Im(\rho)|}{Total_{Ni}}, \tag{6.1}$$

onde $|\Im(\rho)|$ é o número de inversões da permutação ρ e $Total_{Ni} = \frac{N(N-1)}{2} + 1$ é o total de níveis de G_{Liv}.

As tabelas 6.1 e 6.2 relacionam as instâncias em que foram avaliadas as alturas h_ρ. Se não houver diferença entre as soluções ótimas, a altura h_ρ é repetida, caso contrário, a menor altura é destacada em negrito. As tabelas fornecem as alturas calculas com as soluções encontradas na $QAPLIB$, no GRASP desenvolvido por Li et. al [LPR94], o Grasp Ampliado e Algoritmo Genético Híbrido, descritos no capítulo 5.

Ainda com respeito a instâncias com soluções ótimas conhecidas, consideremos as propostas por Hadley, Had12, Had14, Had16 e Had18, que possuem seus ótimos disponíveis na $QAPLIB$ e as permutações correspondentes no grafo G_{Liv} estão, em média, a 31% da altura total do grafo.

instâncias	$QAPLIB$	GRASP	Grasp Ampl.	AG
Nug12	0.3564	**0.3090**	0.3564	0.3090
Nug15	**0.2954**	0.3336	0.3267	0.2954
Nug20	0.3338	0.3338	0.3338	**0.3271**
Rou12	0.2838	0.2838	0.2838	0.2838
Rou15	0.2779	0.2779	0.2779	0.2779
Rou20	0.2970	0.2970	0.2970	não alcançou
Scr12	0.3420	**0.3103**	**0.3103**	**0.3103**
Scr15	0.3349	0.3349	0.3349	0.3349
Scr20	0.3382	0.3382	0.3315	**0.3283**
Els19	0.4036	0.4036	0.4036	0.4036
Kra30a	0.3611	**0.3564**	0.3583	0.3572

Tabela 6.1: Alturas no G_{Liv} dos ótimos das instâncias Nug, Rou, Scr, Els

instâncias	$QAPLIB$	GRASP	Grasp Ampl.	AG
Chr12a	0.3467	0.3467	0.3467	0.3467
Chr12b	0.3649	0.3649	0.3649	0.3649
Chr12c	0.3751	0.3751	0.3751	0.3751
Chr15a	0.4338	0.4338	**0.3946**	0.4338
Chr15b	0.3750	0.3750	0.3750	0.3750
Chr15c	0.4179	0.4179	0.4179	0.4179
Chr18a	0.4097	0.4097	0.4097	**0.3469**
Chr18b	**0.2482**	0.3469	0.5134	0.3472

Tabela 6.2: Alturas no G_{Liv} dos ótimos de instâncias Christofides

Dando prosseguimento a esta análise, consideramos agora, instâncias cujos ótimos ainda não são conhecidos. Tomamos as soluções encontradas pelo GRASP relacionadas em [LPR94] e algumas instâncias resolvidas pelos algoritmos do capítulo 5. O Grasp Restrito é aplicado somente para instâncias de dimensões maiores que 30.

As instâncias de Skorin-Kapov de dimensão 100 chamadas de Sko100a, Sko100b, Sko100c, Sko100d, Sko100e e Sko100f cujos valores das solução viáveis encontradas pelo Grasp Restrito estão bem próximos dos valores apresentados na $QAPLIB$, se localizam, em média, a 41% da altura do grafo G_{Liv}. Como a $QAPLIB$ não transcreve a permutação que gerou o valor da solução viável, não podemos fazer uma comparação. Além disso, Li et. al [LPR94] não aplicaram o GRASP a essas instâncias. Pela não garantia de otimalidade dessas soluções, podemos acreditar que existem melhores soluções e que devem ser representadas por permutações com menor número de inversões. O mesmo ocorre com Sko42, Sko49 e Sko64, tendo

como respectivas alturas 0.371, 0.395 e 0.4034, todas calculadas com as permutações alcançadas pelo Grasp Restrito.

Para a instância proposta por Krarup, Kra30b, não temos registro da solução ótima. Portanto, a melhor viável disponível é a solução GRASP de Li et. al [LPR94], com $h_\rho = 0.3573$, cujo valor coincide com o valor apresentado na $QAPLIB$.

Considerando as instâncias propostas por Steinberg, a permutação alcançada pelo GRASP, disponíveis em [LPR94] para Ste36a, possui a sua correspondente ρ cuja altura é $h_\rho = 0.3855$ de custo 9588. O Grasp Restrito para esta mesma instância obteve uma solução viável com valor 9724 e $h_\rho = 0.3718$, menor que a anterior. Podemos esperar que a solução ótima de Ste36a possua uma permutação correspondente ρ cuja altura no grafo G_{Liv} esteja em torno desses valores mencionados. A Ste36b, foi analisada da mesma forma, isto é, o GRASP [LPR94] alcançou uma permutação cuja representante em G_{Liv} possui altura 0.3684 e custo 15852, porém, a menor altura registrada é 0.3612 e corresponde a solução de valor 16194, encontrada pelo Grasp Restrito. Portanto, as melhores soluções viáveis para estas instâncias estão a uma altura média de 37.2% do grafo G_{Liv}.

A tabela 6.3 mostra as médias das alturas para os grupos de instâncias. Analisando a tabela, temos a média total das alturas em torno de 31.676% para as instâncias cujos ótimos são conhecidos e para os grupos em que temos apenas as melhores soluções viáveis conhecidas, a média é de 39.12%, um pouco acima da anterior. O fato da segunda média ser superior à primeira nos leva a esperar que as melhores soluções conhecidas podem não ser realmente as ótimas.

Instância ótimo	média	Instância viável	média
Had	31%	Sko	38.98%
Nug	31.05%	Sko100	41%
Rou	28.02%	Ste	37.50%
Scr	32.45%	-	-
Chr	35.86%	-	-
Méd. T	31.676%	Méd. T	39.12%

Tabela 6.3: Médias das alturas dos ótimos e melhores viáveis

6.2 Gráficos de Comparação

Nesta seção fazemos a análise da relação entre custos e inversões através de duas curvas, uma representando os custos das soluções e a outra, os números de inversões das permutações a elas correspondente. O objetivo é mostrar que existe uma compatibilidade no crescimento e decrescimento entre elas. Para plotar os gráficos, foram geradas 50 soluções $\varphi_i \in \Pi_n$, calculamos os custos $C(\varphi_i)$ e o número de inversões $|\Im(\rho_i)|$, onde $\rho_i \in \Pi_N$ é a permutação que representa φ_i no grafo G_{Liv}, $i = 1, \ldots, 50$. A primeira curva é descrita pelos pares $(i, C(\varphi_i))$ e a segunda, pelos pares $(i, |\Im(\rho_i)|)$. Foram escolhidas três instâncias para mostrar essa comparação, Rou20, Nug20 e Chr18a. Estas instâncias foram escolhidas em um universo de várias outras da $QAPLIB$ por serem capazes de bem representar os resultados encontrados nos testes realizados. Os seus respectivos gráficos estão ilustrados nas figuras 6.1, 6.2 e 6.3. As 50 permutações ρ_i de cada instâncias não são todas livremente comparáveis entre si, estimulando ainda mais, a busca de soluções PQA que possuam representantes em G_{Liv}, com baixo número de inversões, pois existe uma grande compatibilidade entre as curvas.

Para a instância Rou20, figura 6.1, pode-se observar que 92% dos pontos apresentam uma coincidência no comportamento das curvas. Somente os pontos 14, 24, 26 e 34, dentre os 50 gerados, discordam no crescimento ou decrescimento, por exemplo, $C(\varphi_{13}) > C(\varphi_{14})$ ao passo que $|\Im(\rho_{13})| < |\Im(\rho_{14})|$. Esta relação direta entre custos e inversões também pode ser vista na seção 6.3, quando é feita a análise de regressão linear.

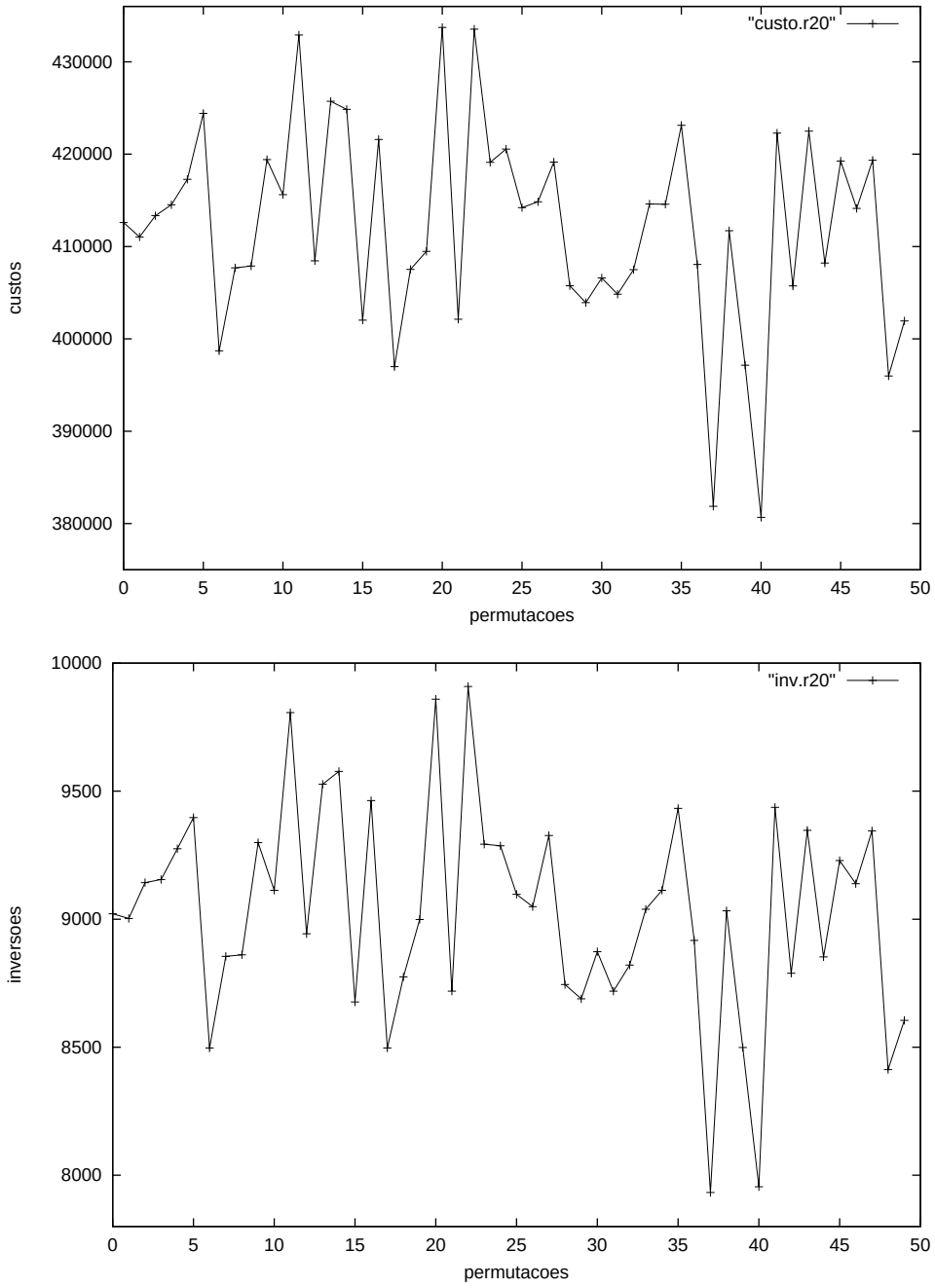


Figura 6.1: Comportamento dos custos e números de inversões de Rou20

As curvas da figura 6.2 representam custos e inversões da instância Nug20. O comportamento no crescimento e decrescimento das curvas coincidem em 43 pontos, dentre os 50 plotados, isto é, 86% dos pontos. Pode-se considerar que as curvas são compatíveis.

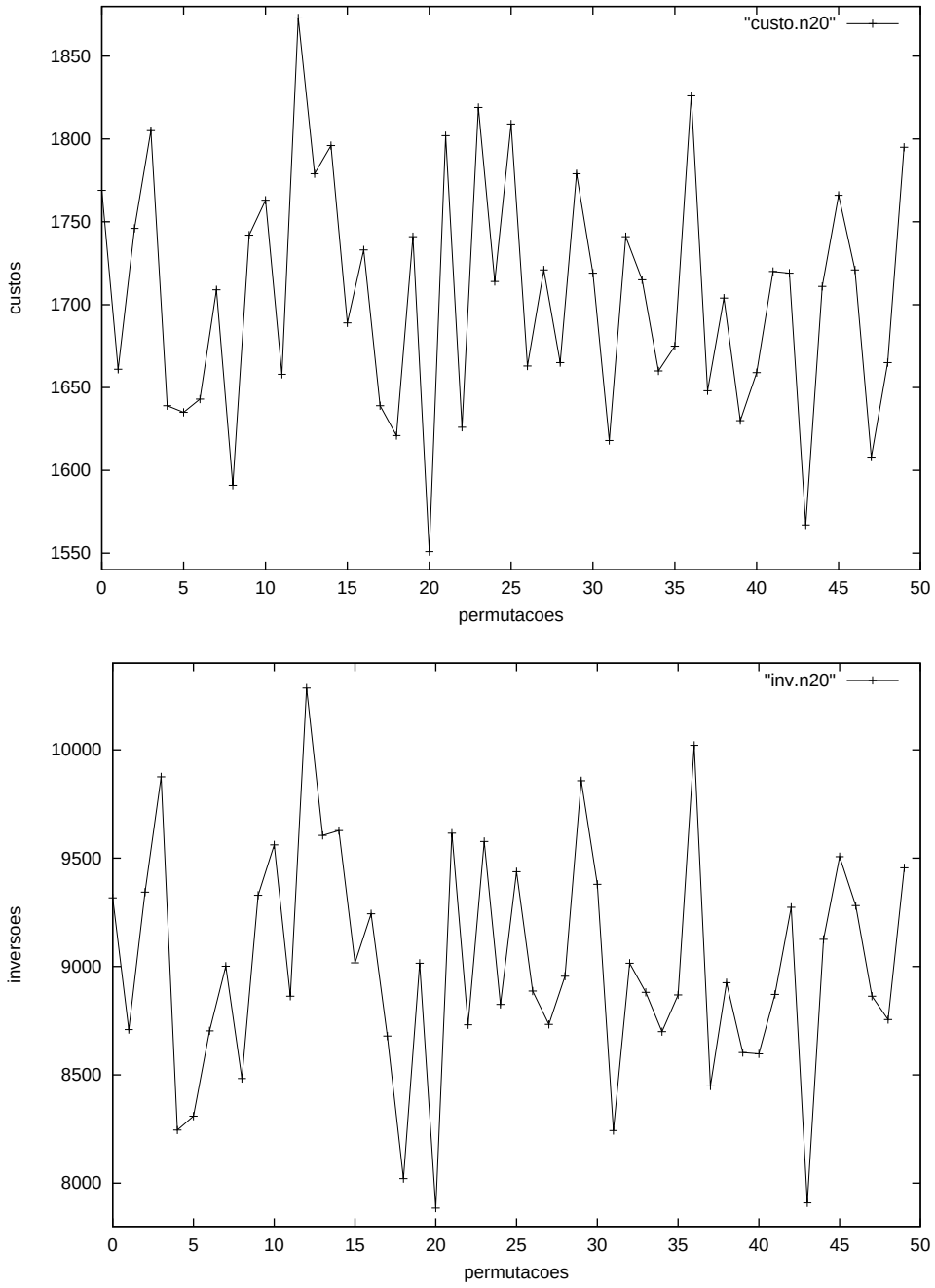


Figura 6.2: Comportamento dos custos e números de inversões de Nug20

A última instância escolhida para a comparação dos gráficos é a Chr18a. Observamos uma menor compatibilidade das curvas, onde 50% dos pontos apresentam coincidência no comportamento. As instâncias propostas por Christofides, disponíveis na $QAPLIB$, em geral, são difíceis de serem resolvidas. O próprio Christofides desenvolveu um algoritmo que explora a estrutura de árvore destas instâncias, [CB89].

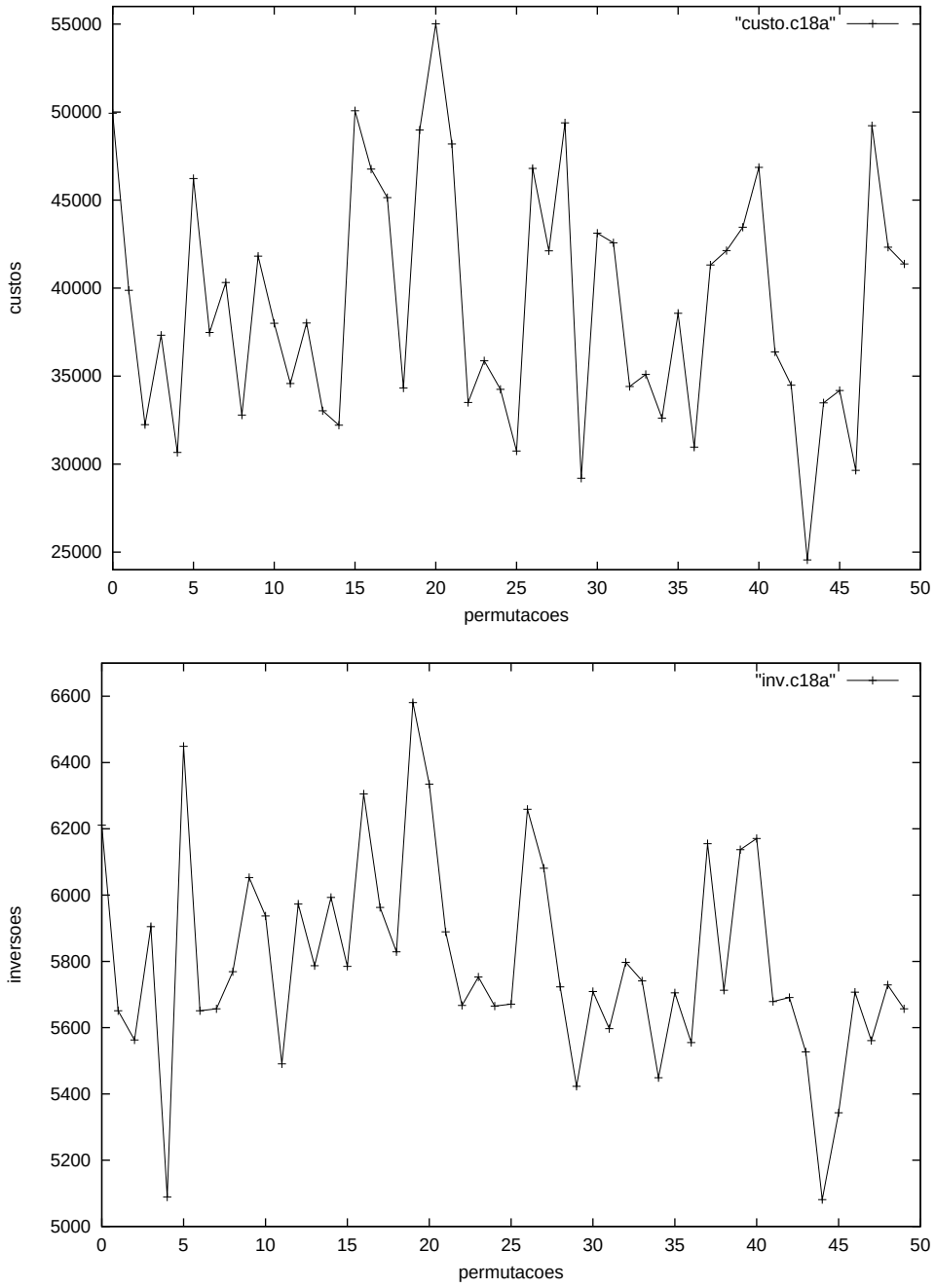


Figura 6.3: Comportamento dos custos e números de inversões de Chr18a

6.3 Análise de Regressão Linear

Um dos objetivos da análise de regressão é predizer o valor de uma variável (variável dependente) dado que seja conhecido o valor de uma variável associada (variável independente). Contudo, em nossos estudos, fazemos uso desta técnica para confirmar a tendência de crescimento dos custos das soluções de uma instância $PQA(F, D)$, quando os números de inversões de suas permutações associadas crescem.

Para construir a amostra das variáveis, geramos aleatoriamente 150 soluções $\varphi \in \Pi_n$ do Problema Quadrático de Alocação, calculamos os custos $C(\varphi)$ e o número de inversões $|\Im(\rho)|$, onde $\rho \in \Pi_N$ é a permutação correspondente à φ no grafo G_{Liv}. Consideramos como variáveis independentes os números de inversões, $|\Im(\rho)|$, e os valores dos custos, $C(\varphi)$, como variáveis dependentes.

Um diagrama de dispersão é um gráfico no qual cada ponto plotado representa os valores das variáveis independentes e dependentes. O valor da variável independente x está no eixo horizontal e o valor da variável dependente y, no eixo vertical. Para a construção dos nossos diagramas de dispersão, ordenamos os números de inversões, em ordem crescente, e plotamos, no plano cartesiano, os pontos $(|\Im(\rho_i)|, C(\varphi_i))$, $i = 1, \ldots, 150$. Se o diagrama indicar uma relação que de modo geral é linear, então ajusta-se os dados por uma linha que forneça a melhor aproximação. A estimação dos parâmetros desta linha é determinada pelo Método dos Mínimos Quadrados, cuja fórmula é dada por:

$$y = a + bx, \tag{6.2}$$

$$b = \frac{\sum_{i=1}^{n} X_i Y_i - n\overline{X}\ \overline{Y}}{\sum_{i=1}^{n} {X_i}^2 - n\overline{X}^2} \quad \text{e} \tag{6.3}$$

$$a = \overline{Y} - b\overline{X}, \tag{6.4}$$

onde X_i = valores de $|\Im(\rho_i)|$ e Y_i = valores de $C(\varphi_i))$, com n = número de amostras. As médias $\overline{X}$ e $\overline{Y}$ são as respectivas média aritmética de X_i e Y_i.

Podemos também calcular o coeficiente de determinação que indica a porcentagem da variabilidade de Y que pode ser explicada estatisticamente pelo conhecimento de X. Para isso, temos a fórmula

$$r^2 = \frac{a\ \sum_{i=1}^{n} Y_i + b\ \sum_{i=1}^{n} X_i Y_i - n\overline{Y}^2}{\sum_{i=1}^{n} Y_i^2 - n\ \overline{Y}^2}. \tag{6.5}$$

Muito embora o coeficiente de determinação r^2 seja relativamente fácil de interpretar a significância, ele não pode ser testado estatisticamente. Ainda há outro coeficiente, o de correlação para dados amostrais, determinado por $r = \sqrt{r^2}$, que pode ser testado estatisticamente, pois está incluído em uma estatística de teste que é distribuída segundo uma distribuição *t-Student*, quando a correlação populacional

ρ é igual a zero. De modo geral, a hipótese nula de interesse é que o coeficiente de correlação da população seja igual a zero, isto é, $\rho = 0$. Se esta hipótese for rejeitada, considerando a significância α estipulada, podemos concluir, efetivamente, que existe uma relação entre as variáveis. Dado que (1) as variáveis são aleatórias, (2) a relação entre as duas variáveis é linear e (3) as variáveis dependentes formam uma distribuição normal, a seguinte estatística amostral envolvendo r é distribuída com uma distribuição *t-Student* com $n - 2$ graus de liberdade, quando $\rho = 0$:

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}. \tag{6.6}$$

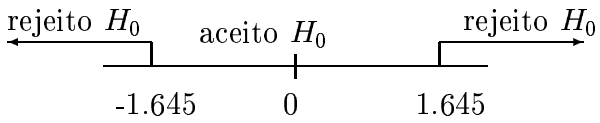
Como nossas amostras são relativamente grandes, $n = 150$, podemos aproximar distribuição *t-Student* pela distribuição normal. O parâmetro t_c crítico, considerando o nível de significância $\alpha = 0.05$, é tabelado em $t_c = 1.645$.

O exemplo a seguir, facilita o entendimento deste teste de hipótese.

Exemplo 6.1 *Considere a instância Chr12a,* $n = 150$*,* $\alpha = 0.05$ *e* $t_c = 1.645$.

Pela expressão (6.5) tem-se $r^2 = 0.3882$ e consequentemente $r = 0.623$. A expressão (6.6) fornece o valor $t = 9.69$. Os intervalos de aceitação ou rejeição de H_0 são demarcados por t_c, conforme o esquema abaixo.

$$H_0 : \rho = 0 \qquad H_1 : \rho \neq 0$$



Como $t > t_c$, rejeita-se a hipótese H_0, existindo uma relação linear entre as variáveis custo e número de inversões.

Dentre as 27 instâncias da $QAPLIB$ escolhidas para a análise de regressão, transcritas nas tabelas 6.4 e 6.5, tem-se para cada uma delas que (1) os custos são variáveis aleatórias, (2) as relações entre custos e números de inversões são lineares e (3) os custos formam uma distribuição normal, exceto para a instância Chr18b.

instâncias	reta	determinação	correlação	$\rho = 0$
Chr12a	$y = -14101.137 + 34.752x$	0.3882	0.623	rejeitou
Chr12b	$y = -9455.944 + 30.704x$	0.2321	0.482	rejeitou
Chr12c	$y = -12769.849 + 33.077x$	0.3755	0.613	rejeitou
Chr18a	$y = -20739.074 + 10.336x$	0.2865	0.535	rejeitou
Chr18b	$y = 1389.720 + 0.229x$	0.0105	0.102	aceitou
Nug12	$y = 150.596 + 0.242x$	0.6887	0.830	rejeitou
Nug15	$y = 296.056 + 0.181x$	0.8259	0.909	rejeitou
Nug20	$y = 638.911 + 0.119x$	0.8073	0.898	rejeitou
Nug30	$y = 1172.417 + 0.061x$	0.7544	0.869	rejeitou
Rou12	$y = 73988.750 + 74.817x$	0.9835	0.992	rejeitou
Rou15	$y = 106151.563 + 46.783x$	0.9769	0.988	rejeitou
Rou20	$y = 169075.578 + 26.921x$	0.9852	0.993	rejeitou
Scr12	$y = 1119.821 + 26.927x$	0.3478	0.590	rejeitou
Scr15	$y = 2566.405 + 17.680x$	0.4070	0.638	rejeitou
Scr20	$y = 22460.293 + 10.240x$	0.2694	0.519	rejeitou
Kra30a	$y = 13173.996 + 1.147x$	0.6174	0.790	rejeitou
Kra30b	$y = 9412.174 + 1.253x$	0.5749	0.760	rejeitou
Ste36a	$y = 4398.343 + 0.068x$	0.0537	0.230	rejeitou
Ste36b	$y = -1414.661 + 0.432x$	0.0437	0.220	rejeitou

Tabela 6.4: Análise de Regressão para instâncias PQA

Através das equações (6.2), (6.3), (6.4), (6.5) e (6.6) apresentamos a tabela 6.4 que contém as retas de regressão, os coeficientes de determinação e os coeficientes de correlação para algumas instâncias disponíveis na $QAPLIB$. A última coluna desta tabela fornece o resultado do teste de hipótese, avaliando a significância dos respectivos parâmetros. Somente a instância Chr18b não satifaz a hipótese (3) impedindo assim, garantir estatisticamente a significância do parâmetro b da reta de regressão $y = 1389.720 + 0.229x$. Nota-se que o coeficiente $b = 0.229$ é positivo, indicando uma relação linear direta entre entre custos e número de inversões, porém sem garantias estatísticas.

Felizmente, os coeficientes encontrados pela expressão (6.3), para as instâncias escolhidas são positivos, indicando que há uma relação direta entre o número de inversões e custos das permutações cujos testes de hipóteses validaram a relação linear e direta, isto é, os parâmetros são significantivamente diferentes de zero. Os gráficos das figuras 6.4, 6.5 e 6.6 ilustram o fato relatado.

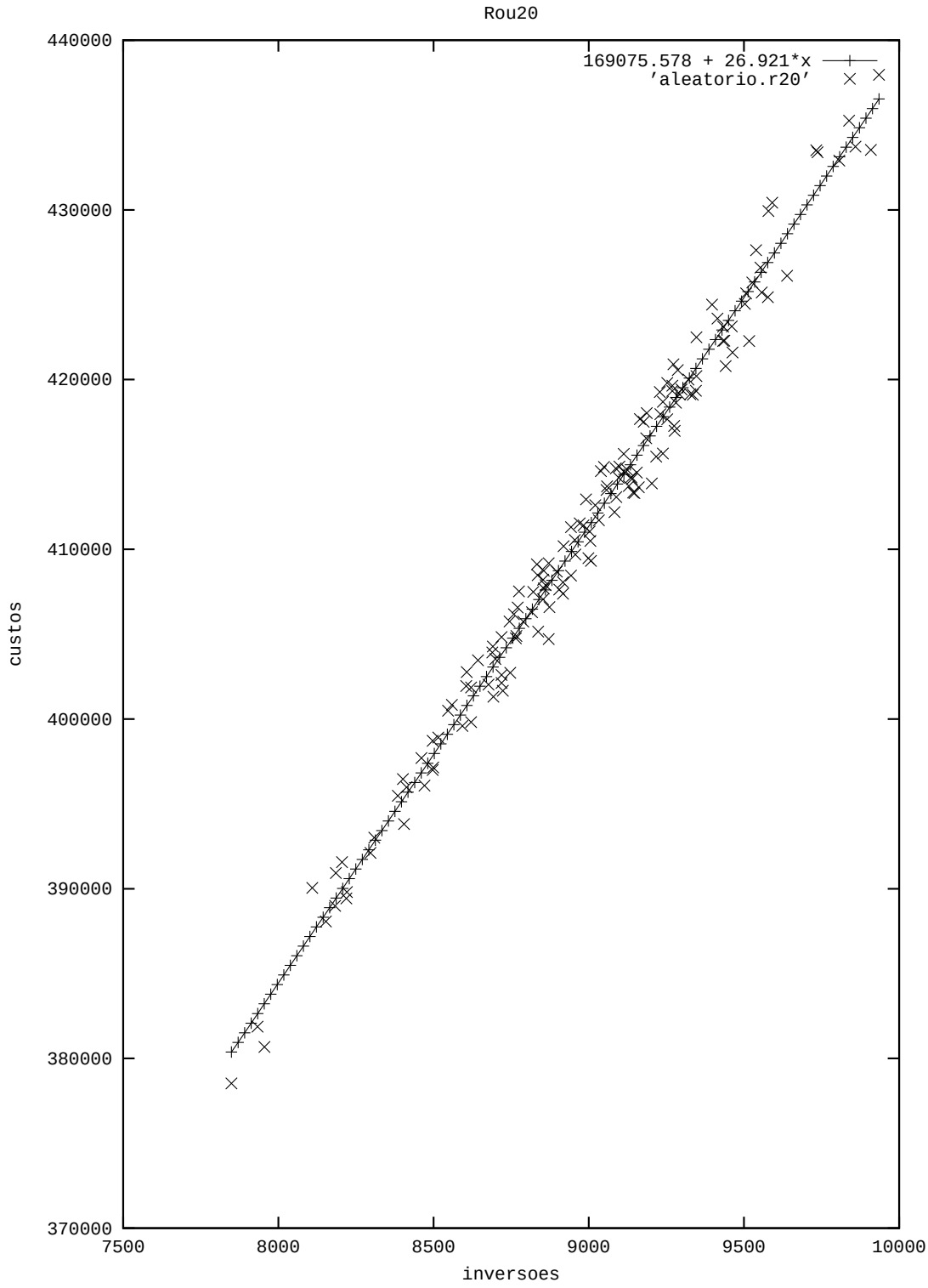


Figura 6.4: Instância Rou20.

O primeiro gráfico exemplifica a instância Rou20 com $r = 0.993$, muito próximo de 1, mostrando uma relação muito forte entre as variáveis.

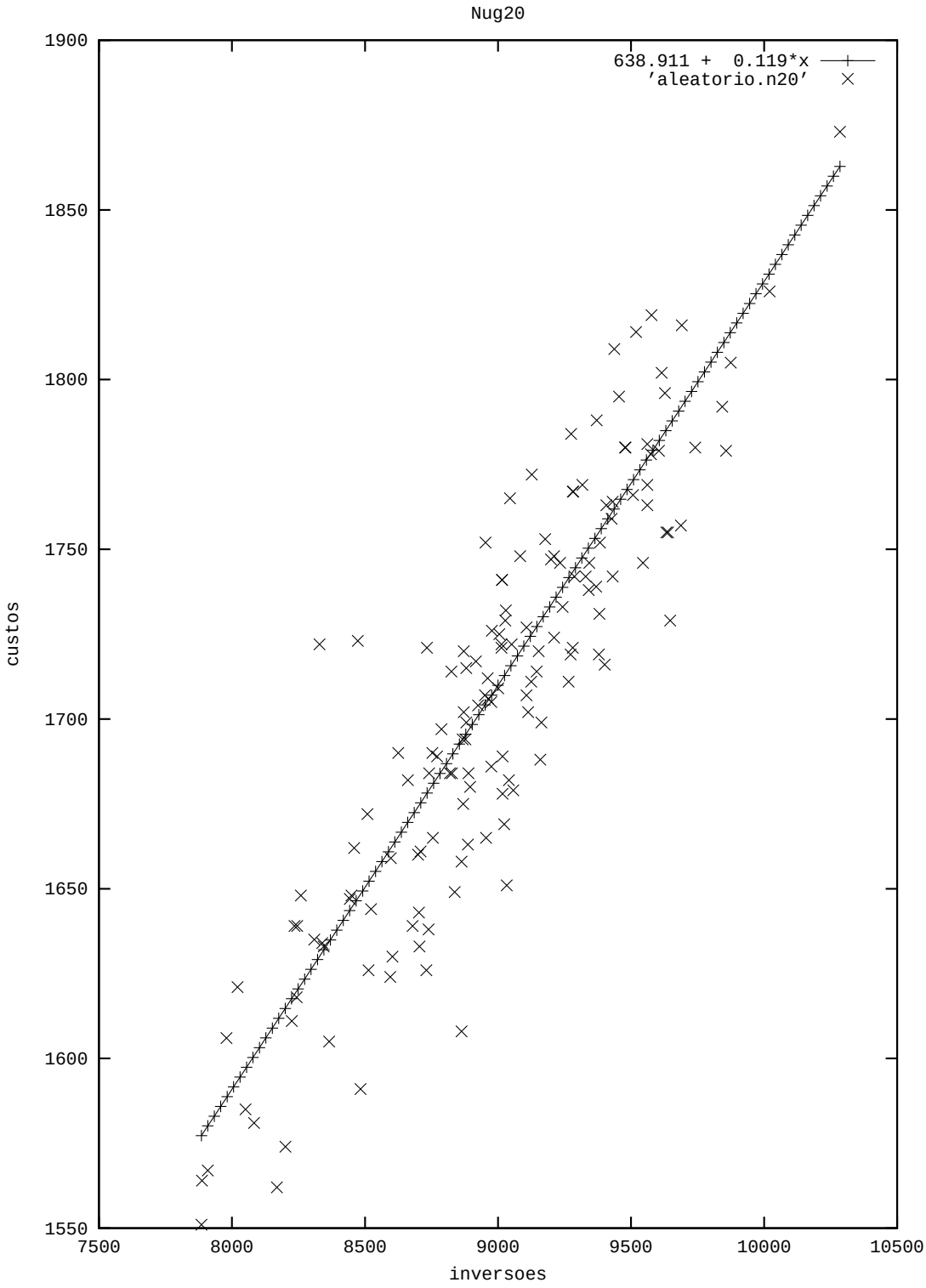


Figura 6.5: Instância Nug20.

No segundo, ilustra-se Nug20, cujo coeficiente de correlação $r = 0.898$ indica a tendência de crescimento do custo com o número de inversões, tal como em Rou20.

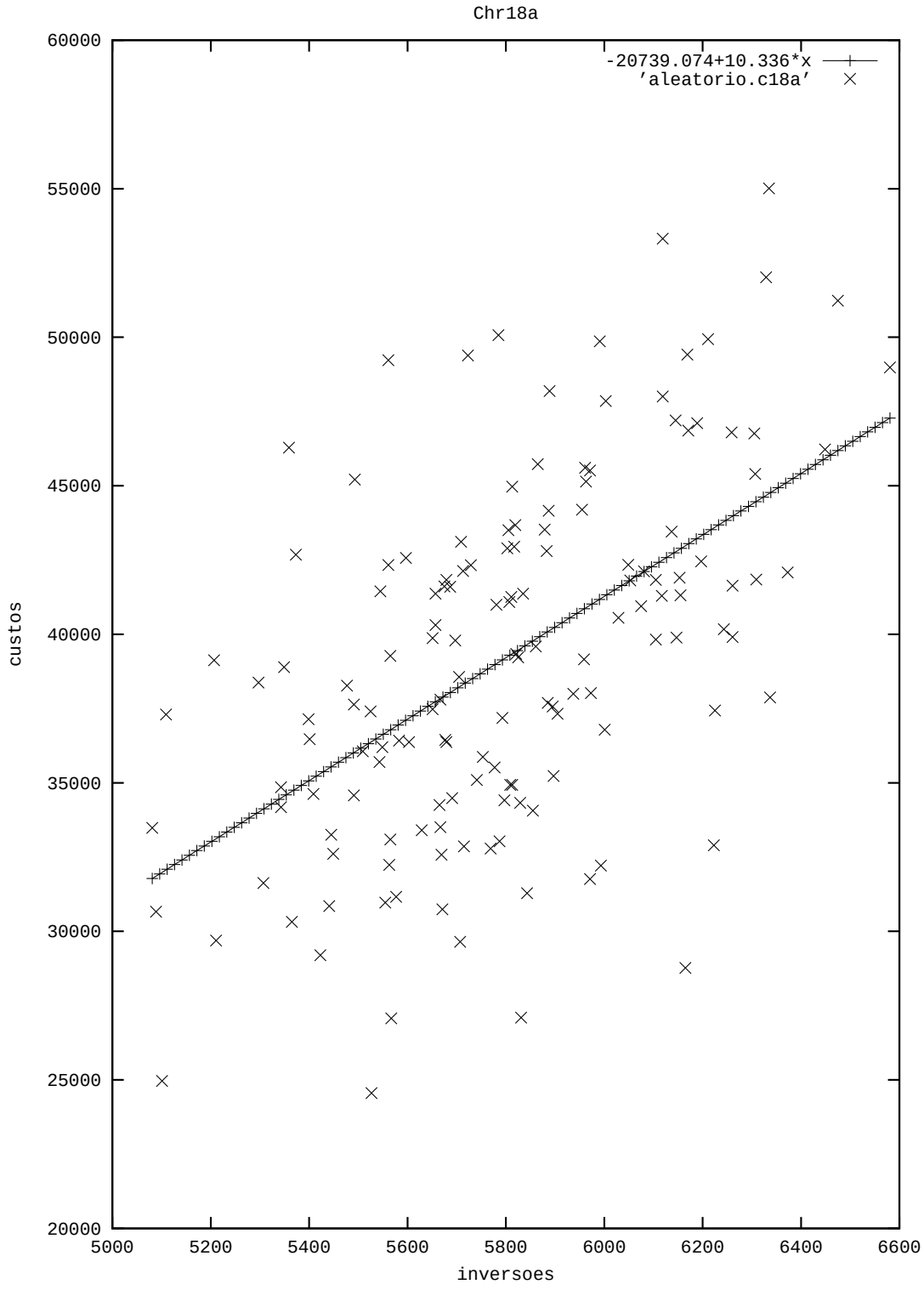


Figura 6.6: Instância Chr18a.

O gráfico da figura 6.6 apresenta uma relação direta estatisticamente comprovada pelo teste de hipótese, no caso da instância Chr18a, apesar de que pela simples observação deste gráfico seria difícil afirmar que os custos crescem com o número de

instâncias	reta	determinação	correlação	$\rho = 0$
Sko42	$y = 3826.173 + 33.598x$	0.6996	0.836	rejeitou
Sko49	$y = 5091.335 + 27.302x$	0.7201	0.849	rejeitou
Sko100a	$y = 40993.691 + 7.878x$	0.7644	0.874	rejeitou
Sko100b	$y = 33223.219 + 9.337x$	0.7203	0.849	rejeitou
Sko100c	$y = 33928.121 + 8.712x$	0.6519	0.807	rejeitou
Sko100d	$y = 40099.258 + 7.795x$	0.7645	0.874	rejeitou
Sko100e	$y = 41491.836 + 7.627x$	0.6355	0.797	rejeitou
Sko100f	$y = 39497.719 + 7.829x$	0.6792	0.824	rejeitou

Tabela 6.5: Análise de Regressão para instâncias Skorin-Kapov

inversões.

Tal como na tabela 6.4, apresentamos os resultados na tabela 6.5 para instâncias maiores, os exemplos de Skorin-Kapov disponíveis na $QAPLIB$. Para calcular os coeficientes a e b da reta para as instância de dimensão $n = 100$, dividimos os números de inversões por 1000 para não haver problemas de precisão de máquina.

Nestes exemplos os custos estão diretamente relacionados com os números de inversões garantidos pelo teste de hipótese descrito no exemplo 6.1. O gráfico ilustrado na figura 6.7 mostra o diagrama de dispersão em conjunto com a linha de regressão calculada para a instância Sko42. Este mesmo gráfico serve para exemplificar as outras instâncias Skorin-Kapov analisadas, dado a semelhança nos coeficiente de correlação apresentados na tabela 6.5, indicando que possuem um mesmo comportamento.

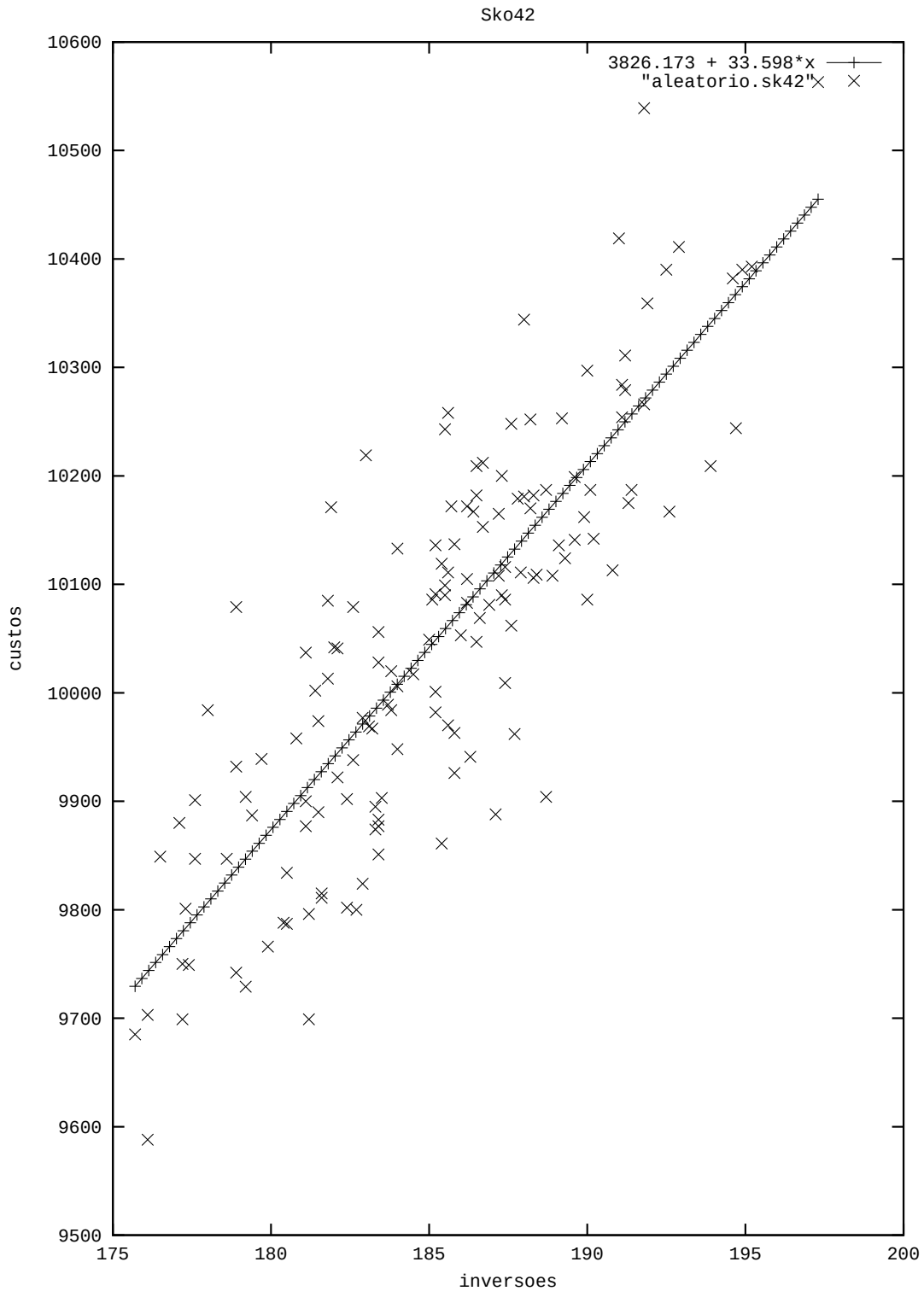


Figura 6.7: Instância Sko42.

Capítulo 7

Conclusão

Apresentamos uma nova prova do **Teorema da Ordenação Parcial Livre** muito mais simples que a prova dada em Abreu [Ab84]. Este teorema determina pares de permutações livremente comparáveis, através da construção de bijeções. Tais bijeções precisam atender critérios e propriedades específicas e em geral, pode existir um grande número delas. Aqui demos uma importante contribuição. Definimos e construímos uma **Bijeção Canônica**, pelo algoritmo chamado **AlgBijeção**, descrito na seção 3.2 e provamos que ela é suficiente para satisfazer o referido teorema, permitindo assim, a comparação livre de pares de permutações em tempo polinomial.

Uma segunda e importante contribuição teórica desta tese está no fato de que demonstramos o **Teorema das Inversões** que garante que *custos de permutações livremente comparáveis "crescem" ou "decrescem" no mesmo sentido que o número de inversões dessas permutações.* Este teorema prova uma conjectura proposta na tese de Abreu [Ab84]. Baseado neste resultado, ainda como conjectura, Querido [Que94] já havia desenvolvido um **Simulated Annealing** para resolver o PQA. Os resultados obtidos pelo seu algoritmo, chamado de **REDINV** pareciam confirmar a validade da conjectura. A nossa demonstração encerra esta questão.

Como contribuições parciais obtidas para provar o **Teorema das Inversões**, podemos destacar o teorema 4.1, o teorema 4.2 e a definição do grafo intermediário G'_{Inv}, isto é, $G_{Inv} \subset G'_{Inv} \subset G_{Liv}$. Desenvolvemos também alguns algoritmos, a citar, **ConstGNi** que constrói o Grafo das Inversões por níveis e o **AlgPath** que encontra um caminho entre permutações livremente comparáveis.

Ao lado das contribuições teóricas, apresentamos duas versões para o GRASP desenvolvido por Li et al. [LPR94]: o Grasp Ampliado e o Grasp Restrito. Na primeira,

fizemos uma expansão no espaço de busca local, utilizando não só as soluções de melhor custo para construir a vizinhança 2-trocas, mas também as soluções cujo número de inversões das permutações associadas fosse menor. Desta forma, foi possível encontrar a solução ótima da instância Nug30, comprovada a otimalidade por Anstreicher e Brixius, [AB00], em junho de 2000 que há 32 anos desafiava os pesquisadores. A segunda modificação foi feita de forma a filtrar as soluções iniciais, geradas na fase inicial do GRASP, que parecem estar "longe" do ótimo. Para isso, foi definido o custo normalizado que estima a distância de tais soluções em relação ao limite inferior. Utilizamos o limite inferior universal das instâncias da família $RelClass(Q^*)$ por ser facilmente calculado a partir do traço da matriz Q^*. Apresentamos esse limite no capítulo dedicado a relaxação linear. Esta modificação demonstrou uma melhora expressiva no tempo computacional do GRASP original com uma perda na qualidade das soluções encontradas muito pequena.

O uso do número de inversões da permutação associada a uma dada solução do PQA como parâmetro de medida de qualidade da referida solução é questão que vem sendo estudada na nossa linha de pesquisa, desde a conjectura proposta por Abreu [Ab84] e seguindo trabalhos tais como as teses de Querido [Que94], Firmo [Fi99] e Moreira [Mo98]. No último capítulo desta tese apresentamos uma análise sobre o desempenho deste parâmetro para este fim, por meio de três abordagens diferentes. No primeiro, concluímos que a solução ótima de qualquer instância PQA se encontra, em média, a 32% da altura do grafo G_{Liv}. Para as instâncias que ainda não foram resolvidas otimamente tem-se para as melhores soluções viáveis conhecidas uma altura, em média, 39% do grafo. O fato da primeira média ser menor que a segunda é perfeitamente aceitável, pois acreditamos que as soluções ótimas dessas instâncias, ainda não encontradas, possuam representantes em G_{Liv} com menor número de inversões. Fizemos o segundo teste por comparação das curvas que descrevem os comportamentos dos custos das soluções e dos números de inversões das permutações correspondentes em G_{Liv}, e observamos que em média, 76% dos pontos plotados apresentam a mesma trajetória. A análise de regressão linear, terceiro e último teste, mostra que os custos estão diretamente relacionados com o número das inversões das permutações associadas, dado que encontramos os coeficientes da reta sempre positivos e significativamente diferentes de zero. Assim, apesar do Teorema das Inversões garantir que pares de custos de permutações livremente comparáveis crescem com o número de inversões, verificamos que mesmo em

relação a pares de permutações não livremente comparáveis, é possível manter esta tendência na maioria dos casos.

Como trabalhos futuros, sugerimos:

- Elaborar um *Simulated Annealing* baseado no REDINV-SA, desenvolvido por Querido em sua tese de doutorado [Que94] com o objetivo de aumentar o número de soluções da vizinhança.
- Utilizar o conceito de soluções pseudo-viáveis, descrito na seção 2.2.1 em técnicas meta-heurísticas. Este conceito amplia o espaço de busca de soluções do PQA no grafo G_{Liv}, de acordo com a propriedade 2.1.
- Aproveitar a teoria algébrica aqui determinada para informar outras meta-heurísticas tais como Busca Tabu e Colônia de Formiga.
- Analisar as instâncias de uma mesma família $RelClass(Q^*)$, com respeito a teoria algébrica estudada nesta tese, a partir de uma instância conhecida.
- Baseando-se no fato de que as curvas dos custos da maioria das instâncias da $QAPLIB$ apresentam distribuição normal ou aproximadamente normal, utilizar parâmetros como variância dos custos, [ABQG00], para estimar limites inferiores para o PQA, através do Teorema do Limite Central.

Bibliografia

[Ab84] Abreu, N.M.M., *Um estudo algébrico e combinatório do problema quadrático de alocação segundo Koopmans e Beckmann*, Tese de D.Sc., Programa de Engenharia de Produção - COPPE/UFRJ, Brasil, (1984).

[ABQG00] Abreu, N.M.M., Boaventura-Netto, P.O., Querido, T.M., Gouvêa, E.F., *Classes of quadratic assignment problem instances: isomorphism and difficulty measure using a statistical approch*, submetido à Discrete Applied Mathematics.

[AOT95] Ahuja, R.K., Orlin, J.B. e Tivari, A., *A greedy genetic algorithm for the quadratic assignment problem*, Working paper, Sloan School of Management, MIT, Cambridge, MA, (1995).

[AQB99] Abreu, N.M.M., Querido, T.M., Boaventura-Netto, P.O., *REDINV-SA: A simulated annealing for the quadratic assignment problem*, RAIRO Operations Research, vol. 33 (1999), pp. 249-273.

[AB00] Anstreicher, K.M. and Brixius, N.W, *A new bound for the quadratic assignment problem based on convex quadratic programming*, disponível no site http://www-unix.mcs.anl.gov/metaneos/nug30/.

[BT94] Battiti, R. e Tecchiolli, G., *The reactive tabu search*, ORSA Journal on Computing, vol. 6 (1994), pp. 126-140.

[BBM93a] Beasley, D., Bull, D.R. e Martin, R.R., *An Overview of Genetic Algorithms: Part 1, Fundamentals*, University Computing, vol. 15 (1993), pp. 58-69.

[BBM93b] Beasley, D., Bull, D.R. e Martin, R.R., *An Overview of Genetic Algorithms: Part 2, Research Topics*. University Computing, vol. 15 (1993), pp. 170-181.

[BCPP98] Burkard, R.E., Çela, E., Pardalos, P.M. e Pitsoulis, L.S., *The quadratic assignment problem*. In Handbook of Combinatorial Optmization, vol. 3, D.-Z. ZHU and P.M. Pardalos, Eds., Kluwer, pp. 241-337 (1998).

[BC97] Burkard, R.E. e Çela, E., *Quadratic and Three-Dimensional Assignments*, Annotated Bibliographies in Combinatorial Optmization, John Wiley & Sons, New York, primeira ed., pp. 373-391 (1997).

[BKR97] Burkard, R.E., Karisch, S.E. e Rendl, F., *QAPLIB - A Quadratic Assignment Problem Library*, Journal of Global Optimization, vol. 10 (1997), pp. 391-403. Acesso internet:
http://www.imm.dk/ sk/qaplib/

[BR83] Burkard, R.E. e Rendl, F., *A termodynamically motivated simulation procedure for combinatorial optimization problems*, European Journal of Operations Research, vol. 17 (1983), pp. 169-174.

[BS78] Burkard, R.E. e Stratman, K.-H., *Numerical investigations on quadratic assignment problems*, Naval Research Logistics Quartely, vol. 25 (1978), pp. 129-148.

[Bur91] Burkard, R.E., *Locations with spatial interactions: the quadratic assignment problem*, Discrete Locations Theory, John Wiley & Sons, New York, pp. 387-437 (1991).

[CM92] Carraresi, P. e Malucelli, F., *A new lower bound for the quadratic assignment problem*, Operations Research, vol. 40 (1992), supplement 1, pp. 22-27.

[Ce98] Çela, E., *The Quadratic Assignment Problem - theory and algorithms*, Kluwer Academic Publishers, Series in Combinatorial Optimization, vol 1, (1998).

[CSk93] Chakrapani, J. e Skorin-Kapov, J., *Massively parallel tabu search for the quadratic assignment problems*, Annals of Operations Research, vol. 41 (1993), pp. 327-342.

[CB89] Christofides, N. e Benavent, E., *An exact algorithm for the quadratic assignment problem on a tree*, Operations Research, vol. 37 (1989), pp. 760-768.

[CP94] Clausen, J. e Perregaard, M., *Solving large quadratic assignment problems in parallel*, Technical Report DIKU TR-94-22, DCS, University Copenhagen, Denmark, aceito para Computacional Optimization and Applications, (1994),

[Co90] Connolly, D.T., *An improved annealing scheme for the QAP*, European Journal of Operations Research, vol. 46 (1990), 93-100.

[Dar1859] Darwin, C.R., *On the Origin of Species*, Londres, Penguin Classics (1859).

[DMM97] Dell'Amico, M., Maffioli, F. e Martello, S., *Annotated Bibliographies in Combinatorial Optmization*, John Wiley & Sons, New York, primeira ed. (1997).

[Ed77] Edwards, C.S., *The derivation of a greedy approximator for the Koopmans e Beckmann quadratic assignment problem*, Proc. CP77 Combinatorial Programming Conference, (1977), pp. 55-86.

[Ed80] Edwards, C.S., *A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem*, Mathematical Programming Study, vol. 13 (1980), pp. 35-52.

[El77] Elshafei, A.N., *Hospital layout as a quadratic assignment problem*, Operations Research Quarterly, vol. 28 (1977), pp. 167-179.

[FBR87] Finke, G., Burkard, R.E. e Rendl, F., *Quadratic assignment problems*, Annals of Discrete Mathematics, vol. 31 (1987), pp. 61-82.

[FF94] Fleurent, C. e Ferland, J.A., *Genetic hybrids for the quadratic assignment problem*, DIMACS Series Discr. Math. Theor. Comp. Sci., vol. 16 (1994), pp. 173-187.

[Fi99] Firmo, A.L.M., *Geradores de Instâncias do Problema Quadrático de Alocação*, Tese de M.Sc., Programa de Engenharia de Produção - COPPE/UFRJ, Brasil, (1999).

[GTD97] Gambardella, L.M., Taillard, E.D. e Dorigo, M., *Ant Colonies for the QAP*, Technical Report IDSIA97-4, IDSIA, Laguno, Switzerland, (1997).

[GMBR95] Garcia, B.B., Machado, F., Boeres, M.C. e Rangel, M.C., *Aplicação de algoritmos genéticos no problema de alocação de disciplinas a professores*

de um departamento, Anais XXVII Simpósio Brasileiro de Pesquisa Operacional, SBPO, (1995), pp. 138-142.

[GP66] Gavett, J.W. e Plyter, N.V., *The optimal assignment of facilities to locations by branch and bound*, Operations Research, vol. 14 (1966), pp. 210-232.

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

[Gr81] Graham, A., *Kronecker products and matrix calculus: with applications*, Halsted Press, Toronto, (1981).

[HRW90] Hadley, S.W., Rendl, F. e Wolkowicz, H., *Bounds for the quadratic assignment problem using continuous optmization techniques*, Integer Programming and Combinatorial Optmization, University of Waterloo Press, pp. 237-248, (1990).

[Hol75] Holand, J., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, (1975).

[KCCE98] Karisch, S.E., Çela, E., Clausen, J. e Espersen, T., *A dual framework for lower bounds of the quadratic assignment problem based on linearization*, Technical Report IMM-REP-1998-02, Department of Mathematical Modeling, Technical University of Denmark, (1998).

[KGV83] Kirkpatrick, S., Gelatt, C.D. e Vecchi, M.P., *Optimization by Simulated Annealing*, Science, vol. 20 (1983), pp. 671-680.

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

[KB57] Koopmans, T.C. e Beckmann, M.J., *Assignment problems and the location of economics activities*, Econometrica, vol. 25 (1957), pp. 53-76.

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

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

[La63] Lawler, E., *The quadratic assignment problem*, Management Science, vol. 9 (1963), pp. 586-599.

[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, vol. 16 (1994), pp. 237-261.

[LPRR94] Li,Y., Pardalos, P.M., Ramakrishnam, K.G. e Resende, M.G.C., *Lower bound for the quadratic assignment problem*, Annals of Operations Research, vol. 50 (1994), pp. 387-411.

[LM86] Lundy, H. e Mees, A., *Convergence of an annealing algorithm*, Mathematical Programming, vol. 34 (1986), pp. 111-124.

[MCD94] Maniezzo, V., Colorni, A. e Dorigo, M., *The Ant System Applied to the quadratic assignment problem*, Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, (1994).

[MR94] Mautor, T. e Roucairol, C., *A new exact algorithm for the solution of quadratic assignment problems*, Discrete Applied Mathematics, vol. 55 (1994), pp. 281-293.

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

[Mc70] McCormik, E.J., *Human Factors Engineering*, McGraw-Hill, New York, (1970).

[Mo98] Moreira, A.S.T., *Aplicação de uma Técnica Branch and Bound a um Modelo Reduzido para p Problema do Caixeirto Viajante*, Tese de M.Sc., Programa de Engenharia de Produção - COPPE/UFRJ, Brasil, (1998).

[MRRTT53] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. e Teller, E., *Equation of state calculations by fast computing machines*, Journal of Chemical Physics, vol. 21 (1953), pp. 1087-1092.

[Mic92] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag (1992).

[PC89] Pardalos, P.M., Crouse, J., *A parallel algorithm for the quadratic assignment problem*, Proceedings Supercomputing Conference (1989), pp. 351-360.

[PPR95] Pardalos, P.M., Pitsoulis, L.S. e Resende, M.G.C., *A parallel GRASP implementation for the quadratic assignment problem*, Parallel Agorithms for Irregular Problems: State of the Art, Kluwer Academic Publishers, pp.111-128, (1995).

[PRW94] Pardalos, P.M., Rendl, F. e Wolkowicz, H., *The quadratic assignment problem: a survey and recent developments*, DIMACS Series Discr. Math. Theor. Comp. Sci., vol. 16 (1994), pp. 1-42.

[Que94] Querido, T.M., *Simulated annealing no grafo das inversões do problema quadrático de alocação*, Tese de D.Sc., Programa de Engenharia de Produção - COPPE/UFRJ, Brasil, (1994).

[RABB98] Rangel, M.C., Abreu, N.M.M., Boaventura-Netto, P.O. e Boeres, M.C.S., *Grasp para o PQA: uma busca local modificada*, Anais do V Encontro Regional de Matemática Aplicada e Computacional (V ERMAC), (1998), pp. 43-48.

[RAB00] Rangel, M.C., Abreu, N.M.M. e Boaventura-Netto, P.O., *GRASP para o PQA: um limite de aceitação para soluções iniciais*, Pesquisa Operacional, vol. 20 (2000), pp. 45-58.

[RABB00] Rangel, M.C., Abreu, N.M.M., Boaventura-Netto, P.O. e Boeres, M.C.S., *Algoritmo Guloso Adaptativo e Aleatório para o Problema Quadrático de Alocação*, Revista Produção, vol. 9 (2000).

[Rou87] Roucairol, C., *A parallel branch and bound algorithm for the quadratic assignment problem*, Discrete Applied Mathematics, vol. 18, pp. 211-225, (1987).

[SG76] Sahni, S. e Gonzalez, T., *P-complete approximation problems*, Journal of the Association of Computing Machinery, vol. 23 (1976), pp. 555-565.

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

[Tai91] Taillard, E.D., *Robust tabu search for the quadratic assignment problem*, Parallel Computing, vol. 17 (1991), pp. 443-455.

[TS94] Tate, D.E. e Smith, A.E., *A genetic approach to the quadratic assignment problem*, Computers and Operations Research, vol. 22, (1995), pp.73-83.

[VRA95] Vernet, O., Rodrigues, R.M.N.D. e Abreu, N.M.M., *Reticulados de Permutações*, Relatório Técnico, EP-03/95 Série P.O., Programa de Engenharia de Produção - COPPE/UFRJ, Brasil, (1995).

[WW87] Wilhelm, M.R. e Ward, T.L., *Solving quadratic assignment problem by simulated annealing*, IEEE Transaction, vol. 19 (1987), pp. 107-119.