Análise do comportamento da resolução do Problema Quadrático de Alocação através de instâncias isomorfas

Ricardo de Magalhães Simões

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Aprovado por:

Profª Drª Maria Cristina Rangel

Profª Drª Nair Maria Maia de Abreu

Profª. Drª Maria Claudia Silva Boeres

VITÓRIA, ES, BRASIL
NOVEMBRO DE 2006

SIMÕES, RICARDO DE MAGALHÃES

Análise do comportamento da resolução do Problema Quadrático de Alocação através de instâncias isomorfas [Vitória] 2006

xii, 82p., 29,7cm (PPGI/UFES, M.Sc., Informática, 2006)

Dissertação, Universidade Federal do Espírito Santo, PPGI

1 – Problema Quadrático de Alocação

2 – Otimização Combinatória

3 – Isomorfismo

4 – Ordenação

Agradecimentos

A Deus, por todas as bênçãos recebidas durante o curso, e principalmente pela proteção e inspiração para a conclusão.

A minha família pelo apoio que me deram para a conclusão dos meus estudos.

A professora Dra. Maria Cristina Rangel, por toda atenção e paciência que teve comigo nesse trabalho. Pela ajuda prestada na elaboração desta dissertação.

Resumo da dissertação submetida ao PPGI/UFES como requisito parcial para obtenção do grau de Mestre em Ciência da Computação

Análise do comportamento da resolução do Problema Quadrático de Alocação através de instâncias isomorfas

Ricardo de Magalhães Simões

Novembro de 2006

Orientadora: Maria Cristina Rangel

Programa: Informática

O Problema Quadrático de Alocação, PQA, foi definido originalmente por Tjalling C. Koopmans and Martin Beckman em 1957, como sendo um problema matemático relacionado a atividades econômicas. Vários pesquisadores tem utilizado o modelo PQA para formular diversos tipos de problemas. O PQA é considerado um dos problemas mais difíceis para ser resolvido computacionalmente e essa dificuldade leva muitos pesquisadores a estudá-lo tentando descobrir novos caminhos que ajudem a sua resolução. Neste contexto, este trabalho apresenta o PQA, as classes de famílias de problemas que possuem características comuns e um algoritmo para a ordenação da instância primitiva do problema, gerando uma instância isomorfa. Esta ordenação é utilizada como estratégia para facilitar a sua resolução. As instâncias retiradas da QAPLIB e as respectivas instâncias isomorfas foram submetidas às metaheurísticas GRASP, Busca Tabu, Colônias de Formigas e um algoritmo construtivo, HeuristicHead, desenvolvido por Resendo e Rangel [82]. Os resultados desses experimentos computacionais indicam que a manipulação dos dados de entrada podem ser considerados como uma boa estratégia de resolução do problema.

Summary of the dissertatoin submitted to the PPGI/UFES as partial requisite for attainment of the degree of Master in Computer Science

Analysis of the behavior of the resolution of the Quadratic Problem of Allocation through isomorfics instances

Ricardo de Magalhães Simões

November of 2006

Advisor: Maria Cristina Rangel

Departament: Informática

The Quadratic Assignment Problem, QAP, was defined originally by Tjalling C. Koopmans and Martin Beckman in 1957, as a mathematical problem related to economic activities. Some researchers have used QAP model to formulate diverse types of problems. The QAP is considered one of the most difficult problems to be solved in a computater and this difficulty takes many researchers to study it, trying to discover new ways that help its resolution. In this context, this work presents the QAP, the relative class of families of problems that have common characteristics and an algorithm for the sort of the primitive instance of the problem, generating an isomorfic instance. This sort is used as strategy to make easy its resolution. The instances take from the QAPLIB and the respective isomorfics instances had been submitted to the metaheuristics GRASP, Tabu Search, Ant Colony and a constructive algorithm, HeuristicHead, developed by Resendo and Rangel [82]. The results of these computational experiments indicate that the manipulation of the input data can be considered as a good strategy to resolution of the problem.

Índice

Tabelas

Figuras

Capítulo 1
Introdução

Qual a melhor localização de uma determinada clínica em um hospital ? Qual o melhor desenho para uma placa de circuito de computador? Estas perguntas abordam dois temas distintos, mas a solução para ambas pode ser obtida seguindo um princípio: alocação e disponibilização de recursos. Estes problemas, e muitos outros, possuem uma característica comum, que é a associação de um recurso (sala ou circuito integrado) com o melhor local para se disponibilizá-lo para que outros possam utilizá-lo. Tais problemas são modelados utilizando-se um modelo de Otimização Combinatória. Em especial, esses problemas fazem parte de uma classe definida como Problema Quadrático de Alocação. O Problema Quadrático de Alocação, PQA, modela diversas aplicações em diferentes áreas como pesquisa operacional, projeto de construção industrial [3], computação paralela [5], linha de produção industrial [17], bio-química [47], planejamento hospitalar [20], e análise estatística de dados [4]. O PQA foi definido originalmente por Tjalling C. Koopmans e Martin Beckmann, em 1957, como sendo um problema matemático relacionado a atividades econômicas [30] que visa encontrar a associação ótima de pares de localidades a pares de atividades, quando são conhecidos os fluxos entre as atividades e as distâncias entre as localidades.

1.1 O Problema Quadrático de Alocação

O objetivo do PQA é fazer a associação entre dois conjuntos, através de uma função bijetora, sendo que a associação é feita entre 2 elementos de um conjunto com 2 elementos do outro conjunto, de forma tal que se otimize o fator dessa associação. Esta associação é feita 2 a 2, porque em cada um dos conjuntos, os elementos estão relacionados entre si através das características dos respectivos conjuntos. Exemplo: suponha que se deseja fazer a alocação física de funcionários em uma determinada empresa. O PQA nesse caso poderá ser formulado da seguinte forma: seja F o conjunto de funcionários e f_{ij} um valor associado a afinidade do funcionário *i* ao funcionário *j* (a afinidade é a dependência das tarefas entre os funcionários). Seja D o conjunto de escritórios da empresa, e d_{kl} a distância entre o escritório *k* e o escritório *l*. Assuma que a pessoa *i* está alocada no escritório *k*, e a pessoa *j* está alocada no escritório *l*. Assim, o custo para essa associação pode ser expresso da seguinte forma: $f_{ij}d_{kl}$. O objetivo final da empresa é ter uma alocação de empregados tal que funcionários com maior afinidade estejam mais próximos uns dos outros.

Um problema que pode ser utilizado para exemplificar a aplicação do PQA é o Problema do Fluxo de Pessoas em um Hospital, onde o objetivo é encontrar uma melhor distribuição das salas que diminua o fluxo entre as mesmas, melhorando assim as condições de trabalho no hospital. A Figura 1.1 mostra uma parte de um Hospital, onde foram destacadas algumas salas e departamentos.

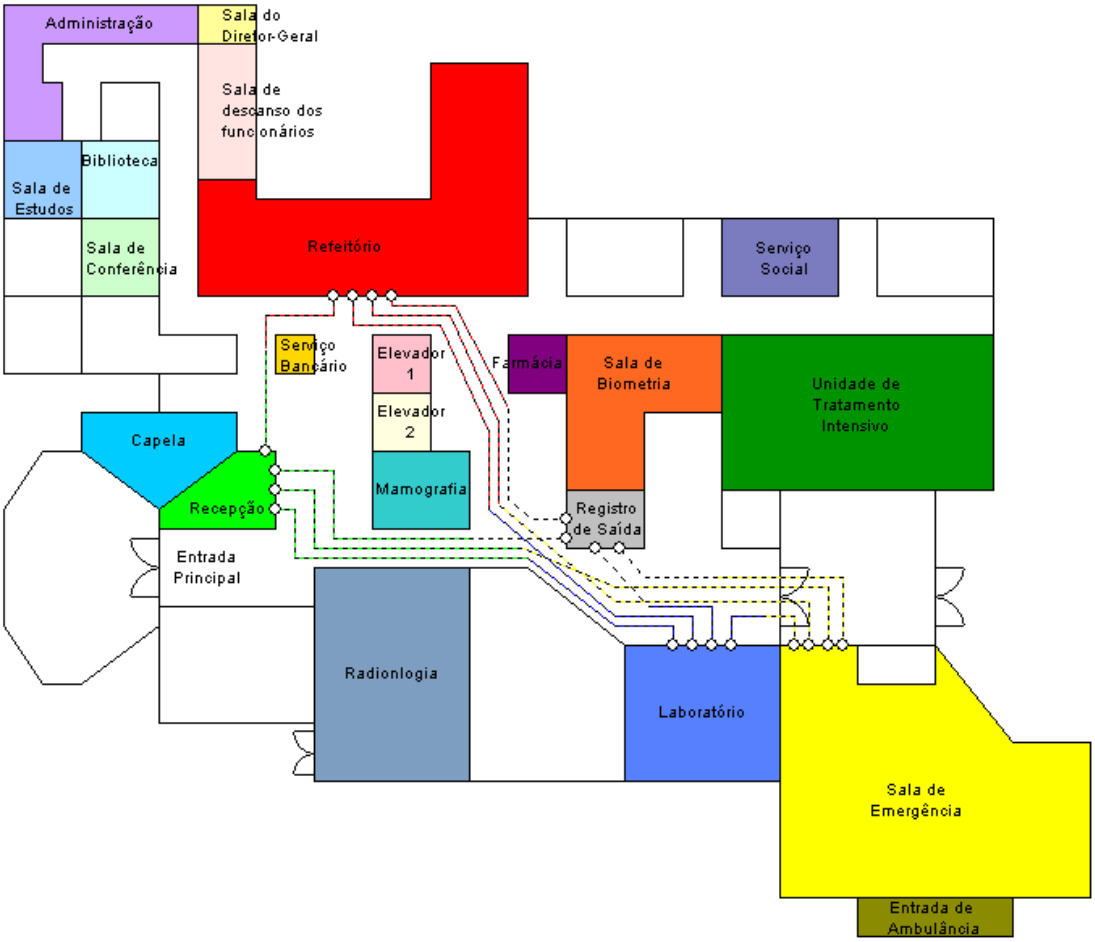


Figura 1.1: Planta parcial de um hospital

Neste caso, será analisado o fluxo de pessoas entre as seguintes salas:

- Recepção
- Refeitório
- Laboratório
- Emergência
- Registro de Saída de Paciente

Esta planta deve ser transformada em um grafo completo pois a partir de qualquer sala pode-se chegar a uma outra sala do Hospital. Este grafo está representado na Figura 1.2 (para simplificar o desenho, apenas algumas arestas foram apresentadas) :

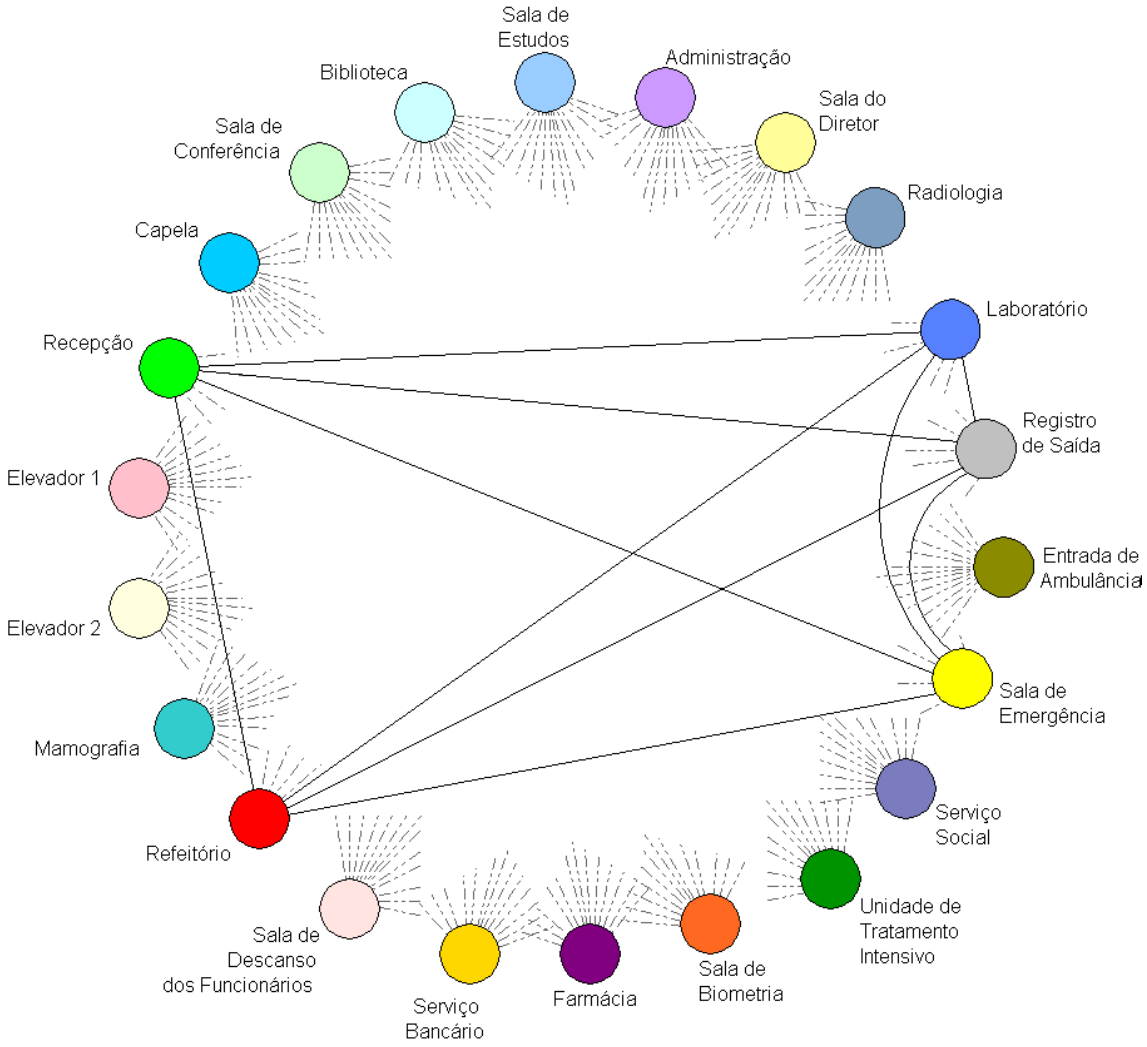
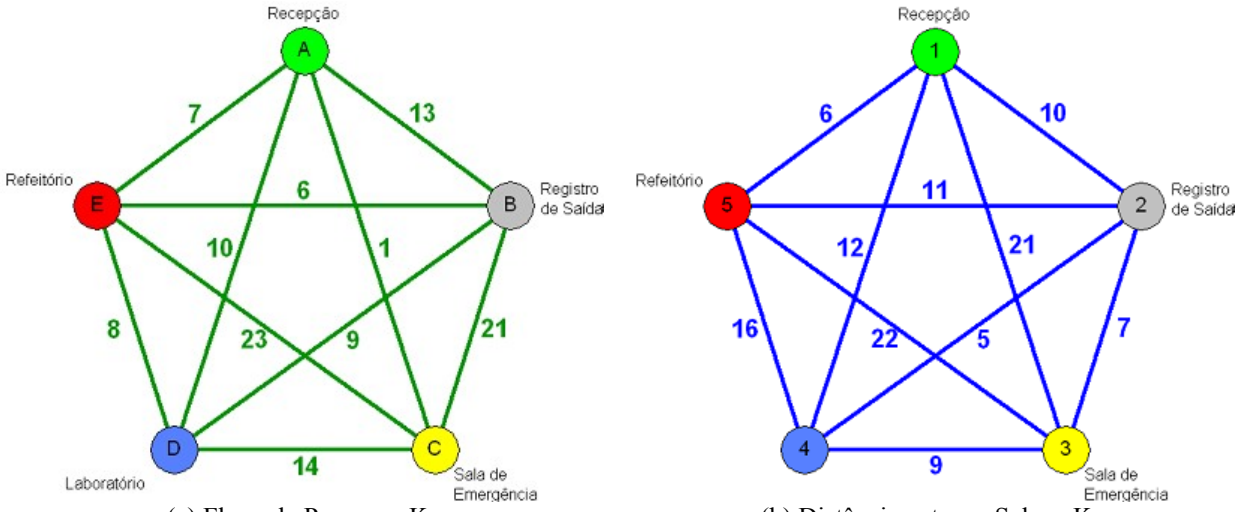


Figura 1.2: Grafo que representa a planta de um hospital

A partir do grafo apresentado na Figura 1.2, serão formadas duas cliques K_F e K_D, que serão utilizadas como base para a construção de dois grafos, um representando o fluxo de pessoas entre as salas, e outro a distância entre as salas, como pode ser visto na Figura 1.3:



(a) Fluxo de Pessoas - K_F (b) Distância entre as Salas - K_D

Figura 1.3: Grafo de Fluxo de Pessoas (a) e Distância entre as Salas (b)

A sobreposição dos vértices das cliques K_F e K_D, dada pela permutação:

$$\pi = \begin{pmatrix} A & B & C & D & E \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \tag{1.1}$$

é a configuração atual de parte do Hospital apresentado na Figura 1.1, e o valor 1331, é o valor atual da relação entre o fluxo de pessoas e as distâncias entre as salas. O custo desta configuração é dado pelo somatório de $f_{ij}d_{\pi(i)\pi(j)}\ \forall i,j = \{A,B,C,D,E\}$:

$$C(\pi) = f_{AB}d_{12} + f_{AC}d_{13} + f_{AD}d_{14} + f_{AE}d_{15} + f_{BC}d_{23} + \ldots + f_{DE}d_{45} = \Sigma f_{ij}d_{\pi(i)\pi(j)} \tag{1.2}$$

$$C(\pi) = 13\text{x}10 + 1\text{x}21 + 10\text{x}12 + 7\text{x}6 + 21\text{x}7 + 9\text{x}5 + 6\text{x}11 + 14\text{x}9 + 23\text{x}22 + 8\text{x}16 \tag{1.3}$$

$$C(\pi) = 1331 \tag{1.4}$$

1.2 Uma Formulação do PQA

Proposto originalmente por Koopmans e Beckmann (1957) como um modelo matemático relacionado a atividades econômicas [30], sendo modelado como um problema de alocação industrial: *n* indústrias serão designadas à *n* localidades, sempre 2 indústrias à 2 localidades, e a relação entre as indústrias pode ser mapeado de acordo com a relação entre as etapas de produção, e a relação entre as localidades de acordo com a distância entre as mesmas. Define-se as matrizes A e

B, onde A=(a_{ij}), sendo a_{ij} o fluxo de material entre a indústria *i* e a indústria *j*, por unidade de tempo, e a matriz B=(b_{kl}), sendo b_{kl} a distância entre a localização *k* e a localização *l,* e a alocação de uma indústria *i* em uma localidade *k* é definido como $\pi(i) = k$ onde π é uma permutação de 1 à n. O custo da alocação simultânea das indústrias *i* e *j* nas localidades *k* e *l* é: $a_{ij} b_{kl} = a_{ij} b_{\pi(i)\pi(j)}$. O custo final do problema será o somatório de todos os custos de pares de indústrias e pares de localidades, sendo π a permutação que define as alocações. Neste trabalho o PQA considera duas matrizes nxn simétricas e com diagonal principal nula. Em alguns casos, as matrizes podem ser assimétricas, como as apresentadas por Burkard et al [7].

A formulação matemática do PQA é feita da seguinte forma: Sejam *A* e *B* duas matrizes nxn tais que A = [a_{ij}] e B = [b_{kl}]. Considere ainda o conjunto { 1, 2, ... n } e seja S_n o conjunto das permutações deste conjunto. Assim o PQA pode ser definido como [16,18]:

$$C(\pi^*) = \min_{\pi \in S_n} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \right) \tag{1.1}$$

A formulação acima é conhecida como *Formulação de Koopmans-Beckmann*.

1.3 Técnicas de resolução do PQA

Para se obter um resultado ótimo para o PQA, deve-se verificar todas as combinações possíveis entre as duas matrizes, e no final verificar qual associação possui a melhor combinação entre os elementos do problema. Isto demandará um esforço computacional considerável, pois serão realizadas n! combinações. Desde a sua primeira formulação, o PQA tem atraído a atenção de pesquisadores em todo o mundo, não apenas pela sua importância prática e teórica, mas principalmente pela sua complexidade. Trata-se de um dos mais difíceis problemas de otimização combinatória, sendo que instâncias de ordem $n > 30$ em geral, não são resolvidas exatamente em tempo computacional aceitável.

Em 2005, Peter Hahn anunciou a resolução exata da instância Tai30b através de um algoritmo de *branch-and-bound* SLC_v3, desenvolvido por Aziz Djerrah e Van-Dat Cung. A enumeração completa consumiu 102 dias de processamento em um computador Dell 7150 (Pentium 3 733MHz), e nesta enumeração foi utilizado o procedimento RLT-2 criado por Hahn, Hightower e Johnson para o cálculo do limite inferior [21]. Este método se mostrou bastante eficiente ao ser

aplicado também a instância Nug28, em que realizou a enumeração completa em aproximadamente 52 dias, sendo que a enumeração feita por um outro algoritmo necessitou de cerca de 100 dias.

Shani e Gonzales demonstraram que qualquer algoritmo que encontre um resultado aproximado para o PQA é um algoritmo NP-Completo, sendo o PQA "o mais difícil dos difíceis" [48]. Existem muitos de problemas que são classificados como pertencentes à classe NP. Problemas relacionados a matemática, otimização, inteligência artificial, biologia, física, economia, indústria entre outros, que se originam em diferentes áreas, e que, com um método de resolução eficiente irão trazer grandes benefícios à toda sociedade. A classe NP é formada pelos algoritmos que não podem ser descritos em um tempo polinomial [53].

Devido a essa característica, em determinados casos a solução adotada para se resolver um PQA é tentar encontrar uma resposta que se aproxime do valor ótimo do problema. E uma das estratégias utilizadas para melhorar o desempenho do processamento na etapa de busca é minimizar a quantidade de operações feitas pelo computador, e isto é feito utilizando-se um valor limite para o problema, que é um valor mínimo para que uma determinada solução possa ser considerada como aceitável, isto é, qualquer solução que ultrapasse esse valor é descartada, e uma nova solução é procurada. Um dos métodos mais utilizados para o cálculo de um limite para o PQA foi desenvolvido pelos pesquisadores Gilmore e Lawler. Também são utilizadas heurísticas e metaheurísticas na resolução dos problemas. Uma heurística é um algoritmo feito visando os seguintes objetivos: rápida velocidade de execução, e boa solução encontrada. Metaheurísticas são heurísticas utilizadas para a resolução de uma classe mais geral de problemas, onde não se conhece um algoritmo satisfatório para resolver os problemas da classe. Não se pode dizer que a solução encontrada seja a solução ótima, apenas em casos em que já se conheça o valor ótimo (encontrado por algum algoritmo exato).

Detalhes sobre o Problema Quadrático de Alocação e suas variações podem ser encontrados em [9] ,[10], [35].

1.4 Objetivos

O objetivo deste trabalho é realizar comparações nos métodos de resolução do Problema Quadrático de Alocação utilizando instâncias isomorfas. É definido no Capítulo 2 o Problema Quadrático de Alocação segundo os aspectos de formulação matemática, onde serão apresentadas

algumas das formulações utilizadas. No Capítulo 3, são apresentadas algumas técnicas de resolução do PQA, a utilização de limites para agilizar a busca da solução, heurísticas e metaheurísticas empregadas na resolução do problema. No Capítulo 4, são definidas as características das instâncias do PQA, as famílias de instâncias e os problemas isomorfos, e o conceito de número de inversões associada a uma solução do PQA. O Capítulo 5 irá abordar a ordenação da instância como estratégia para obtenção de uma instância mais fácil de se resolver e que seja isomorfa à instância original, e também são apresentados os resultados obtidos nos testes realizados com ambas as instâncias. Finalmente, no Capítulo 6, são apresentadas as conclusões deste trabalho, e propostas de desenvolvimento de novos estudos para este tema.

Capítulo 2

Formulação do PQA

Neste capítulo serão apresentadas algumas formulações matemáticas utilizadas para representar o Problema Quadrático de Alocação:

- Programação Inteira;
- Programação Inteira Mista;
- Permutação;
- Traço;
- Programação Semi-Definida;
- Grafos.

Outras formulações, além das apresentadas neste trabalho, serão encontradas em [38].

2.1 Programação Inteira

O PQA será apresentado como um problema de alocação indústrial em determinadas localidades. Sejam x_{ij} variáveis binárias representando a associação da indústria *i* à localidade *j,* e seja o custo da associação da indústria *i* à localização *j* simultaneamente a associação a indústria *k* a localização *l* denotado por c_{ijkl} [37], desta forma o PQA pode ser escrito da seguinte maneira:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} c_{ijkl}\, x_{ij}\, x_{kl} \tag{2.1}$$

s.a.

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \leqslant j \leqslant n; \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \leqslant i \leqslant n; \tag{2.3}$$

$$\sum_{k=1}^{n} x_{kl} = 1 \qquad 1 \leqslant l \leqslant n; \tag{2.4}$$

$$\sum_{l=1}^{n} x_{kl} = 1 \qquad 1 \leqslant k \leqslant n; \tag{2.5}$$

$$x_{ij}, x_{kl} \in \{0,1\} \qquad 1 \leqslant i, j, k, l \leqslant n; \tag{2.6}$$

Neste contexto, quando a indústria *i* estiver na localidade *j*, x_{ij} será igual a 1, da mesma forma, quando a indústria *k* estiver na localidade *l*, x_{kl} será igual a 1. Então, quando o par de

indústrias i,k estiver no par de localização j,l, o valor da expressão $c_{ijkl}x_{ij}x_{kl}$ será igual a c_{ijkl}, caso contrário, será igual a 0 (zero). Maiores informações podem ser encontradas em [36].

2.2 Programação Inteira Mista

Na literatura é encontrado via diversas propostas. Em todas elas, o termo quadrático da função objetivo do problema é substituído por termos lineares: $x_{ik}\,x_{jl} = y_{ijkl}$

O PQA pode ser formulado da seguinte maneira [42]:

$$min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}\,d_{kl}\,y_{ijkl} \tag{2.7}$$

s.a.

$$\sum_{i=1}^{n} x_{ik} = 1 \qquad 1 \leqslant k \leqslant n; \tag{2.8}$$

$$\sum_{k=1}^{n} x_{ik} = 1 \qquad 1 \leqslant i \leqslant n; \tag{2.9}$$

$$\sum_{i=1}^{n} y_{ijkl} = x_{jl} \qquad 1 \leqslant j, k, l \leqslant n; \tag{2.10}$$

$$\sum_{j=1}^{n} y_{ijkl} = x_{ik} \qquad 1 \leqslant i, k, l \leqslant n; \tag{2.11}$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{jl} \qquad 1 \leqslant i, j, l \leqslant n; \tag{2.12}$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ik} \qquad 1 \leqslant i, j, k \leqslant n; \tag{2.13}$$

$$y_{iijj} = x_{ij} \qquad 1 \leqslant i, j \leqslant n; \tag{2.14}$$

$$x_{ik}, x_{jl} \in \{0,1\} \qquad 1 \leqslant i, j, k, l \leqslant n; \tag{2.15}$$

$$0 \leqslant y_{ijkl} \leqslant 1 \qquad 1 \leqslant i, j, k, l \leqslant n; \tag{2.16}$$

A formulação apresentada pelos pesquisadores Alan M. Frieze e J. Yadegar em 1983 [33] utiliza n^4 variáveis reais, n^2 variáveis lógicas, $n^4 + 4n^3 + n^2 + 2n$ restrições para representar o PQA. Maiores informações sobre Programação Inteira Mista podem ser encontradas em [29].

2.3 Permutação

Na Formulação por Permutação, considera-se um conjunto S_n como sendo o conjunto de todas as permutações a *n* elementos, f_{ij} o fluxo entre os objetos *i* e *j*, e $d_{\pi(i)\pi(j)}$ as distâncias entre as localidades $\pi(i)$ e $\pi(j)$, dada pela permutação $\pi \in S_n$. Se para cada permutação π houver uma alocação de um objeto a uma localidade, o PQA pode ser descrito pela expressão:

$$\min_{\pi \in S_n} \sum_{i,j=1}^{n} f_{ij} d_{\pi(i)\pi(j)} \tag{2.16}$$

Nesta formulação, o custo da associação de um par de objetos a um par de localidades é proporcional ao produto entre o fluxo e a distância entre eles [33], sendo equivalente à formulação por programação inteira, pois as restrições (2.2) à (2.5) definem matrizes de permutação $X=[x_{ij}]$, onde:

$$x_{ij} = \begin{cases} 1, se\ \pi(i) = j; \\ 0, se\ \pi(i) \neq j; \end{cases} \tag{2.17}$$

2.4 Traço

Uma outra formulação do PQA pode ser obtida a partir da utilização de matrizes de permutação. Para uma determinada instância PQA(A,B) de tamanho n, uma função f(A,B) pode ser definida como o conjunto Π_n das matrizes de permutação [24]:

$$f_{(A,B)}: \Pi_n \rightarrow \mathbb{R} \tag{2.18}$$

$$X \rightarrow tr(A X B X^t) \tag{2.19}$$

Onde X^t denota a transposta da matriz correspondente, e tr(A) é o traço da matriz A, isto é, $tr(A) = \sum_{i=1}^{n} a_{ii}$, para uma matriz $A_{n \times n}$.

2.5 Programação Semidefinida

Programação Semidefinida é uma extensão da Programação Linear onde as restrições de não-negatividade são substituídas por restrições representadas por matrizes semidefinidas. Embora a programação semidefinida tenha sido estudada no passado como sendo uma parte do problema de Programação Cônica, existe atualmente um interesse renovado graças aos sucessos obtidos em aplicações em otimização discreta [54].

A programação semidefinida é aplicada ao PQA através da relaxação Lagrangeana, transformando o problema original em um problema contendo matrizes simétricas de dimensões muito grandes. Também são adicionadas restrições descritas por matrizes permutação X, que podem também ser representadas como sendo vetores binários. Desta forma o PQA pode ser escrito como:

$$\min_{X \in \Pi_n} f_{A,B}(X) = \min_{X \in \Pi_n} tr(A X B X^t - 2 C X^t) \tag{2.20}$$

s.a.

$$X X^t = X^t X = I \tag{2.21}$$

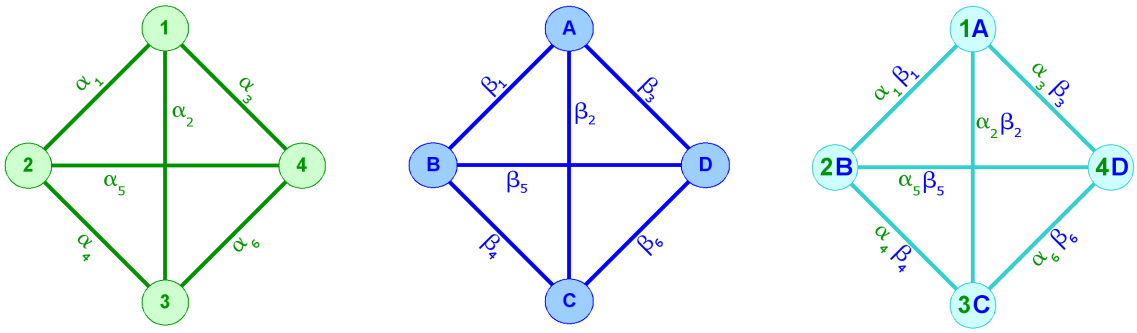
$$X e = X^t e = e \tag{2.22}$$

$$X_{ij}^2 - X_{ij} = 0, \qquad \forall i, j \tag{2.23}$$

Esta formulação utiliza o vetor *e* (um vetor de 1's), que é utilizado para a construção da matriz semidefinida.

2.6 Grafos

A formulação por grafos foi utilizada no exemplo da construção do Problema do Hospital no capítulo anterior deste trabalho. São construídos dois grafos, um com as arestas representando o fluxo e outro com as arestas representando as distâncias entre as salas, como pode ser visto na Figura 2.1. O valor obtido para o PQA será a soma dos produtos das arestas, resultante de uma sobreposição dos nós representados por uma permutação $\pi \in S_n$ (conjunto das permutações de {1,2,..,n}).



(a) (b) (c)

Figura 2.1: Formulação do PQA utilizando Grafos
(a) Fluxo entre as Salas (b) Distância entre as Salas (c) Sobreposição de a em b

Esta formulação pode ser expressa na seguinte pela expressão:

$$C(\pi^*) = \min_{\pi \in S_n} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)} \right) \tag{2.24}$$

que coincide com a expressão (2.16) da formulação por permutação.

2.6.1 Relaxação do PQA

Nesta seção uma relaxação do PQA será apresentada através do Problema de Alocação Linear, PAL. Utilizando o exemplo do Hospital, no capítulo anterior, serão montadas duas matrizes, uma para cada grafo da Figura 1.3, sendo que estas matrizes são a representação matricial do grafo em questão, como pode ser visto na Figura 2.2. A matriz é formada pelos dados das arestas dos grafos, sendo que a primeira linha contém as informações das arestas que saem do nó A, a segunda linha contém as informações das arestas que saem do nó B, e assim sucessivamente. Desta forma, a matriz formada será naturalmente simétrica. (As arestas que saem do último nó já estão representadas pelas arestas dos outros nós).

F =

	A	B	C	D	E
A	0	13	1	10	7
B	13	0	21	9	6
C	1	21	0	14	23
D	10	9	14	0	8
E	7	6	23	8	0

(a)

D =

	1	2	3	4	5
1	0	10	21	12	6
2	10	0	7	5	11
3	21	7	0	9	22
4	12	5	9	0	16
5	6	11	22	16	0

(b)

Figura 2.2: Representaçao das (a) Matriz de Fluxo e (b) Matriz de Distância.

Os elementos das matrizes F = (f_{ij}) e D = (d_{kl}) são armazenados em vetores linhas, de ordem N = $C_{n,2}$, $\vec{F}$ = (f_z) e $\vec{D}$ = (d_z), e as coordenadas dos elementos nestes vetores obedecerá a ordem lexicográfica das arestas em K_F e K_D, segundo a seguinte bijeção ψ (i,j) = (i-j)n – i(i+1)/2 + j, representados na Figura 2.3:

(a) *(b)*

Figura 2.3: Vetores F e D

Os vetores $\vec{F}$ e $\vec{D}$ serão utilizados para a criação da Matriz Q, que representa a sobreposição de todas as arestas dos grafos da Figura 2.4:

	13	1	10	7	21	9	6	14	23	8
10	130	10	100	70	210	90	60	140	230	80
21	273	21	210	147	441	189	126	294	483	168
12	156	12	120	84	252	108	72	168	276	96
6	78	6	60	42	126	54	36	84	138	48
7	91	7	70	49	147	63	42	98	161	56
5	65	5	50	35	105	45	30	70	115	40
11	143	11	110	77	231	99	66	154	253	88
9	117	9	90	63	189	81	54	126	207	72
22	286	22	220	154	462	198	132	308	506	176
16	208	16	160	112	336	144	96	224	368	128

Figura 2.4: Matriz Q

E, consequentemente, possui todos os valores de multiplicação entre os elementos do vetor de Fluxos e os elementos do vetor de Distâncias. É a partir desta multiplicação que se obterá o valor função objetivo. Para calcular o valor da função objetivo, deve-se selecionar um elemento em cada linha e coluna, sendo que apenas um elemento pode ser escolhido em uma linha e uma coluna.

Se for considerada apenas a sobreposição de arestas, o PQA pode ser representado por uma relaxação na forma de um Problema de Alocação Linear PAL(Q), que faz apenas sobreposição de arestas:

$$Z(\xi^*) = \min_{\xi \in \Pi_N} \sum_{i=1}^{N} f_i d_{\xi(i)} \qquad (1.5)$$

onde Π_N é conjunto de todas as permutações ξ de {1, ... , N}. O conjunto de soluções do PAL é maior que o conjunto de soluções do PQA, e desta forma, dada uma solução ξ do PAL(Q), nem sempre existe uma sobreposição de vértices compatível com a sobreposição das arestas, e neste caso, diz-se que a solução do PAL(Q) é uma solução não viável no PQA(F,D) [43]. Se a sobreposição π existir, a seguinte equação deve ser obedecida:

$$f_p d_{\xi(p)} = f_{\Psi^{-1}(p)} d_{\Psi^{-1}(\xi(p))} = f_{ij} d_{kl}; \qquad p = 1,2,\ldots,N \tag{1.6}$$

Desta forma, uma solução ξ no PAL(Q) que possua uma correspondente solução π no PQA(F,D) é chamada de solução viável [42].

Uma ordenação dos vetores de Fluxo e Distância, feita de tal forma que, o vetor de Fluxo seja não-crescente ($\vec{F}^-$) e o vetor de Distância não-decrescente ($\vec{D}^+$), resultará em uma outra matriz, sendo esta nova matriz chamada Matriz Q*:

	23	21	14	13	10	9	8	7	6	1
5	115	105	70	65	50	45	40	35	30	5
6	138	126	84	78	60	54	48	42	36	6
7	161	147	98	91	70	63	56	49	42	7
9	207	189	126	117	90	81	72	63	54	9
10	230	210	140	130	100	90	80	70	60	10
11	253	231	154	143	110	99	88	77	66	11
12	276	252	168	156	120	108	96	84	72	12
16	368	336	224	208	160	144	128	112	96	16
21	483	441	294	273	210	189	168	147	126	21
22	506	462	308	286	220	198	176	154	132	22

*Figura 2.5: Matriz Q**

Encontrar a melhor solução para o PAL(Q) passa a ser uma tarefa simples, bastando apenas realizar o cálculo do traço da Q*, pois o PAL(Q) e o PAL(Q*) possuem o mesmo conjunto de soluções [43]. Neste caso, o valor deste limite é:

$$Z(\xi^*) = tr(Q^*) = 23\text{x}5 + 21\text{x}6 + 14\text{x}7 + 13\text{x}9 + 10\text{x}10 + 9\text{x}11 + 8\text{x}12 + 7\text{x}16 + 6\text{x}21 + 1\text{x}22 \tag{1.7}$$

$$Z(\xi^*) = 1011 \tag{1.8}$$

Para encontrar uma solução no PAL(Q) que corresponda a uma solução no PAL(Q*) deve-se realizar o seguinte procedimento: assumindo que ϕ_F e ϕ_D sejam permutações auxiliares que transformam os vetores $\vec{F}$ e $\vec{D}$ em $\vec{F^-}$ e $\vec{D^+}$, e que $\rho \in \Pi_N$, faz-se: $\xi = \Phi_D^{-1} o \rho o \Phi_F$

Para saber se uma solução no PAL(Q*) é viável no PQA(F,D), a seguinte igualdade deve ser satisfeita: $f_r^- d_{\rho(r)}^+ = f_{\phi_F^{-1}(r)}^- d_{\phi_D^{-1}(\rho(r))}^+ = f_p d_{\xi(p)} = f_{\Psi^{-1}(p)} d_{\Psi^{-1}(\xi(p))} = f_{ij} d_{kl}$

A Matriz Q* possui duas características especiais para o PQA: a soma dos elementos da diagonal principal resultará no limite inferior para as soluções, e a soma dos elementos da diagonal secundária no limite superior para as soluções.

Neste exemplo, a soma dos elementos da diagonal secundária resulta no valor 1689. Os elementos que formam a diagonal principal na Matriz Q* estão presentes na Matriz Q, a diferença é que a posição em que esses elementos estão na Matriz Q é diferente da posição que eles estão na Matriz Q*.

Capítulo 3

Técnicas de Resolução do PQA

O PQA pertence a uma classe de problemas de otimização combinatorial, e a resolução de problemas desta categoria, geralmente, não pode ser feita através da enumeração das soluções, pois problemas de ordem n=15 abrangem um universo de 15! soluções a serem analisadas. Para que o PQA possa ser resolvido, foram desenvolvidas algumas estratégias que auxiliam na busca da solução final do problema, e estas estratégias são a utilização de limites inferiores para a solução e heurísticas para o processamento do PQA.

3.1 Utilização de Limites Inferiores

O estudo de limites inferiores é importante no desenvolvimento de algoritmos de resolução de problemas de programação matemática e de otimização combinatória. Em geral, os métodos exatos trabalham com enumeração implícita, procurando-se garantir o ótimo e ao mesmo tempo evitando-se a enumeração completa das soluções viáveis dos problemas. O desempenho de tais métodos depende diretamente da qualidade e da eficiência computacional dos limites inferiores envolvidos. Desta forma, os limites inferiores tornam-se instrumentos fundamentais para as técnicas *branch-and-bound* e também para testar a qualidade das soluções produzidas por algoritmos heurísticos.

3.1.1 Limite de Gilmore e Lawler (GLB)

Este foi um dos primeiros limites para o PQA proposto pelos pesquisadores Gilmore e Lawler, e se baseia na resolução de um problema de alocação linear [32], onde para um dado problema envolvendo duas matrizes A e E, o GLB(A,E), isto é, o limite Gilmore-Lawler, é definido da seguinte forma: Sejam a_i, e_i, com $i=1,2,..,n$, representações das linhas das matrizes A e E respectivamente. Seja $\hat{a}_i$ o vetor das (n-1) componentes de a_i, excluindo-se a_{ii}, e seja $\hat{e}_i$ o vetor das (n-1) componentes de e_i, excluindo-se e_{ii}, defini-se a matriz $L = [l_{ij}]$ como: $l_{ij} = a_{ii}e_{jj} + \langle \hat{a}_i, \hat{e}_j \rangle^-$, i,j=1,2,...,n. Assim, o GLB(A,E), é definido como sendo a solução do problema:

$$GLB(A,E) = \min_{\pi \in \Pi} \left\{ \sum_{i=1}^{n} (l_{i\pi(i)}) \right\} \tag{3.2}$$

Uma desvantagem do GLB é o fato de a solução encontrada se afastar do valor ótimo de acordo com o tamanho do problema, por exemplo, para o caso de teste conhecido como Nugent-6 (proposto por C.E. Nugent) a solução encontrada pelo GLB é inferior à solução ótima em cerca de 5%, e para o o problema Nugent-30 o valor encontrado é inferior a solução ótima em 25%. Nesse caso, esforços estão sendo realizados em tentativas de se melhorar o cálculo do limite inferior, para um valor mais próximo do ótimo.

3.1.2 Limite Baseados em Autovalores

A criação de limites para um PQA baseados nos autovalores das matrizes envolvidas nos problemas tem sido propostos por vários pesquisadores, destacando-se Hadley [23]. Esses limites são baseados no seguinte teorema: Sejam A e B matrizes simétricas, $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_n$ são os autovalores de A e $\mu_1 \leqslant \mu_2 \leqslant ... \leqslant \mu_n$ são os autovalores de B. Para qualquer $\pi \in \Pi$, tem-se:

$$\sum_{i=1}^{n} \lambda_i \mu_{n-i+1} \leqslant \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)} \leqslant \sum_{i=1}^{n} \lambda_i \mu_i \tag{3.3}$$

Assim, os limites baseados em autovalores podem ser obtidos a partir de:

$$EVB(A, B) = \sum_{i=1}^{n} \lambda_i \mu_{n-i\mp 1} \tag{3.4}$$

Maiores informações sobre limites baseados em autovalores serão encontrados em [25].

3.1.3 Limite Baseados em Reformulações do PQA

Vários pesquisadores se basearam nas idéias de Gilmore e Lawler para aperfeiçoar o GLB. Estes novos limites definem uma sequência finita de problemas equivalentes ou reformulações do problema original, em geral relaxações lineares, e calculam o GLB para cada reformulação [33]. A sequência de problemas $P_0 = P_1, P_2, \ldots, P_k$ produz outra, de soluções que correspondem a relaxações lineares, fornecendo uma sequência de limites inferiores não decrescentes para o PQA, tais limites demandam um tempo computacional menor, mas não existe prova teórica de sua convergência. Ainda existem os limites baseados em formulações duais, onde se utiliza o problema dual de uma relaxação do problema original, em conjunto com o GLB. Os pesquisadores Hahn e

Grant [22] obtiveram resultados mostrando que estes limites tem boa qualidade em comparação aos outros limites, com a vantagem de terem um tempo computacional menor.

3.1.4 Limites Baseados em Reformulações do GLB

Christofides e Gerrard [8] propuseram em 1981 um outro método de cálculo de limite inferior para o PQA, onde o procedimento utilizado foi feito baseado em uma interpretação dos grafos do problema. Considere o PQA(A,B) de tamanho n, sendo A e B duas matrizes de dois grafos completos G_1 e G_2 de n vértices respectivamente. Cada permutação $i \in S_n$ é um isomorfismo entre G_1 e G_2. $Z(A,B,\pi)$ será o custo deste isomorfismo, e assim, a resolução do PQA(A,B) passa a ser a busca por um isomorfismo entre G_1 e G_2 que tenha o menor custo.

A idéia básica deste método consiste na decomposição dos grafos G_1 e G_2 em subgrafos isomorfos $G_1(1)$, $G_1(2)$... $G_1(k)$ e $G_2(1)$, $G_2(2)$... $G_2(k)$ respectivamente, como pode ser visto na Figura 3.1:

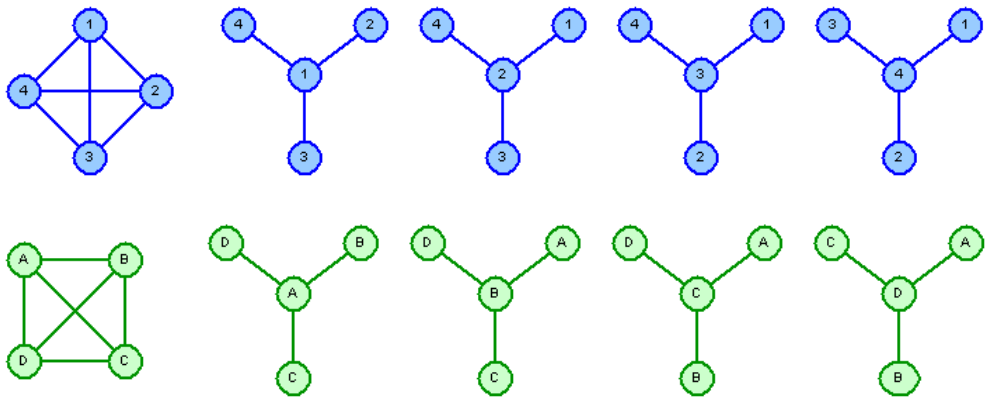


Figura 3.1: Grafos decompostos

Os subgrafos $G_1^{(i)}$ ($G_2^{(i)}$) devem ter o mesmo conjunto de vértices de G_1(G_2), e cada aresta de G_1 (G_2) deve ocorrer em pelo menos um dos subgrafos respectivamente. Além disso, o número de ocorrências de cada aresta *e* de G_1 (G_2) no subgrafo $G_1^{(i)}$($G_2^{(i)}$) é uma constante *d*, para toda ocorrência de *e*. Um isomorfismo entre G_1 e G_2 é mapeado por cada subgrafo $G_1^{(i)}$ e $G_2^{(j)}$, $1 \le i, j \le k$, e, denominando este custo, de realizar o mapeamento entre os subgrafos isomorfos, de cij, define-se como limite inferior do PQA(A,B) o valor ótimo do Problema de Alocação Linear com matriz de custo C = (c_{ij}) mutiplicado por *1/d*.

Este método é chamado de Limite CG. A complexidade dos cálculos deste limite dependerá

da estrutura dos subgrafos, pois esta estrutura irá determinar a quantidade de vértices e arestas, e assim irá influenciar diretamente a matriz C.

Maiores informações sobre limites inferiores podem ser encontrados em [40], [23].

3.2 Heurísticas

Existem três principais métodos exatos de resolução do PQA: programação dinâmica, métodos de corte, e procedimentos *branch-and-bound*. O mais utilizado é o procedimento *branch-and-bound*, entretanto, problemas de tamanho maior que n=25 são considerados de tratamento mais difícil. Para conseguir uma solução para problemas que são considerados intratáveis (n > 30) utiliza-se algoritmos heurísticos, que irão procurar a melhor solução viável para a instância considerada. Este capítulo irá abordar algumas metaheuristicas utilizadas para se encontrar uma solução viável de boa qualidade para o PQA.

3.2.1 GRASP

GRASP, *Greedy Randomized Adaptative Search Procedure*, é uma metaheurística utilizada para solucionar problemas de otimização, onde cada iteração é dividida em duas fases: a primeira fase é de construção de uma solução inicial, e a segunda fase é de melhoria desta solução [42]. O pseudo-código da metaheurística GRASP é apresentada na Figura 3.2, neste caso serão realizadas um total de repetições de acordo com MAX_Iterações, para se encontrar a melhor solução, e a variável Semente é utilizada para a criação das soluções iniciais:

```
Procedimento GRASP(MAX_Iterações, Semente)
1: Lê_Entrada;
2: Para k=1,...,MAX_Iterações Faça
3:     Solução = Construção_Solução(Semente);
4:     Solução = Busca_Local(Solução);
5:     Atualiza_Solução(Solução, Melhor_Solução);
6: Fim;
7: Retorna Melhor_Solução;
Fim GRASP.
```

Figura 3.2: Pseudo-Código da Heurística GRASP.

Na primeira fase, uma solução inicial é criada, e esta solução é passada para a segunda fase, onde será realizado um procedimento de melhoria, através de uma busca local na vizinhança da solução que iniciou a iteração. A solução do GRASP é aquela que for a melhor solução entre todas

as encontradas durante a execução do algoritmo. Esta heurística foi aplicada ao PQA em 1994 por Li, Pardalos e Resende, em um conjunto de 88 instâncias do PQA, encontrando a melhor solução conhecida em quase todos os casos, e melhorando as soluções para algumas instâncias [11].

Na Figura 3.3 é apresentado o pseudo-código para a função de construção da solução inicial, onde a solução será formada a partir de um conjunto de elementos candidatos a entrarem na solução, sem destruir a viabilidade da solução.

A escolha do elemento que será incluído na solução inicial é feita com o auxílio de uma função baseada em método guloso (*greedy*). A cada iteração, será criada uma lista de elementos que poderão ser incluídos na solução, sendo esta lista denominada Lista Restrita de Candidados (LRC), onde cada candidato tentará causar um mínimo aumento no custo da solução que está sendo construída.

A partir da LRC, um elemento será escolhido aleatoriamente (*ramdomized*) para fazer parte da solução. Como, a cada iteração, a solução é modificada, pois é incluído um novo elemento, a Lista Restrita de Candidados deverá ser recalculada para se adaptar (*adaptative*) a nova realidade.

```
Procedimento Construção_Solução(Semente)
1: Solução = 0;
2: Avalie o custo incremental dos elementos candidatos;
3: Enquanto Solução não está completa Faça
4:     Crie a Lista Restrita de Candidatos (LRC);
5:     Selecione um elemento e em LRC aleatoriamente;
6:     Solução = Solução ∪ {e};
7:     Recalcule os custos dos elementos candidatos
8: Fim;
9: Retorne Solução;
Fim Construção_Solução.
```

Figura 3.3: Pseudo-Código do procedimento de Construção da Solução inicial.

Após a etapa de construção da solução inicial, o GRASP irá efetuar a melhoria desta solução, através da busca (*search*) local na vizinhança da solução atual. Esta fase termina quando não for possível encontrar uma solução melhor na vizinhança da solução atual.

O pseudo-código do procedimento de busca local é apresentado na Figura 3.4, e esse procedimento é iniciado a partir da solução obtida na primeira fase descrita anteriormente.

```
Procedimento Busca_Local(Solução)
1: Enquanto Solução não é um Ótimo Local Faça
2:     Encontre s' ∈ V(Solução) tal que f(s') < f(Solução)
3:     Solução = s';
4: Fim;
5: Retorna Solução;
Fim Busca_Local.
```

Figura 3.4: Pseudo-Código do procedimento de Busca Local.

O desempenho da busca local dependerá de vários fatores, como por exemplo: a solução inicial utilizada, a verificação do custo da solução, a estrutura utilizada para os vizinhos da solução, a técnica de busca pela vizinhança da solução, entre outros. A estrutura mais utilizada para a vizinhança de uma solução é a 2-trocas, onde 2 elementos da solução são permutados para a geração da solução vizinha.

Maiores informações sobre a metaheurística GRASP serão encontradas em [15] e [31].

3.2.2 Busca Tabu

Busca Tabu é um método de otimização, relacionado a categoria das técnicas de busca local. A Busca Tabu melhora o desempenho de uma busca local através da utilização de estruturas de memória. Este método utiliza um mecanismo de busca na vizinhança V(x), definida a partir de x, para poder ir de uma solução x para uma solução $x' \in V(x)$, até que um critério de parada seja alcançado [26]. O vizinho de uma solução x é a solução x' que sofreu uma modificação m, ou seja, $x' = x \oplus m$[12].

Esta metaheurística está aprimorando a habilidade de resolução de problemas de importância prática. As atuais aplicações da Busca Tabu estão expandindo os horizontes nas áreas de planejamento de recursos, telecomunicações, desenho de micro-chips (VLSI), análise financeira, distribuição de energia, engenharia molecular, logística, classificação de padrões, produção, gerenciamento de lixo, exploração mineral, análise médica, conservação do ambiente e uma série de outras aplicações [19].

Para analisar regiões que estão na área da busca, mas que poderiam se tornar inexploradas por um método de busca local, a Busca Tabu atualiza a vizinhança de cada solução à medida que o procedimento evolui, desta forma, ótimos locais são evitados favorecendo a convergência da Busca Tabu.

Obtém-se um ganho na eficiência do mecanismo de busca, ao armazenar a solução atual e o caminho percorrido até esta solução. Enquanto a maioria dos métodos de busca armazenam apenas o valor da f(x*) da melhor solução x* visitada, o método de Busca Tabu irá manter, além disso, informações relacionadas à exploração que resultou da última solução encontrada. Esta informação será utilizada para se encontrar a próxima solução a ser verificada, ou seja, a partir de x passa-se para x', como pode ser visto na Figura 3.5. Além disso, é admitido que haja soluções piores que a solução atual para escapar de ótimos locais [18].

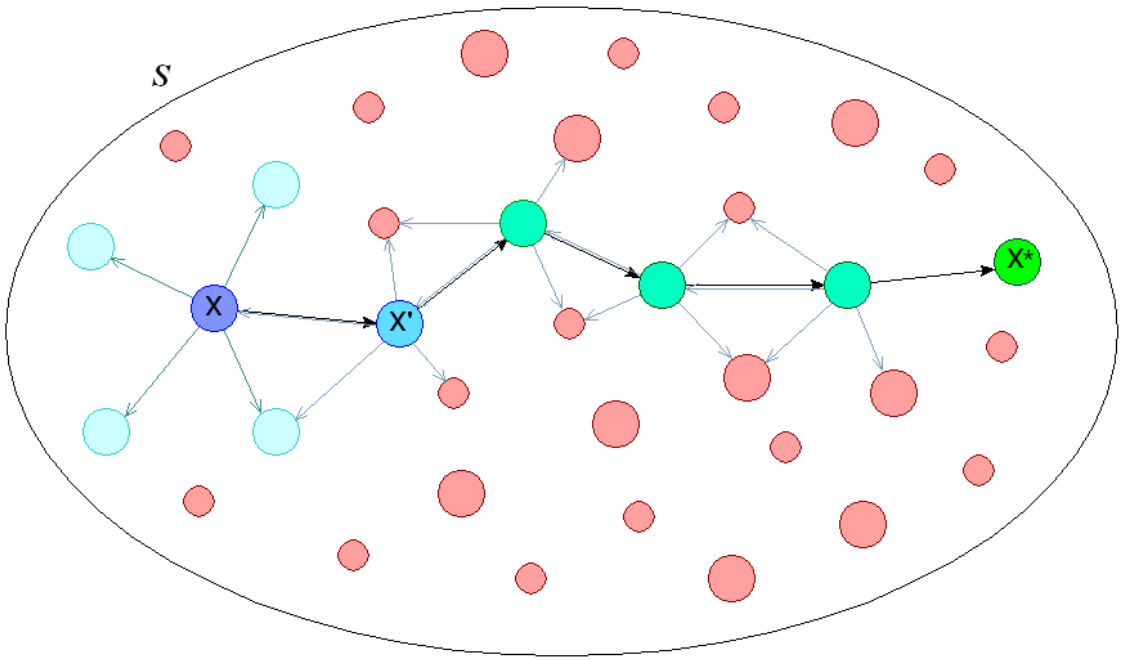


Figura 3.5: Evolução da busca da solução

Uma característica importante na Busca Tabu, é que independente de f(x') ser melhor ou pior que f(x), x' será sempre a nova solução corrente. Para evitar que haja algum retorno a uma solução previamente gerada, o algoritmo utiliza o conceito de Lista Tabu, que define todos os movimentos com um certo atributo como sendo tabu por um determinado número de iterações, chamado tempo tabu.

Esta lista é atualizada quando é verificada a vizinhança de x, para armazenar os movimentos mais recentes e remover movimentos considerados antigos (realizados em iterações anteriores). Os movimentos na Lista Tabu não serão realizados, a não ser que a solução satisfaça um determinado critério, por exemplo, a melhoria da solução encontrada até o momento. O critério adotado é chamado de Condição de Aspiração [27]. O procedimento finaliza quando um determinado critério

de parada é alcançado, sendo o mais adotado, a quantidade de iterações sem que haja melhora da solução.

O pseudo-código da Busca Tabu é descrito na Figura 3.6:

```
Procedimento Busca_Tabu
1:  Seja x0 a solução inicial;
2:  x = x0;
3:  x* = x0;
4:  Iter = 0;      {Contador do número de iterações}
5:  MelhorIter = 0;{Iteração mais recente que forneceu x*}
6:  Seja BTMax o máximo de iterações sem melhora em x*;
7:  T = {}; {Lista Tabu}
8:  Inicialize a função de aspiração A;
9:  Enquanto ( Iter – MelhorIter <= BTMax )
10:     Iter = Iter + 1;
11:     Criação de V(x)
12:     Seja x' = x⊕m tal que m ∉ T ou
        x' atenda a condição de aspiração (f(x') < A(f(x)));
13:         x = x';
14:         Se f(x) < f(x*)
15:         Então x* = x;
16:          Atualize a função de aspiração A;
17:          MelhorIter = Iter;
18:         Fim Se;
19: Fim Enquanto;
20: Retorno x*;
Fim Busca_Tabu.
```

Figura 3.6: Algoritmo para a Busca Tabu

Na linha 11 será criada a vizinhança de x, onde serão visitados os elementos $\{x_1', x_2', ..., x_k'\}$, e serão armazenados os movimentos $\{m_1, m_2, ..., m_k\}$ na lista tabu, é neste momento também que os movimentos antigos serão excluídos da lista tabu.

Na linha 12 é feita a escolha de x', apenas nos casos em que o movimento m ainda não tenha sido efetuado, ou se todos os movimentos de x já foram efetuados, mas f(x') atende a condição de aspiração.

Na linha 13 pode ser visto que x' sempre será escolhido como solução atual, mas a solução ótima só será escolhida se o valor da f(x) for menor que o valor da f(x*), visto nas linhas 14, 15.

Maiores informações sobre Busca Tabu podem ser encontradas em [39].

3.2.3 Colônia de Formigas

A metaheurística Colônia de Formigas foi definida por Dorigo [13] como sendo um sistema multi-agente, inspirado na organização de um formigueiro real, em um estudo feito pelo pesquisador S. Goss no trabalho "*Self Organized Shortcuts in the Argentine Ant*". Neste estudo, uma colônia de formigas tem acesso à comida através de um caminho que possui duas pontes, uma pequena e outra grande. As pontes foram organizadas de tal forma que, cada formiga, ao caminhar do formigueiro para a comida, deveria escolher uma das duas pontes para passar, tanto na ida quanto na volta, como pode ser visto na Figura 3.7.

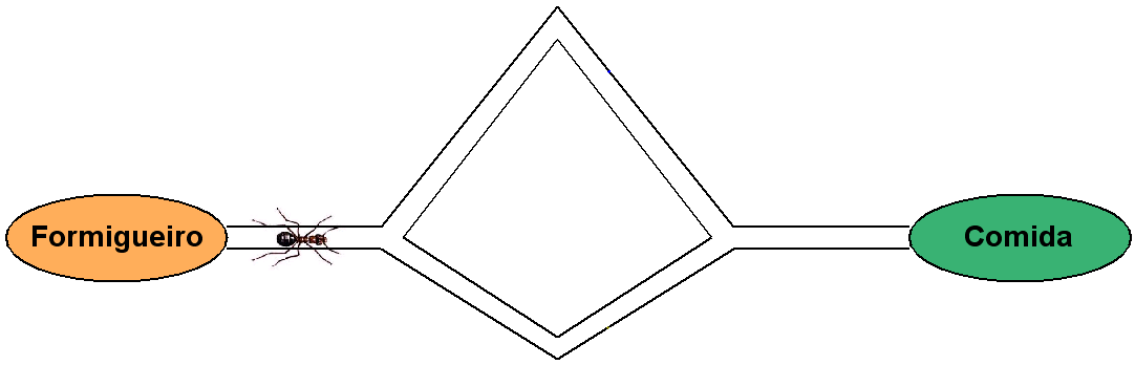


Figura 3.7: Formiga irá do formigueiro para a comida escolhendo 1 de 2 caminhos

O experimento mostrou que após alguns instantes, a maioria das formigas passou a escolher a ponte de menor comprimento, tanto na ida quanto na volta. E ainda, que a escolha do menor caminho acontecia com maior velocidade a medida que se aumentava a diferença de tamanho entre as pontes.

Enquanto as formigas percorriam o caminho entre o formigueiro e a comida, era liberado uma substância chamada *ferômonio* pelo caminho. As formigas escolhem o caminho por qual irão aleatoriamente, baseadas na quantidade de feromônio que existe no caminho. Inicialmente não existe feromônio nos dois caminhos, e as formigas que escolhem o caminho menor, chegam mais rapidamente a comida e voltam mais rapidamente para o formigueiro, em relação as formigas que escolhem o caminho mais longo. Desta forma, a quantidade de ferômonio depositada no caminho mais curto, com o passar do tempo, será muito maior que a quantidade de feromônio no caminho mais longo, então, quando as formigas tiverem que decidir por qual caminho passar, haverá uma tendência maior de que o caminho mais curto seja escolhido, até que todas as formigas passem a utilizar o menor caminho.

Uma característica que esta metaheurística deve possuir é a garantia de evaporação do feromônio com o passar do tempo, pois sem isso, os caminhos ficarão sempre marcados, não havendo distinção entre o caminho mais utilizado e os outros caminhos.

O pseudo-código para a metaheurística Colônia de Formigas é apresentado na Figura 3.8:

```
Procedimento Colônia de Formigas()
1: Enquanto (critério de parada não satisfeito) Faça
2:     Geração e Ativação de formigas();
3:     Evaporação do feromônio();
4: Fim Enquanto
Fim do Procedimento
```

```
Procedimento Geração e Ativação de formigas();
1: Enquanto (Recursos estão disponíveis)
2:     Agendar a criação de uma nova formiga();
3:     Nova formiga ativa();
4: Fim Enquanto
Fim do Procedimento
```

```
Procedimento Nova formiga ativa()
1:  Inicializar formiga();
2:  M = Atualizar memória da formiga();
3:  Enquanto (Estado Atual ≠ Estado Procurado)
4:      A = Ler Tabela Local de Rotas da formiga();
5:      P = Calcular Probabilidade de Transição(A,M,Ω)
6:      Próximo Estágio = Aplicar Decisão da Formiga(P,Ω);
7:      Ir para o próximo estágio(Próximo Estágio);
8:      Se(Atualização Automática do Feromônio passo-a-passo)
9:            Deposite Feromônio no arco visitado();
10:           Atualizar Tabela de Rotas da formiga();
11:     Fim Se
12:     M = Atualizar Estágio Interno();
13: Fim Enquanto
14: Se(Atualização Automática do Feromônio com atraso)
15:     Para cada arco visitado ∈ Ψ faça
16:          Depositar Feromônio no arco visitado();
17:          Atualizar Tabela de Rotas da formiga();
18:     Fim Para cada
19: Fim Se
20: Morra();
Fim do Procedimento
```

Figura 3.8: Pseudo-Código para a metaheurística Colônia de Formigas

O parâmetro Ψ representa o conjunto de soluções viáveis do problema.

O parâmetro Ω do pseudo-código é definido da seguinte forma:

- Dado um conjunto de componentes C = { c_1, c_2, c_3,..., c_{n_c}},
- define-se o conjunto L de possíveis conexões/transições entre os elementos de C sobre um subconjunto $\overline{C}$ do produto cartesiano C x C:

$$L = \{ l_{c_i c_j} . (c_i, c_j) \in \overline{C} \}, [\![L]\!] \leq n_c^2$$

- Para cada $l_{c_i c_j} \in L$ uma função de custo da conexão $J_{c_i c_j} = J(l_{c_i c_j}, t)$, parametrizado por alguma medida de tempo t, é definida.
- O conjunto Ω é então o conjunto finito de restrições Ω =Ω (C,L,t).

Maniezzo [34] formulou o seguinte conjunto de regras que devem ser observadas quando se faz a utilização desta metaheurística no estudo do PQA:

1 – Quando a atividade *j* for associada a localização *i*, é deixado o feromônio τ_{ij} marcando esta alocação (i,j);

2 – Será escolhido uma alocação de uma atividade a uma localização com uma probabilidade baseada em uma função η_{ij} que avaliará o potencial de melhora e a quantidade de feromônio presente na alocação (i,j).

3 – Para construir uma permutação completa, localidades e atividades que já estão associadas são fixadas até que todas as atividades sejam alocadas.

Maiores informações sobre Colônia de Formigas serão encontrados em [16], [14] e [49].

3.2.4 HeuristicHead

A heurística HeuristicHead é uma heurística híbrida, combinando um método construtivo de soluções iniciais com uma técnica de melhoramento da busca local [46]. Na etapa de construção das soluções iniciais e no melhoramento prévio é utilizado a matriz HeadQ, esta matriz é definida da seguinte forma: utilizando-se o algorítmo de construção de soluções viáveis apresentado por Rangel [42] cria-se a lista de soluções viáveis para o problema, como pode ser visto na Figura 3.9:

```
Procedimento SolViável
1:  Entrada: ρ, φF e φ-1D;
2:  ξ = φ-1D o ρ o φF;
3:  Para t = 1, ... , N Faça
4:      ListaIJ[t] = Ψ-1(t);
5:      ListaKL[t] = Ψ-1(ξ(t));
6:  Fim Para
7:  Se ∃ ρ ∈ (1,...,n) tal que φ(1)=ρ para as (n-1) primeiras combinações
8:  Então
8:      φ(1) = ρ
9:      Para i = 2,...,n Faça
10:         [φ(i) = l ≠ p] ∨ [ φ(i) = k ≠ p] para ListaKL[i - 1];
11:     Fim Para
12:         Se φ construída possui todas as N combinações compatíveis
13:         Então
13:             ρ é viável;
14:         Senão
15:             ρ não é viável;
16:         Fim Se
17: Senão
18:     não é viável;
19: Fim Se
Fim do Procedimento
```

Figura 3.9: Pseudo-Código para o Procedimento SolViável

Este algorítmo se tornará ineficiente se for realizada a enumeração de todos as soluções lineares para se encontrar as soluções viáveis. Não há necessidade de se verificar todas as soluções lineares do problema, precisa-se apenas das n! primeiras soluções, sendo estas chamadas de pseudo-viáveis, sendo este teorema demonstrado por Rangel em [19], e o pseudo-código é apresentado na Figura 3.10:

```
Procedimento ConstróiHead(N)
1: Para t = 1, ... , N Faça
2:     ListaIJ[t] = Ψ-1(t) = (i,j)
3:     HeadQ[i,j - 1] = t
4:     HeadQ[j,i] = t
5: Fim Para
Fim do Procedimento
```

Figura 3.10: Pseudo-Código para o Procedimento ConstróiHead

Na fase de melhoramento das soluções geradas é usado uma método de busca local em torno de uma vizinhança de cada solução. O pseudo-código desta heurística pode ser visto na Figura 3.11:

```
Procedimento HeuristicHead(n)
1: Entrada de Dados;
2: Construção das soluções iniciais na matriz HeadQ;
3: Para i =1, ... , n Faça
4:     Constróiφ(HeadQ[i]);
5:     BuscaLocal(φ);
6:     Atualizar Solução;
7: Fim Para
8: Retorna a melhor solução encontrada;
Fim do Procedimento
```

Figura 3.11: Pseudo-Código para o algoritmo HeuristicHead

A fase de construção das soluções iniciais, é composta por várias etapas:

I - Leitura dos dados;

II - Ordenação dos vetores de Fluxo e Distância;

III - Construção da ϕ_F e ϕ_D;

IV - Construção da matriz *HeadQ*;

V - Construção da matriz $HeadQ^*$;

VI - Ordenação das linhas na matriz $HeadQ^*$, com as respectivas trocas da matriz *HeadQ*.

O itens II e III devem pertencer a todo algoritmo que trabalhe com uma correspondência entre o Problema Quadrático de Alocação e sua relaxação na forma do Problema de Alocação Linear, já que são os procedimento justamente encarregados desta relaxação.

A construção da matriz *HeadQ* é independente dos procedimento I, II e III. A construção da *HeadQ* e da $HeadQ^*$ é o que caracteriza esta Heurística. Uma vez de posse das permutações lineares será realizada a ordenação de tais permutações, com base no Teorema das Inversões, para ser utilizada como critério de uma melhoria prévia das soluções iniciais. Essa melhoria acelera o procedimento de busca local, consequentemente, aumentando o desempenho da heurística [46].

Capítulo 4

Características especiais das instâncias do PQA

Vários pesquisadores têm realizado estudos sobre o PQA e suas características. Uma das características que o PQA possui é a Matriz Q e Q*, sendo esta a matriz que contém todos os valores possíveis obtidos pela associação das matrizes de fluxo e distância, como foi apresentado no capítulo 1, na Figura 2.4 e 2.5.

Três características serão abordadas neste capítulo, a primeira refere-se ao Isomorfismo no PQA, a segunda é a Família de Instância obtida a partir do estudo da Matriz Q*, e a terceira refere-se ao Número de Inversões de uma instância do PQA.

4.1 Isomorfismo no PQA

Na Teoria dos Grafos, o estudo do isomorfismo é definido como sendo a relação entre dois grafos onde são mantidas as seguintes características: quantidade de vértices, a quantidade de arestas e a preservação das arestas. Mais precisamente, o isomorfismo entre grafos existe quando houver uma bijeção entre os nós dos grafos que preserve as arestas [52]. No Isomorfismo de Instâncias do PQA, além das características citadas, deve se verificar outra característica importante: as arestas que compartilham um mesmo nó em um grafo, também compartilham um mesmo nó no grafo isomorfo. O isomorfismo de instâncias no PQA pode ser facilmente entendido quando se utiliza a representação por cliques valoradas da instância do problema (Seção 2.6).

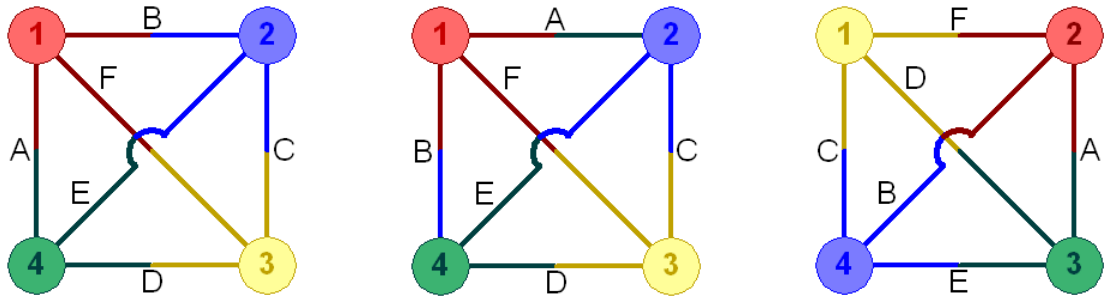


Figura 4.1: Isomorfismo no PQA

Na Figura 4.1 são apresentados 3 grafos isomorfos *a*, *b* e *c*. A diferença entre o grafo da

Figura 4.1a e o grafo da Figura 4.1b é que o segundo não manteve as arestas dos nós 2 e 4 (do primeiro) em um mesmo nó. No grafo da Figura 4.1c todas as arestas estão em posição diferente da apresentada na Figura 4.1a, mas todas as arestas que compartilham um nó no grafo 4.1a continuam compartilhando um nó no grafo 4.1c.

A manutenção desta característica é importante, pois cada aresta do grafo representa uma característica comum entre 2 nós, por exemplo, o fluxo de material entre duas instalações, ou a distância entre duas salas. Esta característica não pode ser perdida, pois seria o mesmo que haver o particionamento de uma instalação ou o rompimento de uma estrada entre duas localidades.

Formalmente, define-se o Isomorfismo de Instâncias do PQA da seguinte maneira: Sejam duas instâncias PQA(F_1,D_1) e PQA(F_2,D_2), onde K_{F_1}, K_{F_2}, K_{D_1} e K_{D_2} são cliques valoradas que representam fluxos e distâncias, então PQA(F_1,D_1) será isomorfo a PQA(F_2,D_2), quando K_{F_1} for isomorfo a K_{F_2} e K_{D_1} for isomorfo a K_{D_2}:

$$PQA(F_1,D_1) \approx PQA(F_2,D_2) \leftrightarrow K_{F_1} \approx K_{F_2} \text{ e } K_{D_1} \approx K_{D_2}.$$

Esta definição foi verificada por Abreu et al [1].

4.2 Famílias de Instâncias

Vários problemas ao terem seus vetores $\vec{F}$ e $\vec{D}$ reduzidos à forma $\vec{F}^-$ e $\vec{D}^+$, chegam aos mesmos vetores e possuem a Matriz Q* em comum, essa característica foi utilizada por Abreu et al [1] para classificar as instâncias do PQA em famílias de instância. A família é definida como:

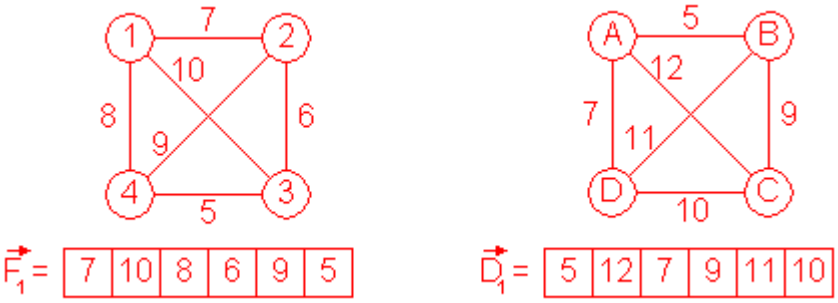
$$RelClass(F^-, D^+) = \{PQA(F, D) : (F^-)(D^+)^t = Q^*\} \quad (4.1)$$

Onde $\forall$ F e $\forall$ D as coordenadas sejam as mesmas, e a ordenação seja diferente. Dentre as instâncias pertencentes à RelClass, PQA($\vec{F}^-$, $\vec{D}^+$) é denominada de *instância padrão*, e sua resolução é feita em tempo polinomial [33].

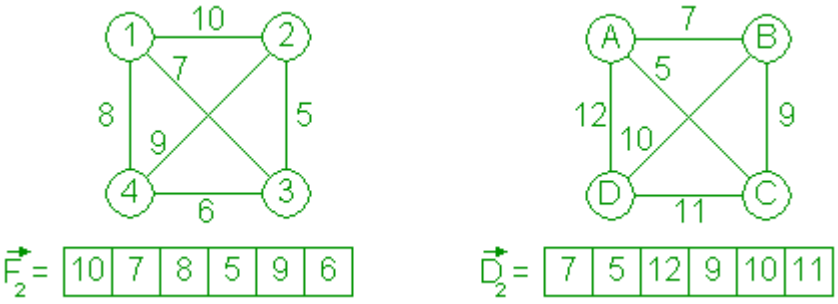
A seguir serão apresentados algumas instâncias de problemas que serão utilizados para

exemplificar a caracterização de uma família de instâncias.

Na Figura 4.2 são exibidas duas instâncias do PQA:



(a)



(b)

Figura 4.2: Instâncias da mesma família

Estas duas instâncias, quando são reduzidas para o formato $\vec{F}^-$ e $\vec{D}^+$ chegam aos mesmos vetores: $\vec{F}^-$ = {10,9,8,7,6,5} e $\vec{D}^+$ = {5,7,9,10,11,12}. Consequentemente, obtém-se a mesma matriz Q*, como pode ser visto na Figura 4.3:

	10	9	8	7	6	5
5	50	45	40	35	30	25
7	70	63	56	49	42	35
9	90	81	72	63	54	45
10	100	90	80	70	60	50
11	110	99	88	77	66	55
12	120	108	96	84	72	60

*Figura 4.3: Matriz Q**

Mas para estas duas instâncias os valores ótimos são diferentes, para a primeira o valor ótimo de solução é: 389, com solução igual a $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}$, enquanto para a segunda instância os resultados obtidos são: 382 e $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$.

A diferença nas soluções ótimas para as duas instâncias que pertencem a mesma família ocorreu pois os grafos que representam as instâncias não possuem um isomorfismo plenoo. A instância apresentada na Figura 4.4 é isomorfa a instância da Figura 4.2a:

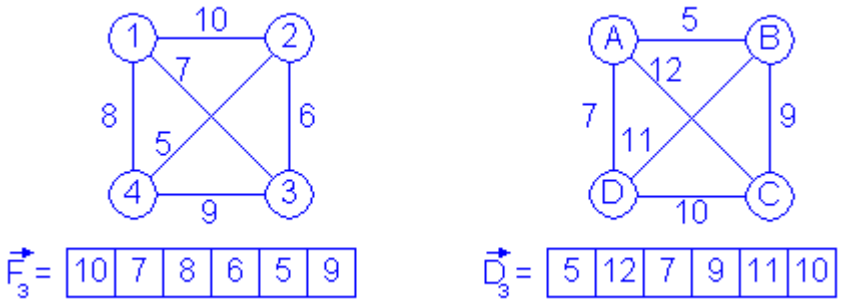


Figura 4.4: Instância do PQA

Ao se resolver a instância 4.4, encontra-se o valor ótimo 389, e a permutação obtida é igual a permutação $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}$. Esta instância preserva o agrupamento das arestas, em relação a instância 4.2a, ou seja, neste caso tem-se um isomorfismo pleno entre os dois grafos que definem as duas instâncias do PQA. Além disso, a instância da Figura 4.4 tem seus vetores $\vec{F}^-$ e $\vec{D}^+$ com uma sequência numérica bem próxima da instância padrão.

Em testes realizados com um conjunto de instâncias do QAPLIB, foi verificado empiricamente que instâncias parcialmente ordenadas, para ficarem em um formato mais próximo da instância padrão, eram mais facilmente resolvidas que as instâncias originais [50].

4.3 Teorema das Inversões

Antes de enunciar o Teorema das Inversões serão apresentados dois conceitos que auxiliam a sua compreensão:

- Uma inversão em uma permutação é definido como sendo o par ($\pi(\iota)$, $\pi(j)$) onde $\pi(j) < \pi(i)$ com $i < j$, $i,j = 1,2,...N$. Seja $\vartheta(\pi)$ o conjunto de pares de inversões de π, o número de inversões será definido como sendo a cardinalidade do conjunto $\vartheta(\pi)$, ou seja, $|\vartheta(\pi)|$.

- Seja Π_N o conjunto das permutações de {1,2,..,N} e considerando o conjunto dos produtos escalares definido como $C(Z_\rho)=\{Z_\rho=\langle F^-, \rho(D^+)\rangle : \rho \in \Pi_N\}$, tem-se que duas permutações são ditas livremente comparáveis quando $Z_{\rho 1} \leq Z_{\rho 2}$ ou $Z_{\rho 2} \leq Z_{\rho 1}$, independente das coordenadas dos vetores $\vec{F}^-$ e $\vec{D}^+$.

O Teorema das Inversões diz que "a ordenação dos custos das permutações livremente comparáveis é compatível com a ordenação do número de inversões destas soluções", ou seja, o número de inversões decresce (ou cresce) de acordo com o valor do custo das permutações [42].

Sabe-se que o conjunto das soluções do PQA(F,D) está contido no conjunto das soluções do PAL(Q*), mas nem todas as soluções do PAL(Q*) estão contidas no conjunto de soluções do PQA(F,D). Com a utilização do Teorema das Inversões pode-se realizar uma associação do custo de uma solução do PAL(Q*) com o seu número de inversões, tornando-se um parâmetro para a avaliação da qualidade das soluções encontradas no PQA[42]. A verificação da qualidade de uma solução é feita comparando-se o número de inversões da solução correspondente no PAL(Q*) , lembrando que soluções do PQA e PAL são representadas por permutações.

A partir da matriz Q* obtém-se a melhor solução do PAL(Q*) realizando o cálculo do traço da matriz, e que a solução encontrada não possui inversões, pois ela é formada pelos vetores $\vec{F}^-$ (ordenado de forma não-crescente) e $\vec{D}^+$ (ordenado de forma não-decrescente). Esta característica motivou a seguinte questão:

"Um problema PQA(F,D) que tenha seus vetores $\vec{F}$ e $\vec{D}$ ordenados de forma que se obtenha dois vetores $\vec{F}'$ e $\vec{D}'$ com menor número de inversões, e que as cliques K_F e K_D possuam um isomorfismo pleno com as cliques $K_{F'}$ e $K_{D'}$, formando o problema PQA(F',D') terá uma melhor solução ou, de acordo com as características da heurística utilizada, será mais fácil de resolver ?"

Para responder esta questão, foi criado um procedimento que realiza a ordenação da instância do PQA(F,D) formando o PQA(F',D'), sendo que o objetivo da ordenação é de obter-se vetores com um menor número de inversões, e que seja mantido o isomorfismo pleno entre os problemas.

Também foi realizada uma bateria de testes com as heurísticas GRASP, Busca Tabu, Colônias de Formigas e um algoritmo construtivo desenvolvido por Resendo e Rangel [46] para se verificar o comportamento da resolução dos problemas PQA(F,D) e PQA(F',D').

Capítulo 5

Ordenação das instâncias do PQA

A ordenação da instância é realizada através de um algoritmo que irá redistribuir os valores dentro das vetores $\vec{F}$ e $\vec{D}$ de forma que o vetor $\vec{F}$ se aproxime do vetor $\vec{F}^{-}$, e o vetor $\vec{D}$ se aproxime do vetor $\vec{D}^{+}$. Desta forma estaremos tentando facilitar a resolução do PQA através da instância isomorfa.

5.1 Algoritmo de Ordenação

O algoritmo de ordenação ideal das matrizes trabalharia da seguinte forma: a ordenação deve ser feita de forma que a matriz final esteja com todos os seus elementos em ordem não-decrescente (ou não-crescente). Mas a ordenação também tem que garantir que a matriz final possua isomorfismo pleno em relação a matriz primitiva, não havendo descaracterização do problema original. Assim, o algoritmo de ordenação deve trabalhar da seguinte forma: a ordenação deve ser feita de forma que a matriz final esteja com a maior quantidade de elementos em ordem não-crescente (ou não-decrescente), estabelecendo um isomorfo pleno em relação à matriz original. A Figura 5.1 ilustra um exemplo onde uma matriz foi ordenada com o objetivo de deixar todos os seus elementos em ordem não-crescente:

0	5	9	14	26	7	2
5	0	3	16	1	6	7
9	3	0	5	9	2	6
14	16	5	0	4	9	21
26	1	9	4	0	3	11
7	6	2	9	3	0	8
2	7	6	21	11	8	0

(a)

0	26	21	16	14	11	9
26	0	9	9	8	7	7
21	9	0	6	6	5	5
16	9	6	0	4	3	3
14	8	6	4	0	2	2
11	7	5	3	2	0	1
9	7	5	3	2	1	0

(b)

Figura 5.1: Ordenação completa de uma Matriz para a forma não-decrescente

A primeira matriz na Figura 5.1a, mostra a configuração inicial do problema antes da ordenação. As cores foram utilizadas para agrupar os elementos que compartilham um mesmo nó no grafo, e também para facilitar a visualização do resultado da ordenação. Neste caso, a descaracterização pode ser vista através da desconexão dos elementos de mesma cor, na Figura 5.1b, que agora apresenta, por exemplo, os elementos de cor verde ocupando seis linhas e colunas diferentes, isto aconteceu porque na ordenação, as arestas que estavam todas agrupadas em um

mesmo nó inicialmente, foram divididas em outros nós.

Na Figura 5.2 são apresentadas duas matrizes com isomorfismo pleno:

0	5	9	14	26	7	2
5	0	3	16	1	6	7
9	3	0	5	9	2	6
14	16	5	0	4	9	21
26	1	9	4	0	3	11
7	6	2	9	3	0	8
2	7	6	21	11	8	0

(a)

0	26	2	14	5	7	9
26	0	11	4	1	3	9
2	11	0	21	7	8	6
14	4	21	0	16	9	5
5	1	7	16	0	6	3
7	3	8	9	6	0	2
9	9	6	5	3	2	0

(b)

Figura 5.2: Ordenação parcial de uma Matriz para a forma não-decrescente

Neste caso, os elementos de mesma cor continuaram ocupando apenas uma linha e coluna diferente, mesmo tendo mudado de posição em relação à matriz. Como foi visto no capítulo 4, o mesmo valor da solução ótima será encontrada para ambos os problemas. A estratégia de ordenação adotada é a seguinte: a ordenação será feita em cada linha da matriz, havendo posteriormente uma normalização da matriz para que ela fique no formato correto, isto é, simétrica com diagonal principal nula, isto pode ser visto na Figura 5.3:

0	5	9	14	26	7	2
5	0	3	16	1	6	7
9	3	0	5	9	2	6
14	16	5	0	4	9	21
26	1	9	4	0	3	11
7	6	2	9	3	0	8
2	7	6	21	11	8	0

(a)

0	26	14	9	7	5	2
5	1	16	3	6	0	7
9	9	5	0	2	3	6
14	4	0	5	9	16	21
26	0	4	9	3	1	11
7	3	9	2	0	6	8
2	11	21	6	8	7	0

(b)

0	26	14	9	7	5	2
26	0	4	9	3	1	11
14	4	0	5	9	16	21
9	9	5	0	2	3	6
7	3	9	2	0	6	8
5	1	16	3	6	0	7
2	11	21	6	8	7	0

(c)

Figura 5.3: Ordenação da primeira linha da matriz

Após a primeira iteração, a primeira linha da matriz está ordenada de forma não-decrescente, passa-se então a ordenação da linha seguinte, como pode ser visto na Figura 5.4:

0	26	14	9	7	5	2
26	0	4	9	3	1	11
14	4	0	5	9	16	21
9	9	5	0	2	3	6
7	3	9	2	0	6	8
5	1	16	3	6	0	7
2	11	21	6	8	7	0

(a)

0	26	2	9	14	7	5
26	0	11	9	4	3	1
14	4	21	5	0	9	16
9	9	6	0	5	2	3
7	3	8	2	9	0	6
5	1	7	3	16	6	0
2	11	0	6	21	8	7

(b)

0	26	2	9	14	7	5
26	0	11	9	4	3	1
2	11	0	6	21	8	7
9	9	6	0	5	2	3
14	4	21	5	0	9	16
7	3	8	2	9	0	6
5	1	7	3	16	6	0

(c)

Figura 5.4: Ordenação da segunda linha da matriz

A ordenação final, mostrada na Figura 5.4c, afetou a primeira linha, isto aconteceu pois a matriz resultante deve ser isomorficamente plena em relação a matriz primitiva, e a ordenação seguirá até que todas as linhas estejam ordenadas.

O pseudo-código para o algoritmo de ordenação é apresentado na Figura 5.5:

```
Procedimento Ordena Matriz(matriz M, vetor v, vetor v', int linha)
1:  n = Tamanho(M);
2:  Para x = 1 até n-2 Faça {da primeira até a antepenúltima linha}
3:      v  = Le Linha(M,x);
4:      v' = Ordena(v,1,n-x);
5:      Para y = 1 até n-x Faça
6:          z[y] = Posição(v,v'[y]); {retorna a posição do elemento
                                      v'[y] no vetor v}
7:      Fim Para
8:      Para i = 1 até n Faça
9:          Para j = 1 até x Faça
10:             M'[i][j] = M[i][j];   {faz uma cópia da linha}
11:         Fim Para
12:         Para j = 1 até n-x Faça
13:             k = z[i] + x;
14:             M'[i][j+x] = M[i][k]; {ajuste das linhas}
15:         Fim Para
16:     Fim Para
17:     Para i = 1 até n Faça
18:         k = z[i] + x;
19:         Para j = 1 até t Faça
20:             M[i+x][j] = M'[k][j]; {ajuste das colunas}
21:         Fim Para
22:     Fim Para
23: Fim Para
Fim do Procedimento
```

Figura 5.5: Pseudo-Código do método de ordenação das matrizes

O comando de repetição principal, na linha 2, irá executar o algoritmo em um determinado intervalo, limitado pelo tamanho da matriz. Mais precisamente, até o limite *n-2*, sendo *n* o tamanho da matriz, pois a última linha contém o valor zero, e a penúltima linha contém apenas um número diferente de zero, não havendo necessidade de ordenação.

Cada linha da matriz será utilizada na construção de um vetor *v*, e este vetor será ordenado formando o vetor *v'*. A partir do vetor *v'* faz-se uma atualização da matriz *M*, nas linhas e colunas, e pelo fato de a matriz ser simétrica, pode-se utilizar o mesmo vetor. Esta atualização é feita da seguinte forma: a partir de *v'* será criado um outro vetor *z*, contendo a posição relativa dos elementos de *v'* em relação ao vetor *v*. Assim, a partir do vetor z será possível fazer a troca dos elementos na matriz *M*.

A ordenação do vetor foi feito com base no algoritmo de Ordenação por Junção (*mergesort*). O pseudo-código da ordenação do vetor pode ser visto na Figura 5.6:

```
Procedimento Ordena(vetor vet, int ini, int fim)
1:  Se (ini=fim)
2:  Então Retorna vet
3:  Senão ini_esq = ini;
4:        fim_esq = ⌊(fim-ini)/2⌋;
5:        vet_esq = Copia(vet,ini_esq,fim_esq);
6:        Ordena(vet_esq,ini_esq,fim_esq);
7:        ini_dir = ⌈(fim-ini)/2⌉;
8:        fim_dir = fim;
9:        vet_dir = Copia(vet,ini_dir,fim_dir);
10:       Ordena(vet_dir,ini_dir,fim_dir);
11: Fim Se
12: j = k = 0;
13: Para i = ini até fim Faça
14:    Se (vet_esq[j] < vet_dir[k])
15:    Então vet[i] = vet_esq[j];
16:          j = j + 1;
17:    Senão vet[i] = vet_dir[k];
18:          k = k + 1;
19:    Fim Se
20: Fim Para
21: Retorna vet;
Fim do Procedimento
```

Figura 5.6: Pseudo-Código do método de ordenação do vetor

O tempo de execução do algoritmo de ordenação da matriz é:

$$T = (n-2).\{(n.\log(n) + (n+1).n/2 + n.[(n+1).n] + n.[(n+1).n/2]\}.$$

A ordem de grandeza deste algoritmo é: $O(n^4)$.

5.2 Resultados Computacionais

Nesta seção serão apresentados os resultados computacionais obtidos a partir dos testes realizados nas instâncias do QAPLIB, em sua forma primitiva, isto é, sem alteração da instância, e na forma isoforma, onde foi aplicado o procedimento de ordenação das matrizes F e D. As instâncias utilizadas para os testes foram: chr12a, chr15a, chr18a, chr20a, chr25a, esc16a, esc32a, nug12, nug15, nug20, nug30, rou20, tai30a e tai60a.

Os testes foram feitos com programas escritos em linguagem C que utilizavam as meta-heurísticas GRASP, Colônia de Formigas, Busca Tabu, *Simulated Annealing*, *HeuristicHead*, e foi utilizado um computador com a seguinte configuração: processador Pentium 4 2,26GHz, com 256 MB de memória principal. O sistema operacional utilizado foi Linux Fedora Core 4, e o compilador utilizado foi o GCC 4.0.

5.2.1 Testes com o GRASP

Como visto no capítulo 3.2.1, a metaheurística GRASP possui duas fases de execução, a primeira é de construção da solução inicial, e na segunda é feita a busca local para encontrar uma melhor solução, e é importante ressaltar que a etapa de construção tem um fator aleatório de construção da solução inicial.

Para os testes, foram realizadas as seguintes estratégias:

1º – O teste foi feito em 3 execuções, sendo que em cada uma havia um valor limite para o número de iterações: nas instâncias com $n < 20$ os limites foram 10, 100 e 1000; nas instâncias com $n \geq 20$ os limites foram 50, 500, 5000.

2º – Como existe um fator aleatório no GRASP, e este fator influencia diretamente o seu desempenho, cada execução de uma instância foi resolvida 10 vezes, para se obter uma média no desempenho do programa naquela execução.

3º – A semente inicial gerada em cada uma das 10 resoluções da instância primitiva foi mantida para a respectiva instância isomorfa, como pode ser visto na tabela 5.1:

primitiva	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
isomorfa	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}

Tabela 5.1: relação das sementes utilizadas na execução do GRASP

A seguir serão mostrados os os resultados obtidos na execução do programa GRASP. Primeiramente são apresentados os resultados referentes ao desempenho de execução, e depois os gráficos para a verificação das inversões foram escolhidas aleatoriamente algumas descidas da árvores de busca local do GRASP, sendo que a mesma foi escolhida na instância primitiva e na isomorfa. Os gráficos apresentam os valores médios em cada uma das 3 execuções.

Na tabela, a coluna “Tempo Max” indica o maior tempo entre as 10 resoluções de cada uma das 3 execuções, a coluna “Tempo Médio” indica a média dos tempos entre as 10 resoluções, e a coluna “Tempo Min” indica o menor tempo entre as 10 resoluções. O mesmo critério é adotado para as colunas “Custo Max”, “Custo Médio” e “Custo Min”. Os gráficos apresentam uma comparação do desempenho entre as instâncias isomorfas e primitivas (resultado obtido pela instância isomorfa sobre o resultado obtido pela instância primitiva), quando o valor for menor que 1, indica que a instância isomorfa teve um desempenho melhor que a instância primitiva, e quando o valor for maior que 1, indica que a instância primitiva foi melhor que a instância isomorfa. Quando não houver indicação de valor no gráfico, significa que a instância isomorfa e primitiva tiveram um desempenho igual.

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr12a-prim	1	0,05	0,03	0,00	5757	5200	4776
chr12a-prim	2	0,40	0,26	0,03	5048	4830	4776
chr12a-prim	3	3,63	1,34	0,02	4776	4776	4776

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr12a-iso	1	0,05	0,03	0,00	5972	5328	4776
chr12a-iso	2	0,40	0,24	0,02	5096	4871	4776
chr12a-iso	3	2,65	0,62	0,08	4776	4776	4776

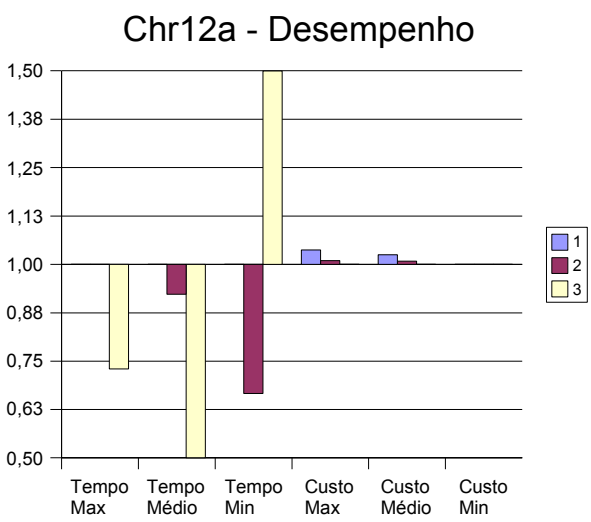


Figura 5.7: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Chr12a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr15a-prim	1	0,13	0,08	0,02	6110	5504	5005
chr15a-prim	2	1,43	0,79	0,03	5352	5157	4948
chr15a-prim	3	13,40	7,96	3,02	5032	4979	4948

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr15a-iso	1	0,15	0,09	0,03	5950	5598	5183
chr15a-iso	2	1,32	0,77	0,03	5489	5252	5035
chr15a-iso	3	14,13	7,81	1,82	5151	5019	4948

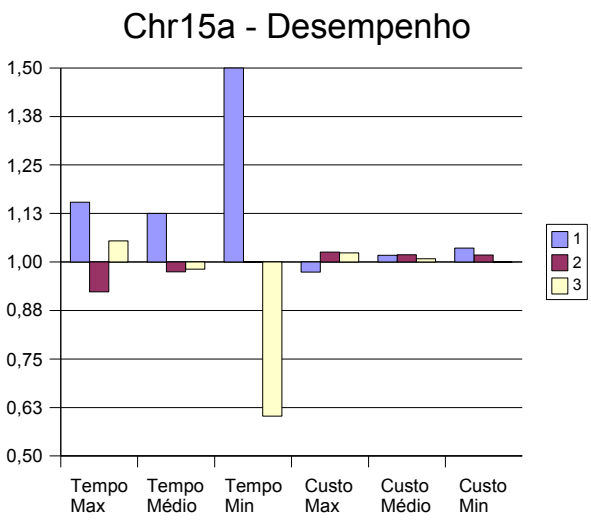


Figura 5.8: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Chr15a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr18a-prim	1	0,37	0,14	0,03	8700	7191	6269
chr18a-prim	2	3,43	1,99	0,08	6788	6423	6194
chr18a-prim	3	30,40	16,31	1,52	6215	5785	5549

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr18a-iso	1	0,35	0,17	0,03	8356	7608	6913
chr18a-iso	2	3,22	1,61	0,10	6916	6335	5998
chr18a-iso	3	29,22	16,22	2,28	6311	5997	5571

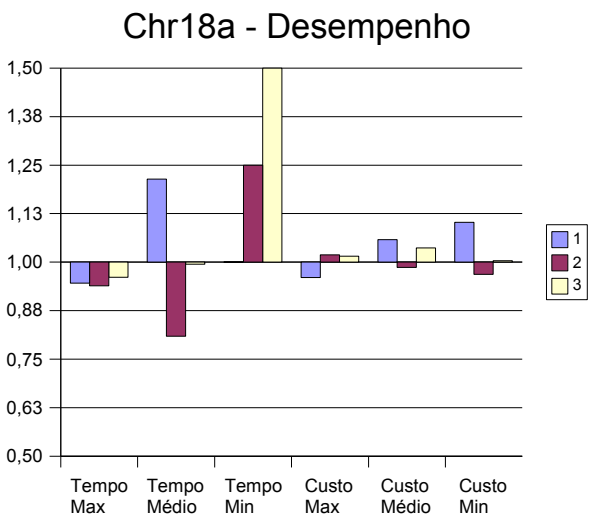


Figura 5.9: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Chr18a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr20a-prim	1	0,62	0,39	0,18	1509	1405	1249
chr20a-prim	2	5,82	2,84	1,23	1318	1267	1186
chr20a-prim	3	51,07	30,12	6,30	1226	1191	1156

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr20a-iso	1	0,57	0,40	0,10	1488	1384	1239
chr20a-iso	2	5,50	3,13	0,47	1338	1264	1203
chr20a-iso	3	54,23	35,10	17,97	1191	1153	1122

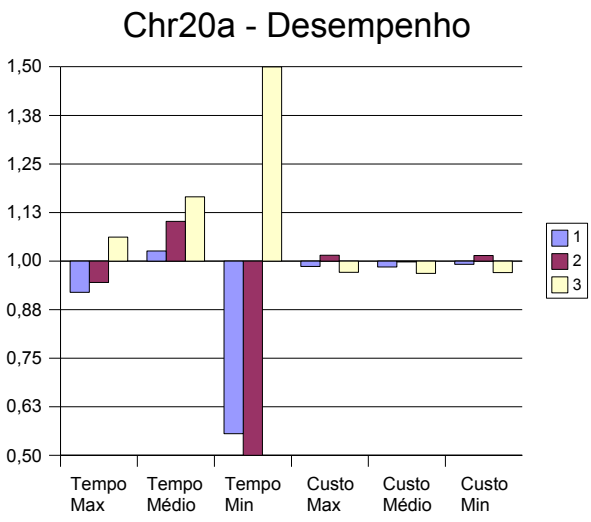


Figura 5.10: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Chr20a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr25a-prim	1	1,82	0,80	0,20	2839	2527	2176
chr25a-prim	2	18,15	10,08	2,12	2418	2328	2210
chr25a-prim	3	193,08	124,00	30,13	2219	2105	1933

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
chr25a-iso	1	1,75	0,95	0,22	2866	2600	2451
chr25a-iso	2	16,58	10,26	3,42	2381	2321	2226
chr25a-iso	3	127,93	83,42	7,07	2311	2220	2050

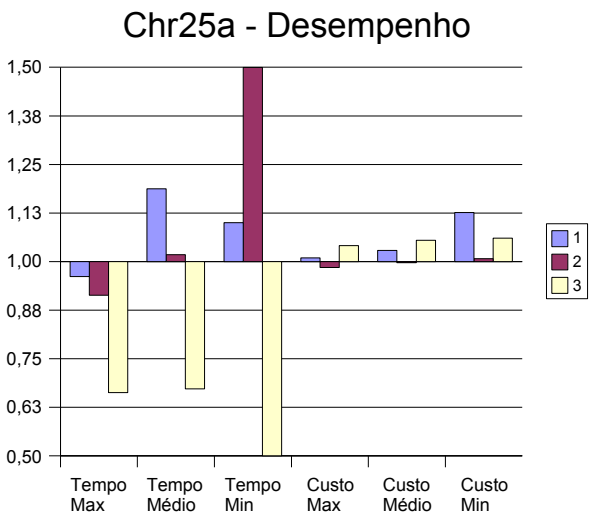


Figura 5.11: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Chr25a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
esc16-prim	1	0,13	0,05	0,02	34	34	34
esc16-prim	2	0,08	0,04	0,02	34	34	34
esc16-prim	3	0,13	0,04	0,02	34	34	34

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
esc16-iso	1	0,07	0,03	0,02	34	34	34
esc16-iso	2	0,12	0,05	0,02	34	34	34
esc16-iso	3	0,03	0,03	0,02	34	34	34

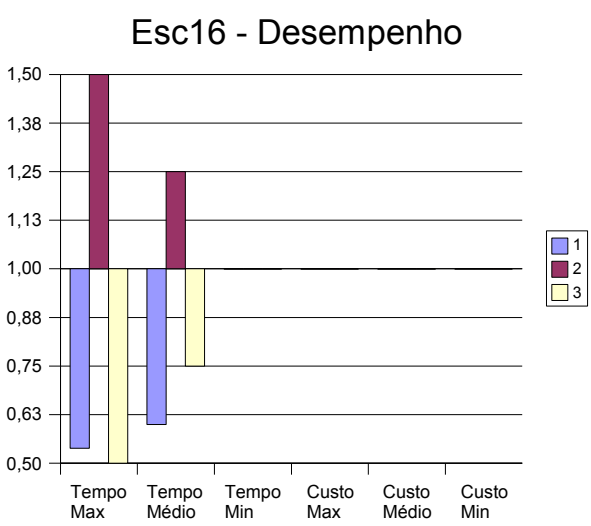


Figura 5.12: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Esc16

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
esc32-prim	1	5,45	2,37	0,67	77	72	69
esc32-prim	2	56,52	34,81	4,12	72	70	68
esc32-prim	3	581,60	267,70	118,47	68	67	66

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
esc32-iso	1	5,08	2,85	0,60	79	73	70
esc32-iso	2	52,43	25,32	2,58	73	70	66
esc32-iso	3	463,05	203,88	48,13	68	67	67

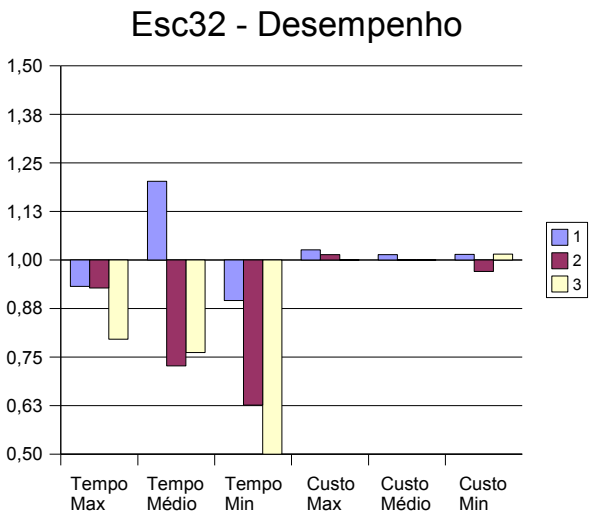


Figura 5.13: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Esc32

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug12-prim	1	0,05	0,02	0,00	301	295	289
nug12-prim	2	0,37	0,19	0,02	291	290	289
nug12-prim	3	1,15	0,58	0,20	289	289	289

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug12-iso	1	0,05	0,02	0,00	297	293	289
nug12-iso	2	0,47	0,17	0,00	291	289	289
nug12-iso	3	0,87	0,33	0,00	289	289	289

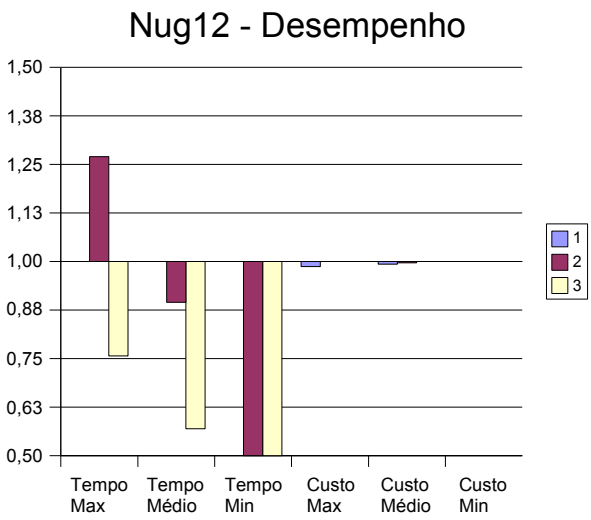


Figura 5.14: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Nug12

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug15-prim	1	0,15	0,08	0,02	591	584	580
nug15-prim	2	1,30	0,70	0,20	578	575	575
nug15-prim	3	2,83	1,18	0,03	575	575	575

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug15-iso	1	0,17	0,09	0,02	592	582	575
nug15-iso	2	1,53	0,58	0,03	576	575	575
nug15-iso	3	2,05	0,43	0,05	575	575	575

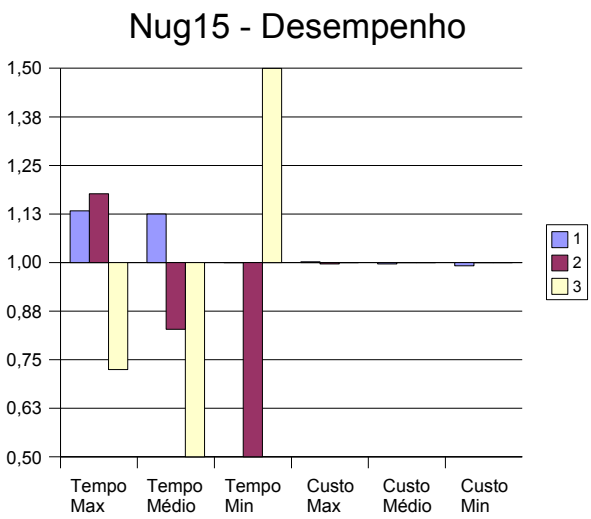


Figura 5.15: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Nug15

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug20-prim	1	0,72	0,42	0,08	1317	1305	1287
nug20-prim	2	5,82	2,18	0,60	1301	1290	1285
nug20-prim	3	44,02	15,75	0,57	1287	1285	1285

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug20-iso	1	0,75	0,32	0,07	1329	1309	1293
nug20-iso	2	6,80	4,16	0,22	1300	1291	1287
nug20-iso	3	35,02	13,94	2,22	1293	1286	1285

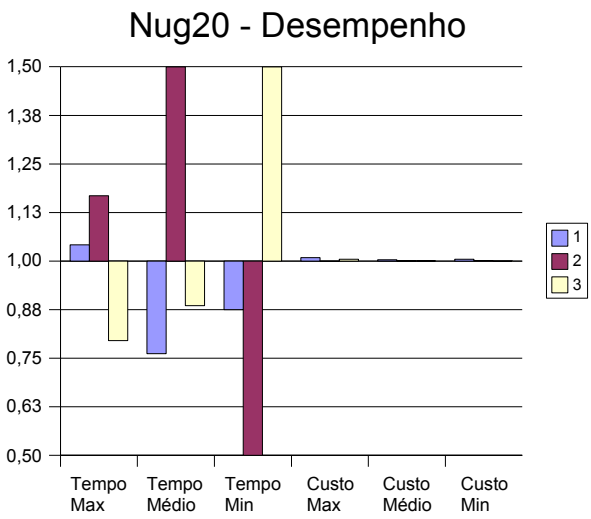


Figura 5.16: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Nug20

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug30-prim	1	5,42	4,75	3,18	3141	3126	3105
nug30-prim	2	53,58	29,88	3,87	3110	3094	3078
nug30-prim	3	574,60	345,80	52,77	3089	3077	3064

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
nug30-iso	1	5,82	3,48	1,23	3139	3112	3097
nug30-iso	2	56,88	31,22	5,00	3114	3092	3071
nug30-iso	3	566,90	203,48	5,78	3086	3078	3062

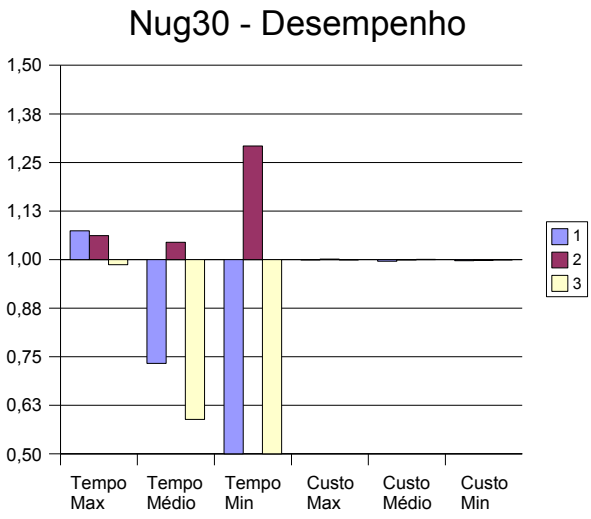


Figura 5.17: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Nug30

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
rou20-prim	1	0,63	0,38	0,07	376730	372001	367990
rou20-prim	2	6,02	3,95	0,60	371115	367320	363927
rou20-prim	3	54,88	34,23	4,73	364799	363700	363050

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
rou20-iso	1	0,50	0,23	0,05	376037	371303	365355
rou20-iso	2	6,35	4,11	0,08	369422	366901	363927
rou20-iso	3	42,07	19,90	2,45	364404	363372	362761

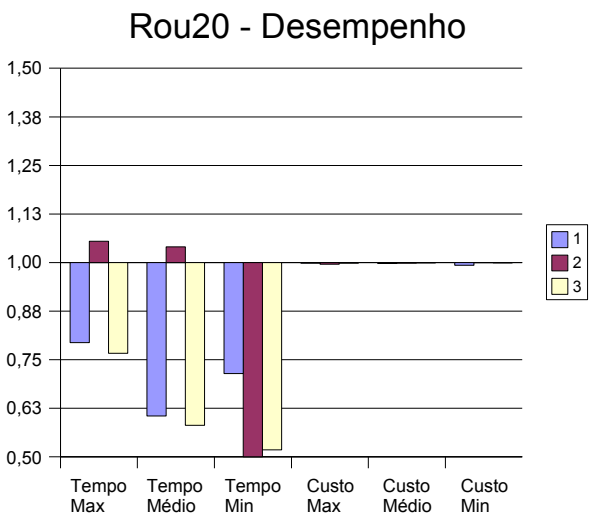


Figura 5.18: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Rou20

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
tai30a-prim	1	442,77	249,29	3,83	930323	927513	924192
tai30a-prim	2	4,68	3,03	0,47	949706	941454	929851
tai30a-prim	3	48,32	31,77	14,53	939680	932703	927017

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
tai30a-iso	1	4,48	2,30	0,98	948321	939543	929220
tai30a-iso	2	46,40	17,56	1,28	936454	934350	930191
tai30a-iso	3	373,23	247,89	43,47	931862	927002	919554

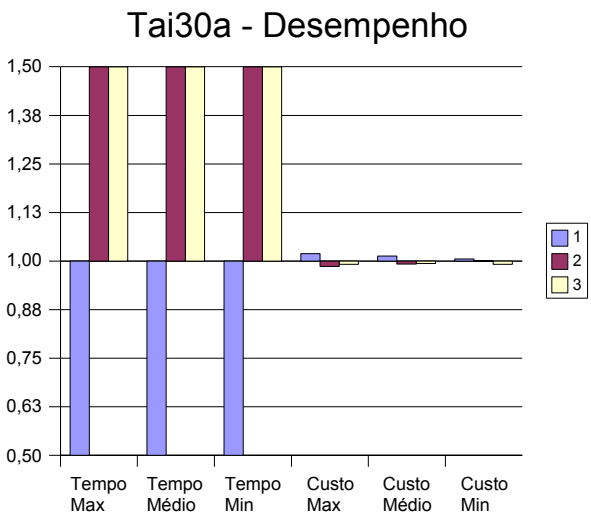


Figura 5.19: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Tai30a

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
tai60a-prim	1	175,18	103,86	18,62	3749034	3740754	3724426
tai60a-prim	2	1713,23	908,02	53,02	3739727	3728252	3717423
tai60a-prim	3	16091,97	11382,99	294,02	3720578	3709506	3693577

		Tempo Max	Tempo Médio	Tempo Min	Custo Max	Custo Médio	Custo Min
tai60a-iso	1	168,82	121,34	55,50	3761390	3740527	3714612
tai60a-iso	2	1602,75	894,50	159,57	3730146	3720219	3711318
tai60a-iso	3	17741,40	7881,76	1775,02	3719681	3709716	3697323

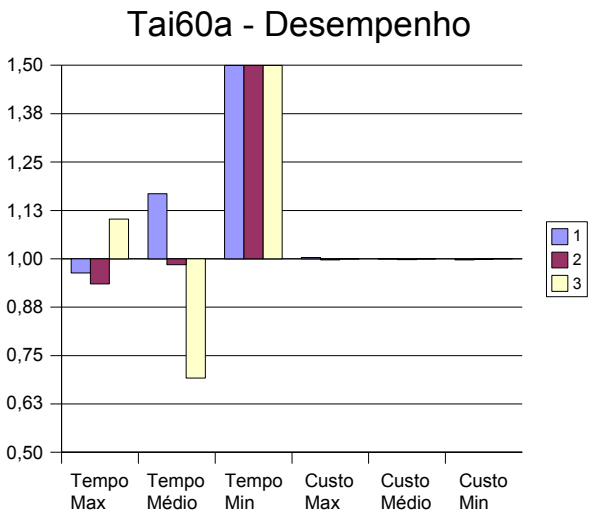


Figura 5.20: Comparação do desempenho da execução da instância Primitiva e Isomorfa de Tai60a

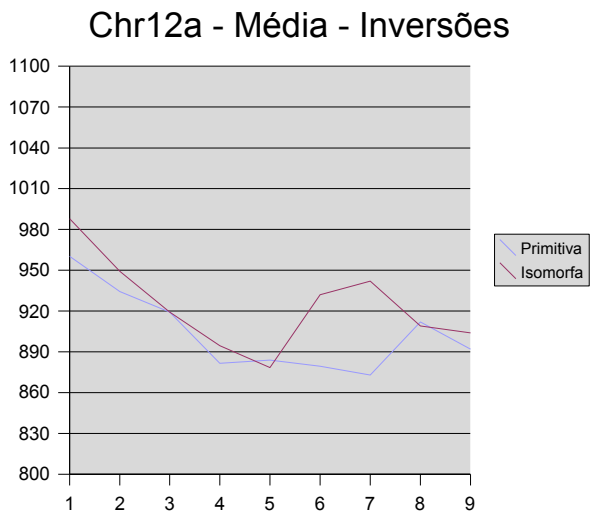


Figura 5.21: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Chr12a

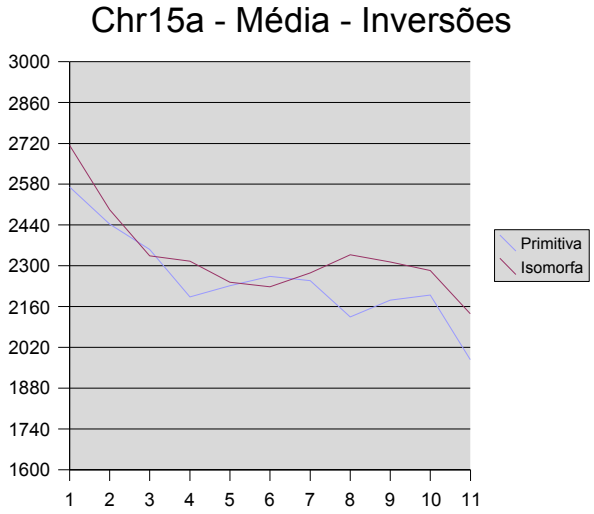


Figura 5.22: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Chr15a

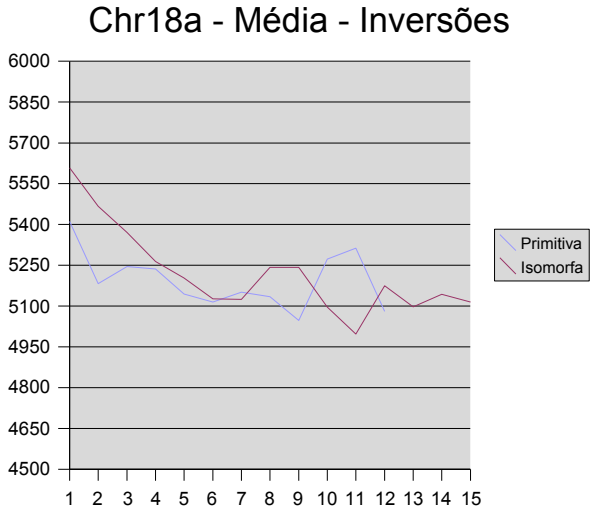


Figura 5.23: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Chr18a

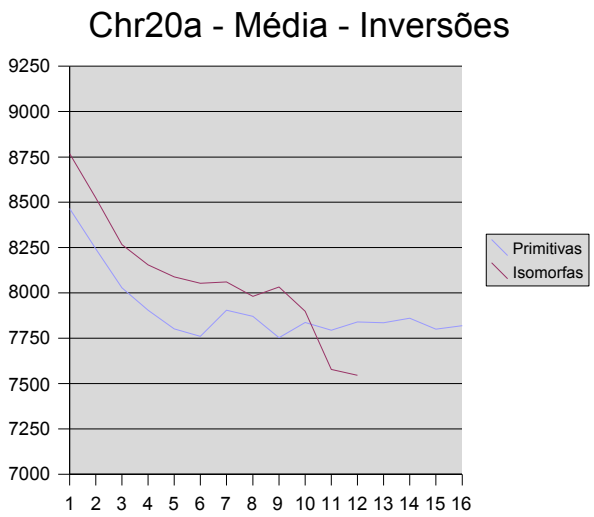


Figura 5.24: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Chr20a

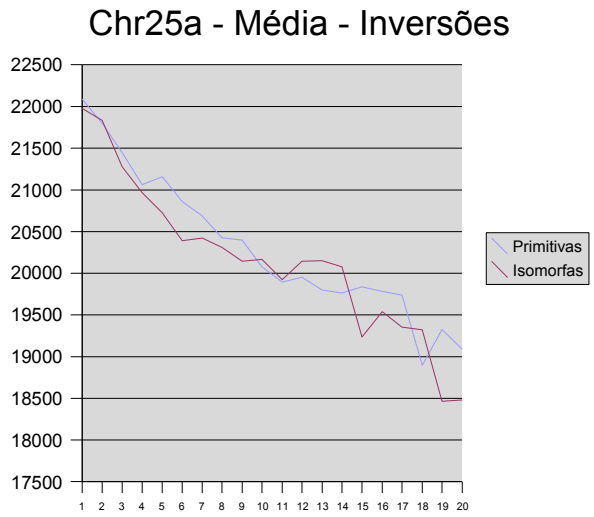


Figura 5.25: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Chr25a

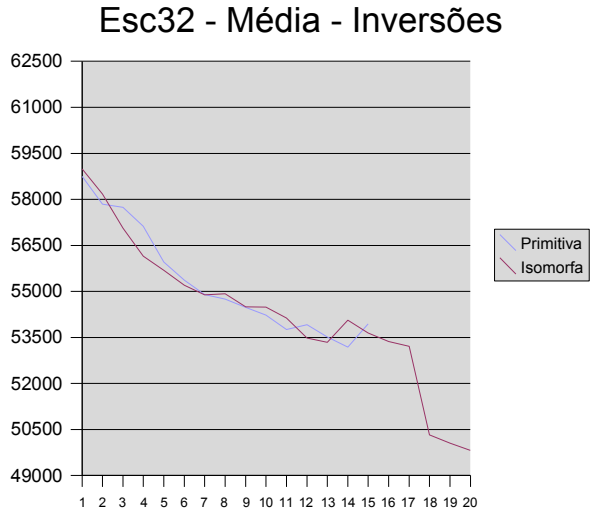


Figura 5.26: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Esc32

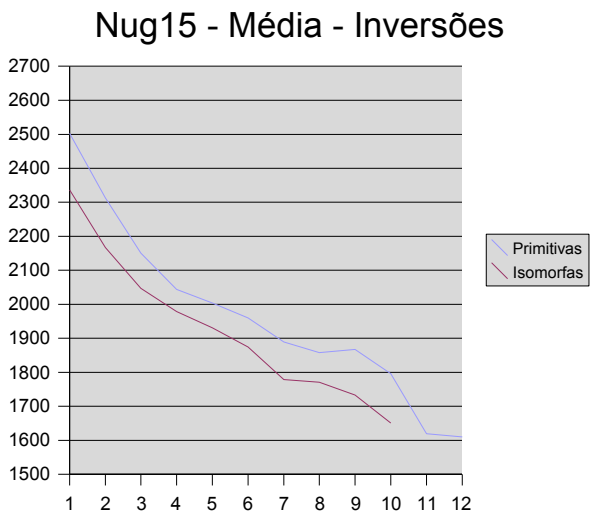


Figura 5.27: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Nug15

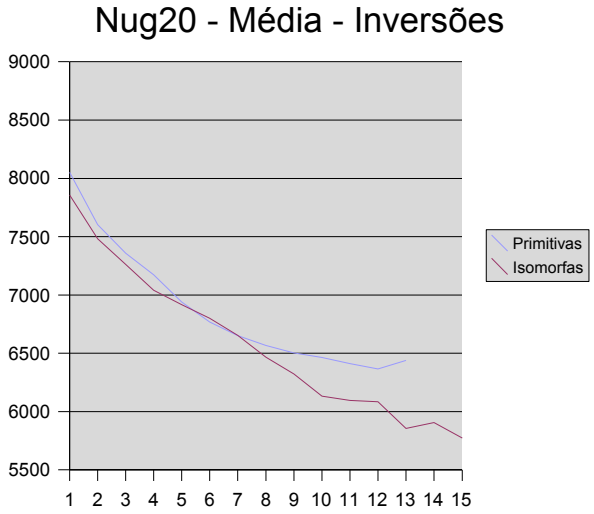


Figura 5.28: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Nug20

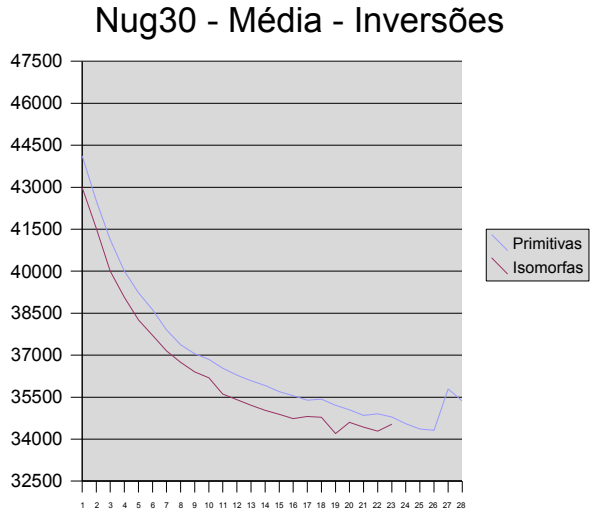


Figura 5.29: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Nug30

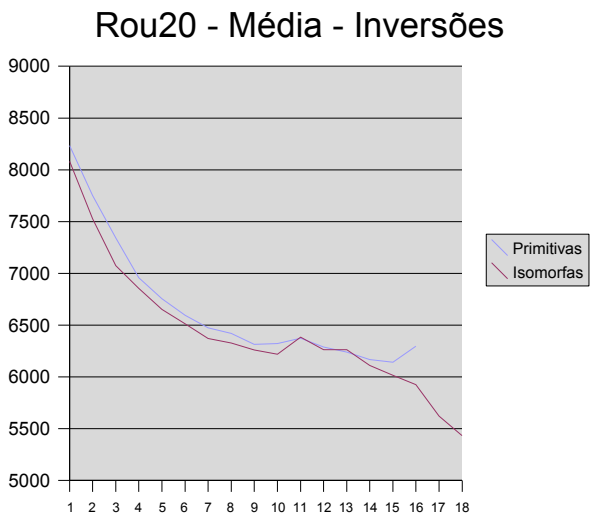


Figura 5.30: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Rou20

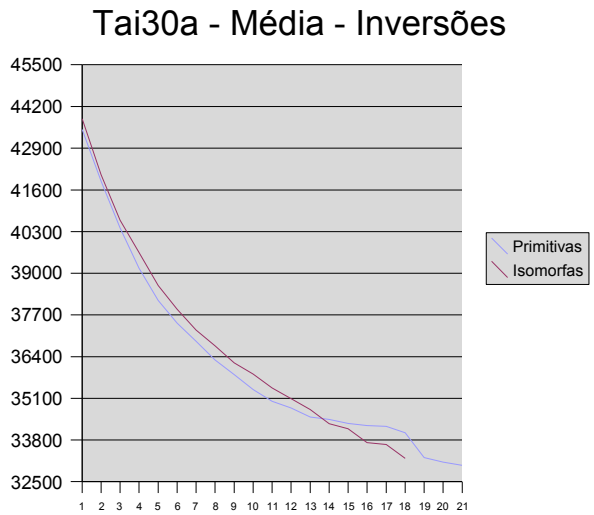


Figura 5.31: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Tai30a

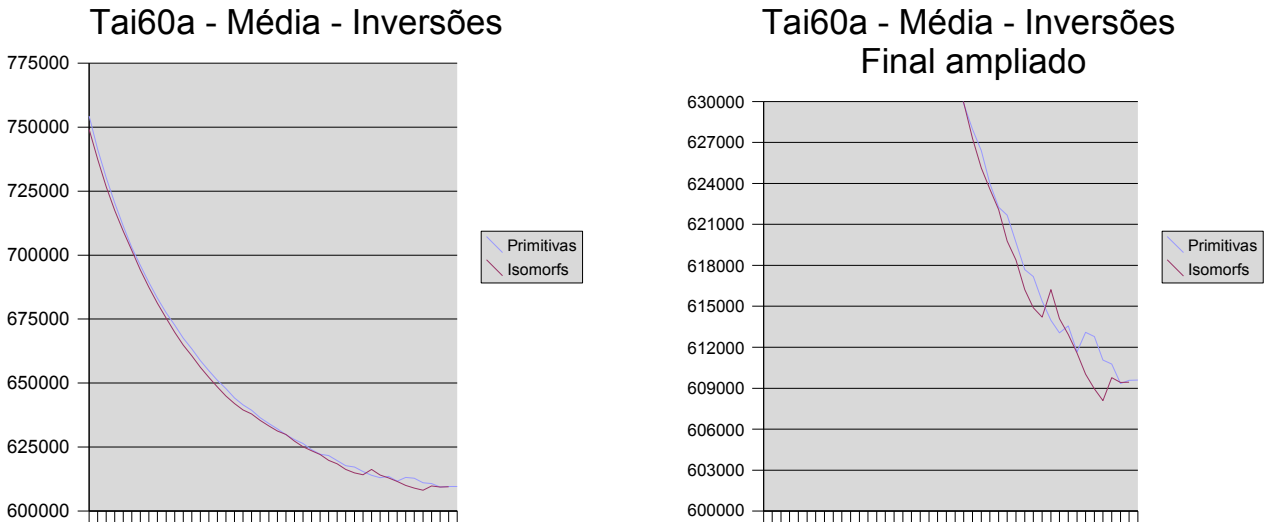


Figura 5.32: Comparação do valor médio das inversões das intâncias Primitivas e Isomorfas de Tai60a

Pode-se concluir pelos testes realizados que a estratégia de ordenação para resolução do PQA utilizando a metaheurística GRASP foi bem sucedida, pois a média geral de desempenho, considerando o somatório das médias de tempo e custo, é de 0,91, ou seja, está abaixo do valor 1, revelando que o desempenho da resolução das instâncias isomorfas foi melhor que o desempenho das instâncias primitivas. E o comportamento dos gráficos de inversões mostra uma curvatura mais suave para as instâncias isomorfas, e em alguns casos a curva das isomorfas está abaixo da curva das instâncias primitivas.

5.2.2 Busca Tabu

Nesta bateria de testes foram feitas as seguintes verificações:

- Desempenho na resolução das instâncias primitivas e isomorfas
- Comportamento do número de inversões durante a resolução das instâncias
- Comportamento da melhoria da soluçao durante a resolução das instâncias

O terceiro item irá mostrar quando foi encontrado um ponto de ótimo local de difícil atualização. Mais detalhes serão explicados adiante.

	Custo	Iteração
chr12a-prim	9552	755
chr12a-iso	9552	1143
chr15a-prim	9896	22026
chr15a-iso	9896	18286
chr18a-prim	11098	20948
chr18a-iso	11098	7306
chr20a-prim	2196	179628
chr20a-iso	2192	117066
chr25a-prim	3796	927111
chr25a-iso	3874	716835
esc16-prim	68	34
esc16-iso	68	13
esc32-prim	130	32020
esc32-iso	130	41779
nug12-prim	578	287
nug12-iso	578	118
nug15-prim	1150	589
nug15-iso	1150	2572
nug20-prim	2570	1290
nug20-iso	2570	82
nug30-prim	6124	3278
nug30-iso	6124	11138
rou20-prim	725522	21596
rou20-iso	725522	6098
tai30a-prim	1818146	104312
tai30a-iso	1818146	145117
tai60a-prim	7287044	498443
tai60a-iso	7292500	332641

Tabela 5.2: Comparação dos resultados obtidos entre as instâncias primitivs e isomorfas

A seguir é mostrado o gráfico comparativo de desempenho entre as instâncias primitivas e isomorfas:

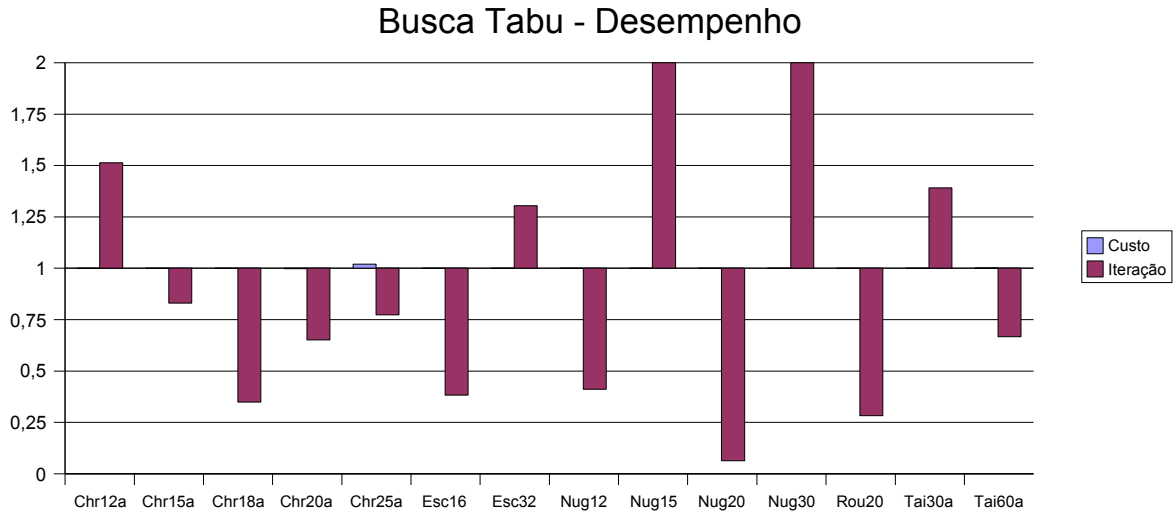


Figura 5.33: Gráfico comparativo do desempenho alcançado pelas instâncias primitivas e isomorfas

Os próximos gráficos irão mostrar o comportamento do número de inversões durante a resolução das instâncias e também o comportamento do melhoramento das instâncias, nos gráficos de iterações. O objetivo deste segundo gráfico é identificar se houve algum ponto onde a heurística ficou presa em um ótimo local.

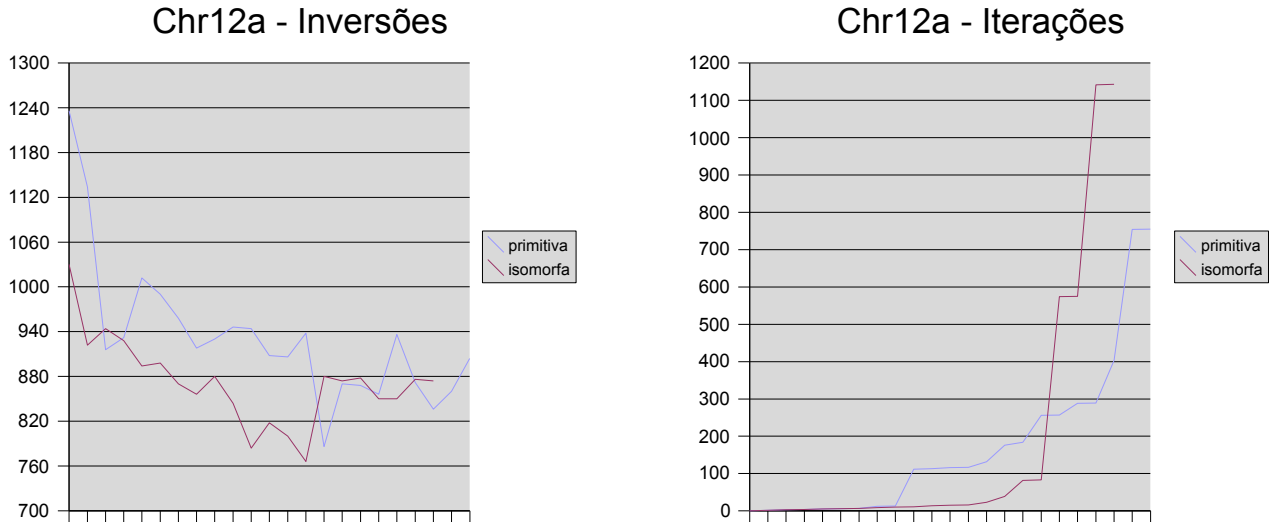


Figura 5.34: Gráfico comparativo para a instância Chr12a

Na Figura 5.34, no gráfico de Iterações, podemos ver que a linha vermelha, referente a instância isomorfa, teve 2 momentos em que houve uma mudança brusca na sua curvatura. Isto aconteceu porque perto da centésima iteração foi encontrado um ótimo local, e este ótimo só foi superado perto da iteração de número 600, onde foi encontrado novamente um ótimo local, superado apenas por volta da iteração 1150.

A seguir estão os resultados para as outras instâncias:

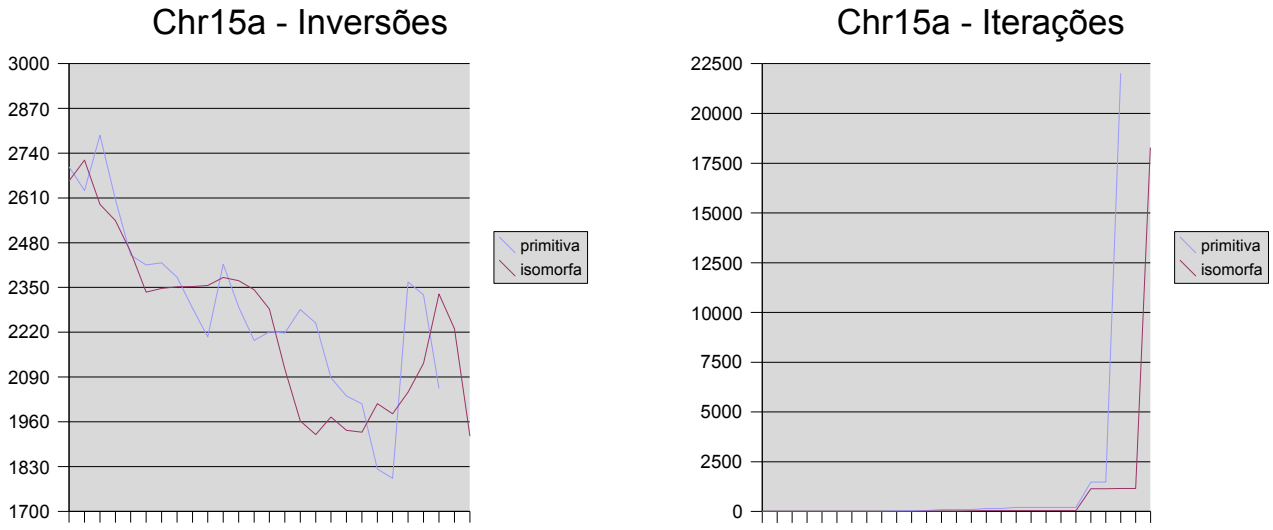


Figura 5.35: Gráfico comparativo para a instância Chr15a

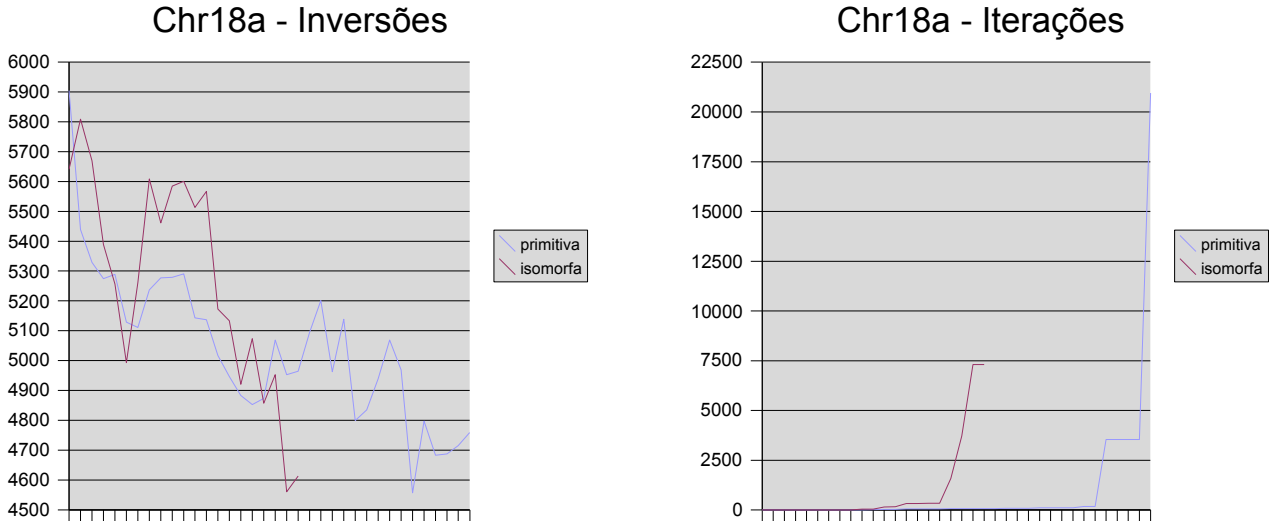


Figura 5.36: Gráfico comparativo para a instância Chr18a

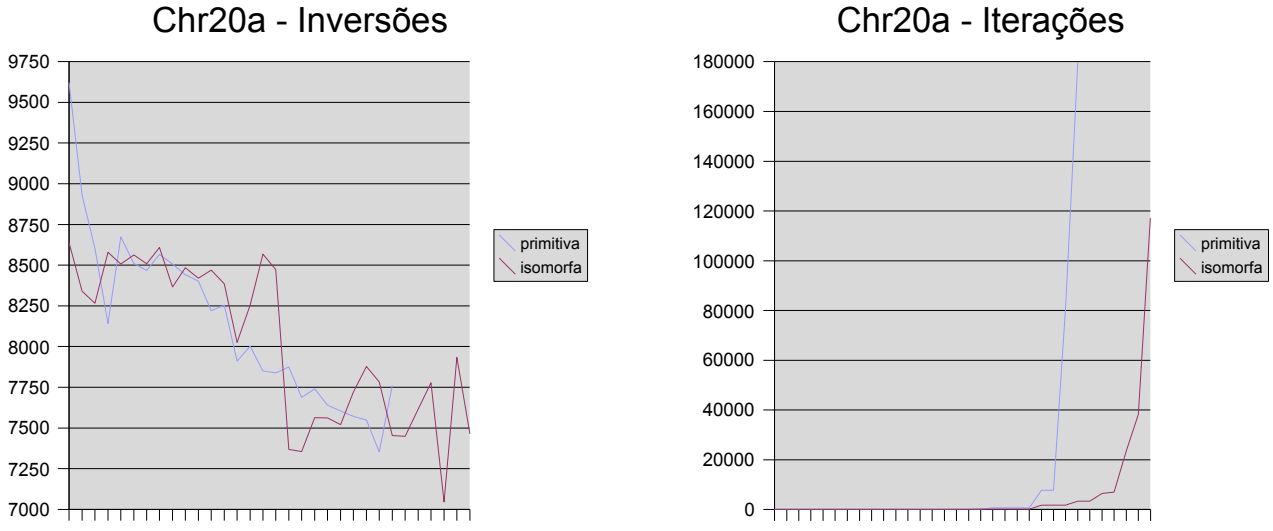


Figura 5.37: Gráfico comparativo para a instância Chr20a

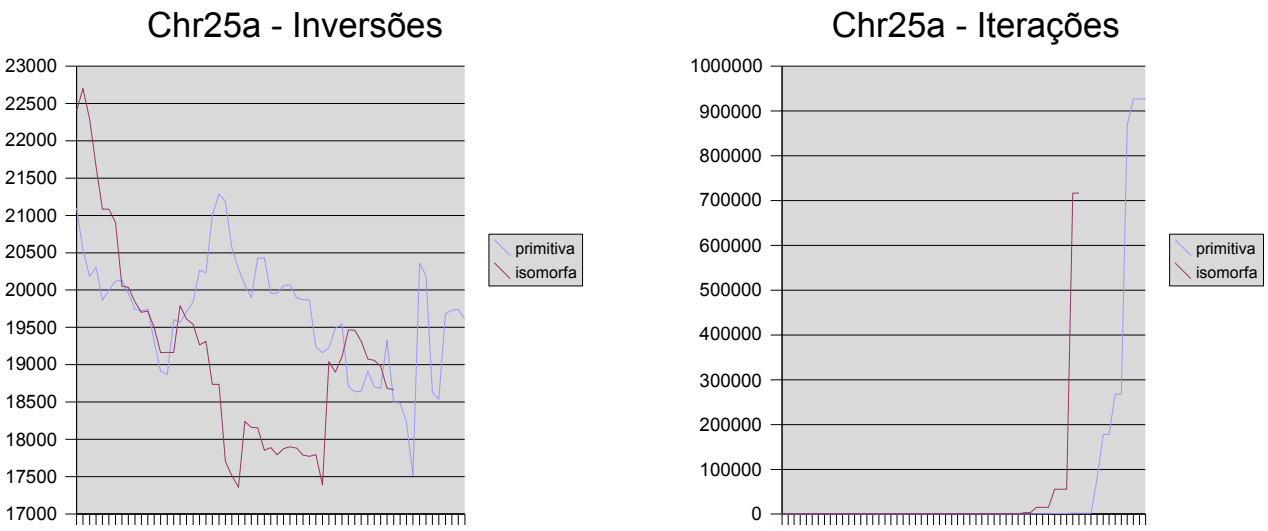


Figura 5.38: Gráfico comparativo para a instância Chr25a

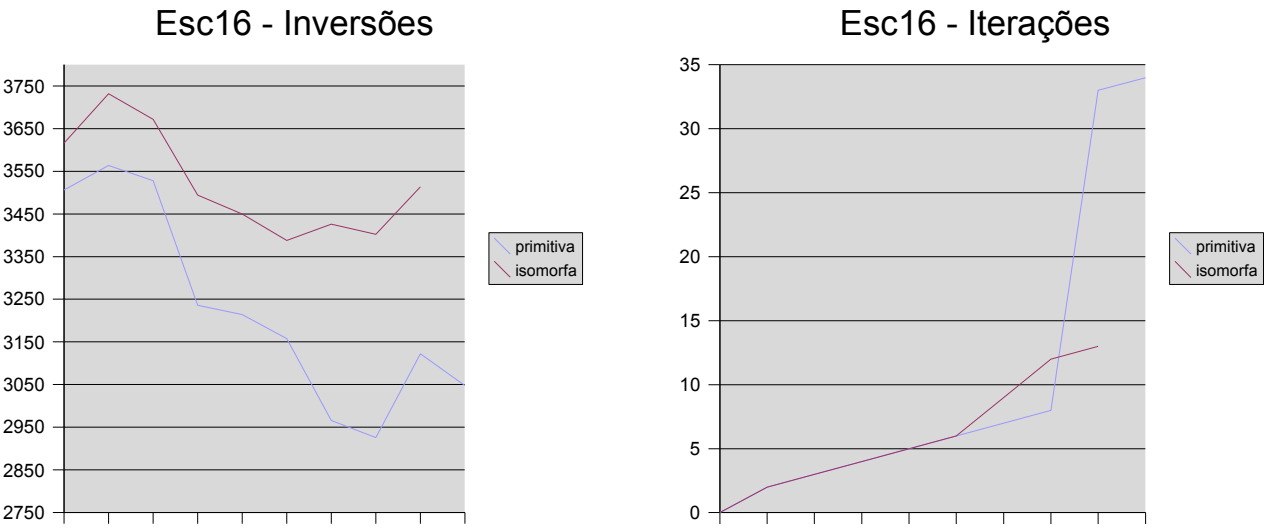


Figura 5.39: Gráfico comparativo para a instância Esc16

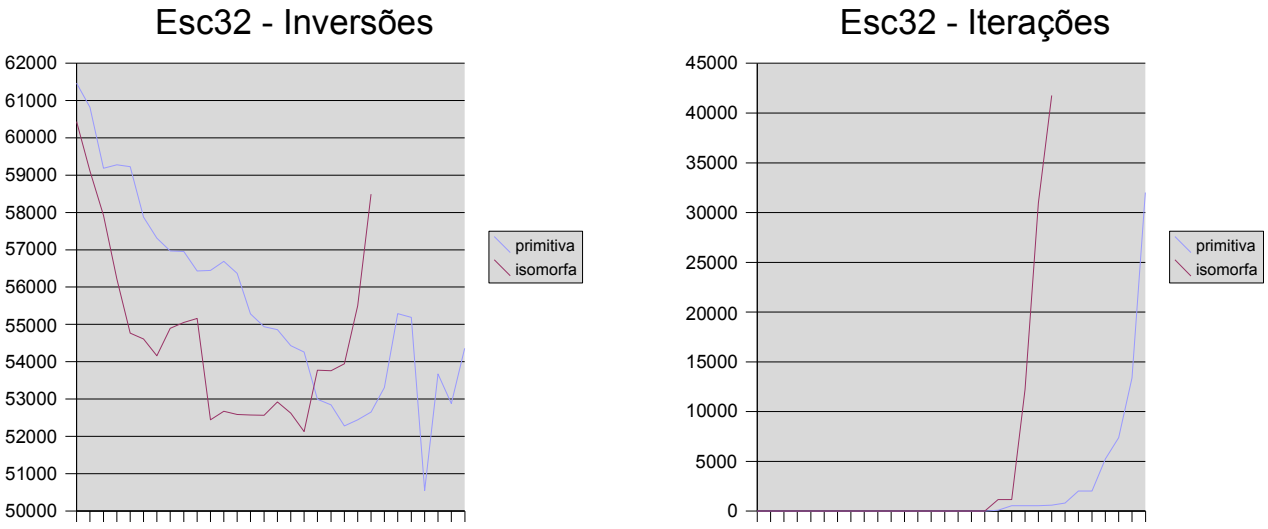


Figura 5.40: Gráfico comparativo para a instância Esc32

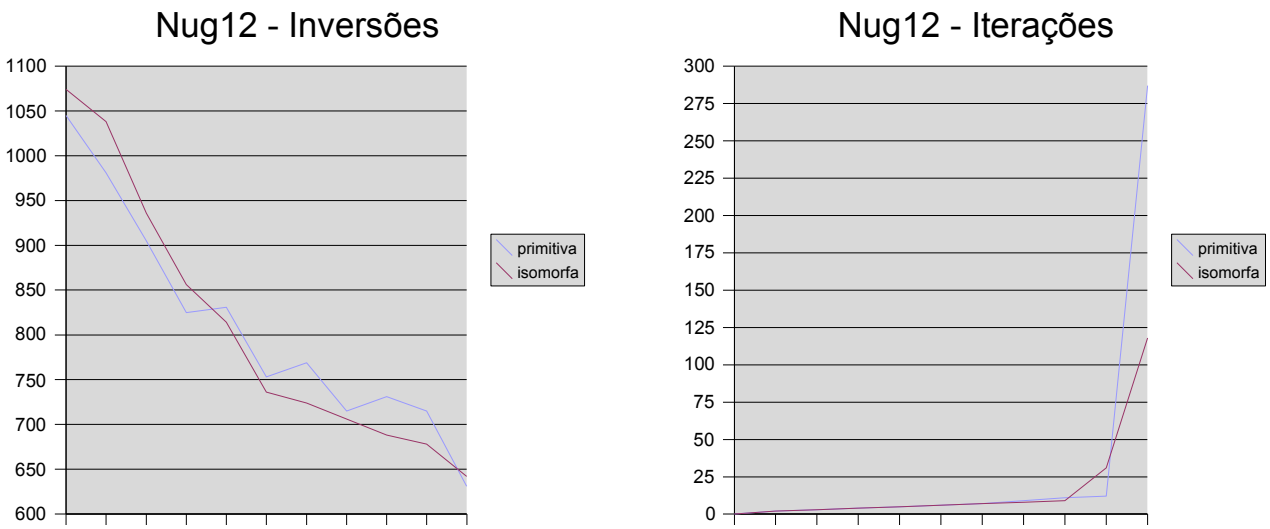


Figura 5.41: Gráfico comparativo para a instância Nug12

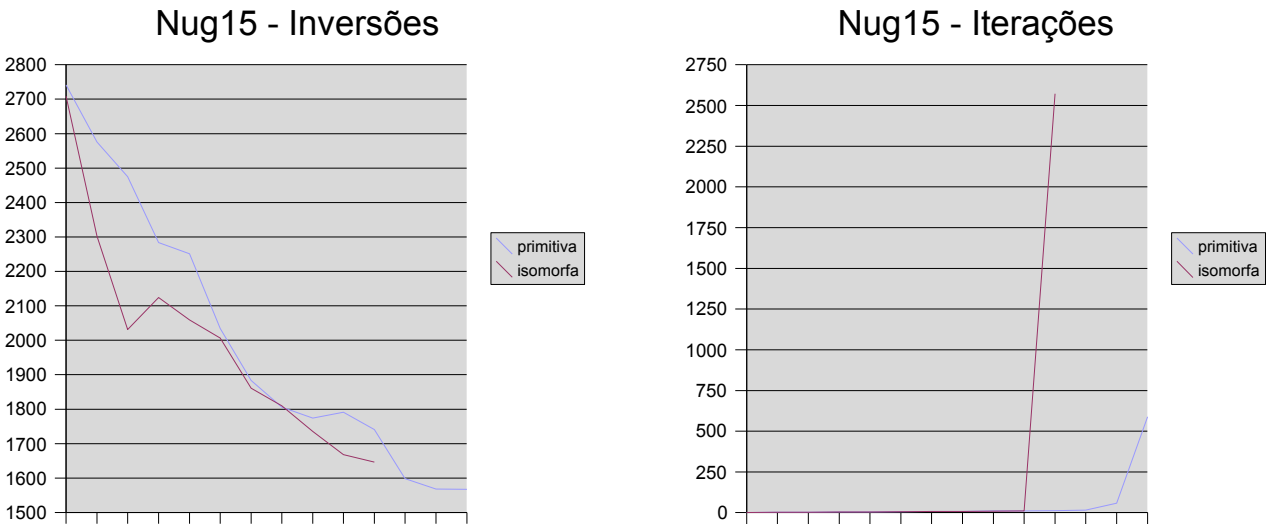


Figura 5.42: Gráfico comparativo para a instância Nug15

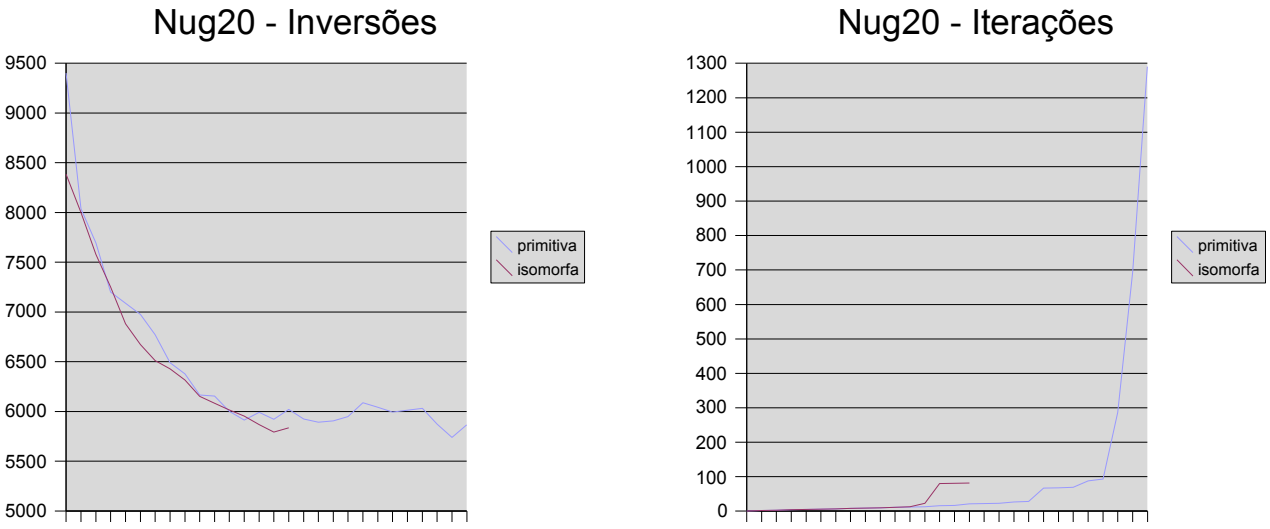


Figura 5.43: Gráfico comparativo para a instância Nug20

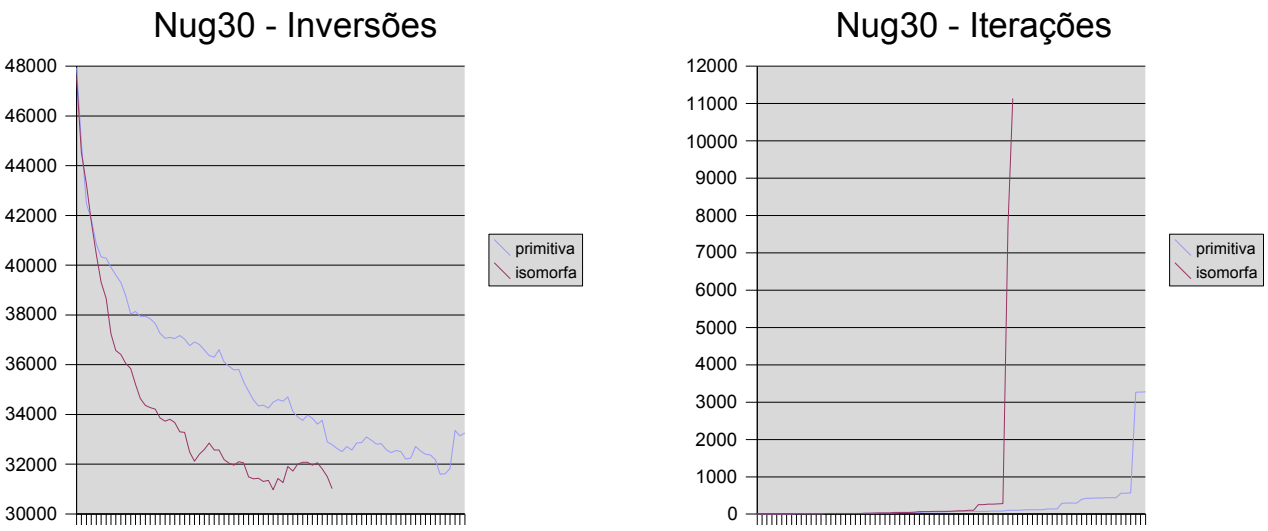


Figura 5.44: Gráfico comparativo para a instância Nug30

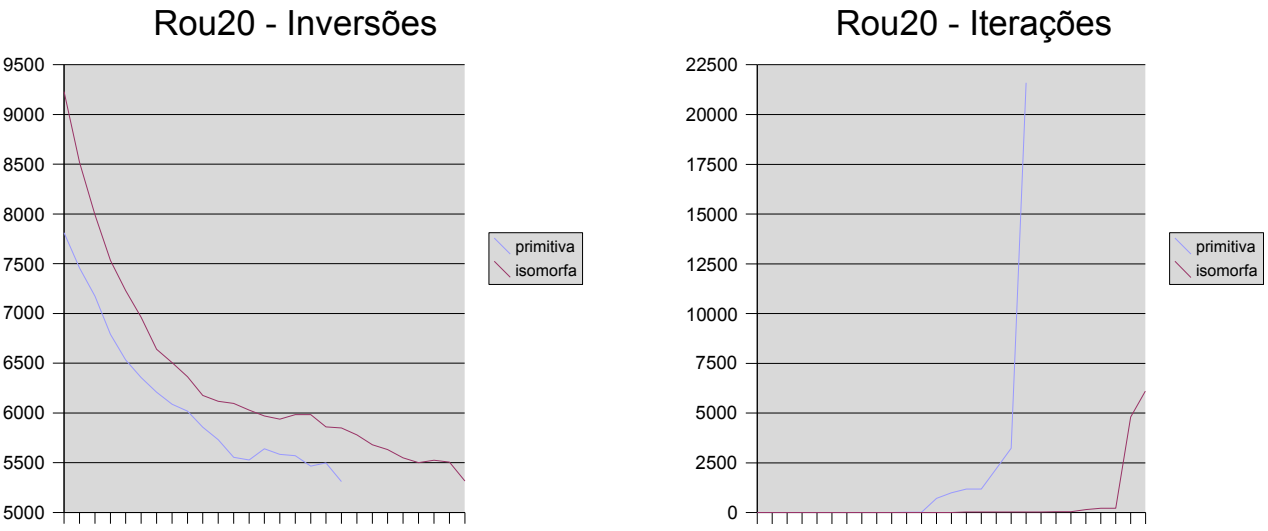


Figura 5.45: Gráfico comparativo para a instância Rou20

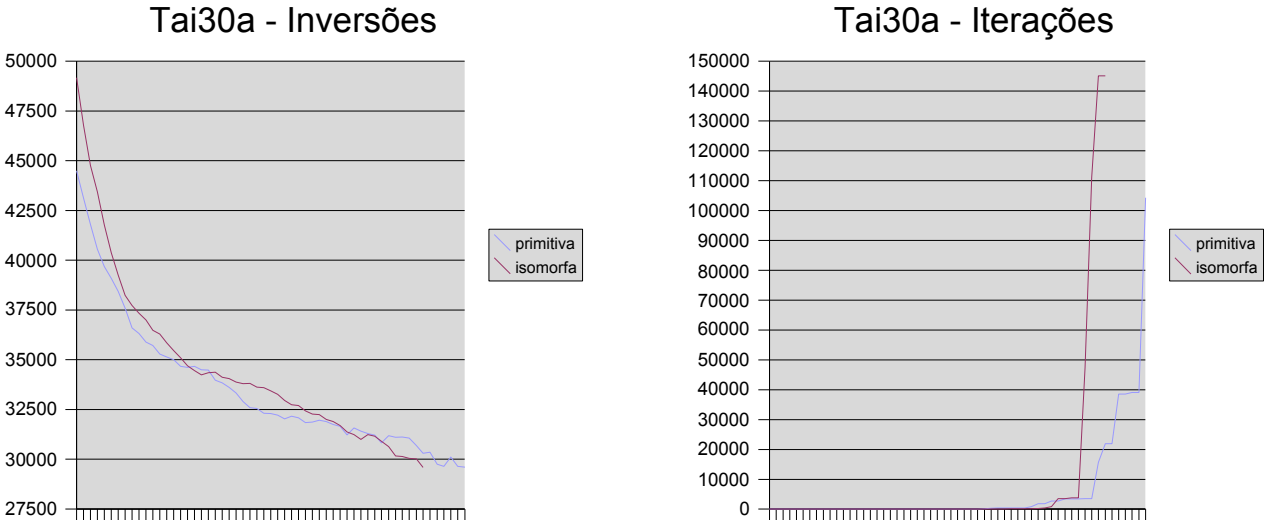


Figura 5.46: Gráfico comparativo para a instância Tai30a

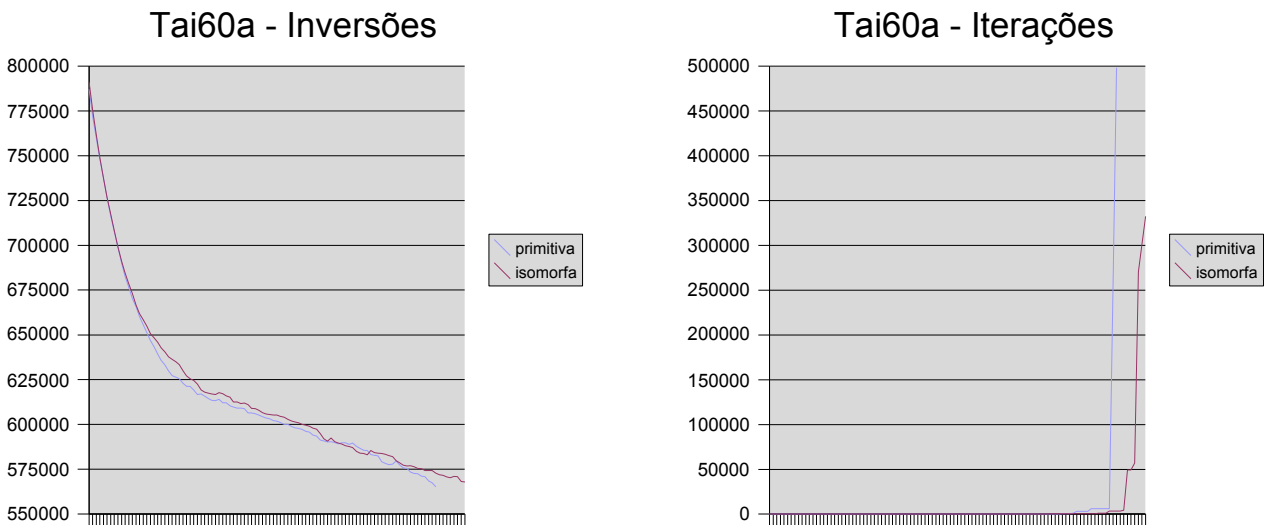


Figura 5.47: Gráfico comparativo para a instância Tai60a

A resolução das instâncias isomorfas foi melhor que as instâncias primitivas, alcançando o melhor resultado em menos tempo. Como pode ser visto nos gráficos de iterações, em determinados momentos as instâncias ficavam presas em ótimos locais, caso esses ótimos locais fossem contornados, o resultado do desempenho poderia ser melhor.

5.2.3 Colônia de Formigas

Os testes com a metaheurística Colônia de Formigas foram realizados da seguinte forma: foram escolhidos dois fatores para evaporação do ferômonio: 10 (baixa evaporação) e 50 (alta evaporação) e duas formas de execução: com limite de 1000 iteração e sem lilmite de iterações, onde o objetivo é alcançar o valor ótimo para o problema.

1º Teste: Fator de evaporação 10 com limite máximo de 1000 iterações.

2º Teste: Fator de evaporação 50 com limite máximo de 1000 iterações.

3º Teste: Fator de evaporação 10 sem limite de iterações.

4º Teste: Fator de evaporação 50 sem limite de iterações.

	10-1000		50-1000	
	Iterações	**Custo**	**Iterações**	**Custo**
chr12a-prim	94	9552	157	9552
chr12a-iso	433	9552	7	9552
chr15a-prim	225	9896	281	9936
chr15a-iso	170	9936	311	9896
chr18a-prim	220	11098	222	11142
chr18a-iso	313	11142	305	11608
chr20a-prim	590	2350	876	2336
chr20a-iso	785	2434	46	2516
chr25a-prim	601	4140	320	3978
chr25a-iso	52	4372	215	3796
esc16a-prim	2	68	4	68
esc16a-iso	2	68	2	68
esc32a-prim	498	130	765	130
esc32a-iso	72	134	391	130
nug12-prim	82	578	25	578
nug12-iso	313	578	23	578
nug15-prim	2	1150	4	1150
nug15-iso	92	1150	212	1150
nug20-prim	82	2570	310	2570
nug20-iso	64	2570	211	2570
nug30-prim	539	6148	83	6124
nug30-iso	175	6128	720	6128
rou20-prim	355	728512	323	729066
rou20-iso	457	728020	293	727920
tai30a-prim	797	1837966	666	1851354
tai30a-iso	125	1856292	33	1837290
tai60a-prim	736	7288750	338	7346450
tai60a-iso	582	7409746	415	7323058

Tabela 5.3: Execução da Meta-Heurística Colônia de Formigas com limite de iterações

	10-infinito		50-infinito	
	Iterações	**Custo**	**Iterações**	**Custo**
chr12a-prim	94	9552	157	9552
chr12a-iso	433	9552	7	9552
chr15a-prim	225	9896	16668	9896
chr15a-iso	7604	9896	311	9896
chr18a-prim	220	11098	8900	11098
chr18a-iso	106249	11098	2518	11098
chr20a-prim	132676	2192		
chr20a-iso	5755	2192		
chr25a-prim	22005	3796		
chr25a-iso	56553	3796		
esc16a-prim	2	68	4	68
esc16a-iso	2	68	2	68
esc32a-prim				
esc32a-iso				
nug12-prim	82	578	25	578
nug12-iso	313	578	23	578
nug15-prim	2	1150	4	1150
nug15-iso	92	1150	212	1150
nug20-prim	82	2570	310	2570
nug20-iso	64	2570	211	2570
nug30-prim				
nug30-iso				
rou20-prim	65742	725522	120091	725522
rou20-iso	105221	725522	72116	725522
tai30a-prim				
tai30a-iso				
tai60a-prim				
tai60a-iso				

Tabela 5.4: Execução da Meta-Heurística Colônia de Formigas sem limite de iterações

Na tabela 5.4 alguns campos aparecem sem valor pois o programa não terminou de executar, isto é, não alcançou o valor ótima para a instância.

Os gráficos comparativos do desempenho das instâncias primitivas e isomorfas são mostrados a seguir, novamente, valores maiores que 1 indicam que a resolução das instâncias isomorfas foi melhor, e menores que 1 inficam que a resolução das instâncias primitivas foi melhor:

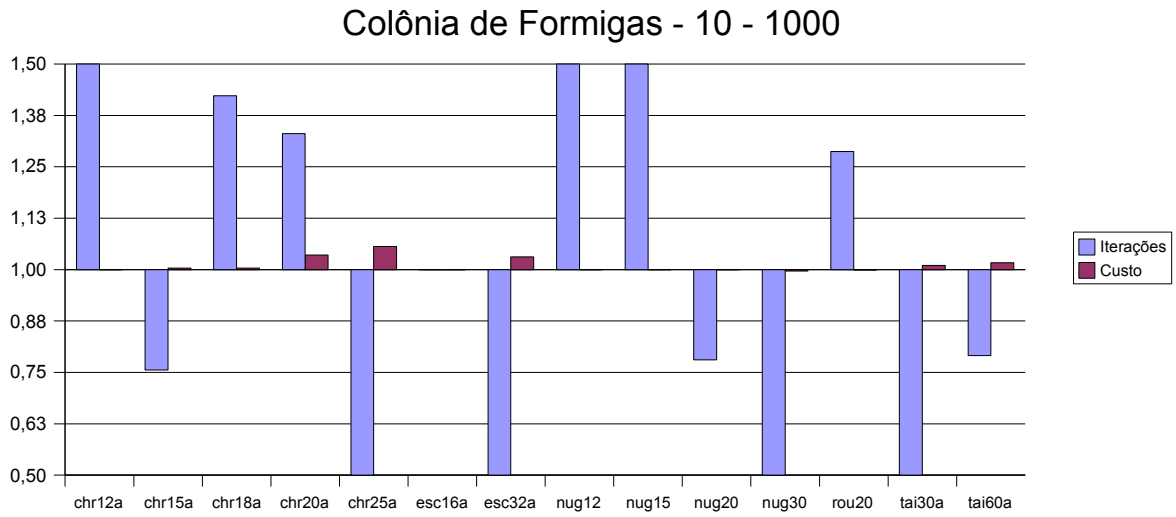


Figura 5.48: Primeiro teste de desempenho da Meta-Heurística Colônia de Formigas

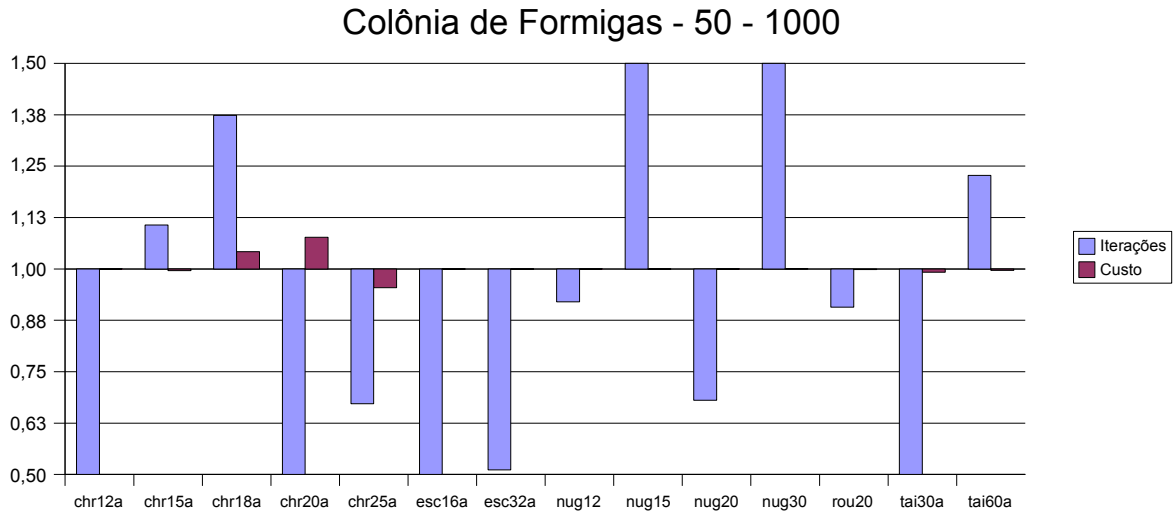


Figura 5.49: Segundo teste de desempenho da Meta-Heurística Colônia de Formigas

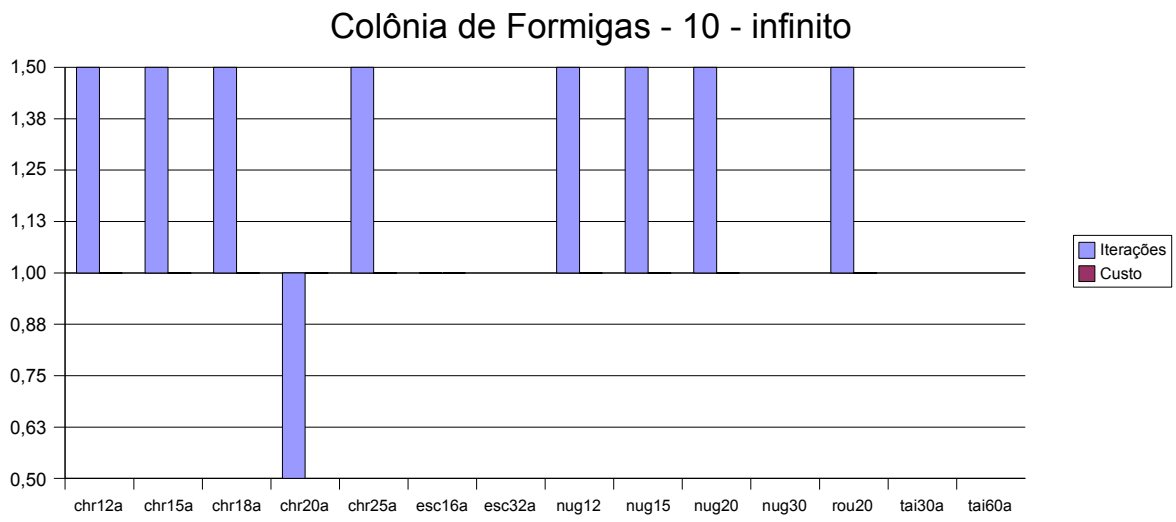


Figura 5.50: Terceiro teste de desempenho da Meta-Heurística Colônia de Formigas

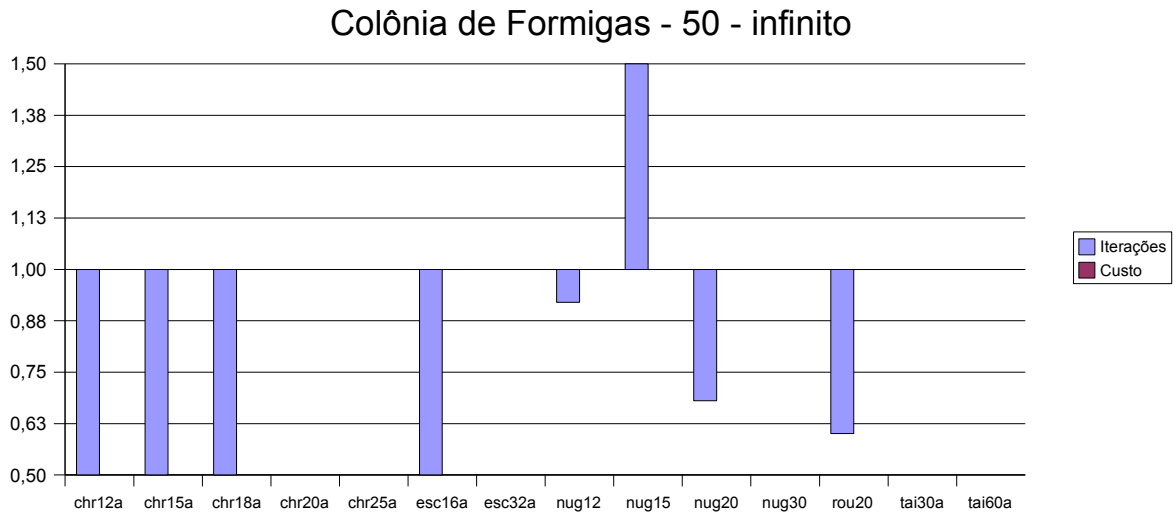


Figura 5.51: Quarto teste de desempenho da Meta-Heurística Colônia de Formigas

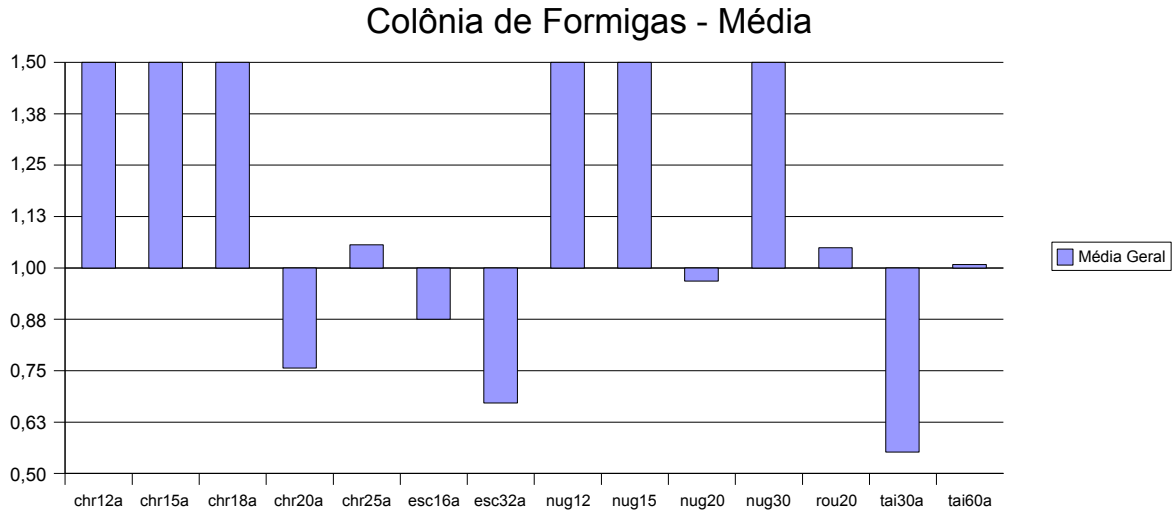


Figura 5.52: Média dos testes de desempenho da Meta-Heurística Colônia de Formigas

A seguir seguem alguns gráficos comparativos de inversões. Para esta bateria de testes não serão incluídos os gráficos para todas as instâncias, pois os resultados obtidos mostraram que os gráficos, tanto das instâncias primitivas quanto das instâncias isomorfas, não apresentaram uma curvatura suave.

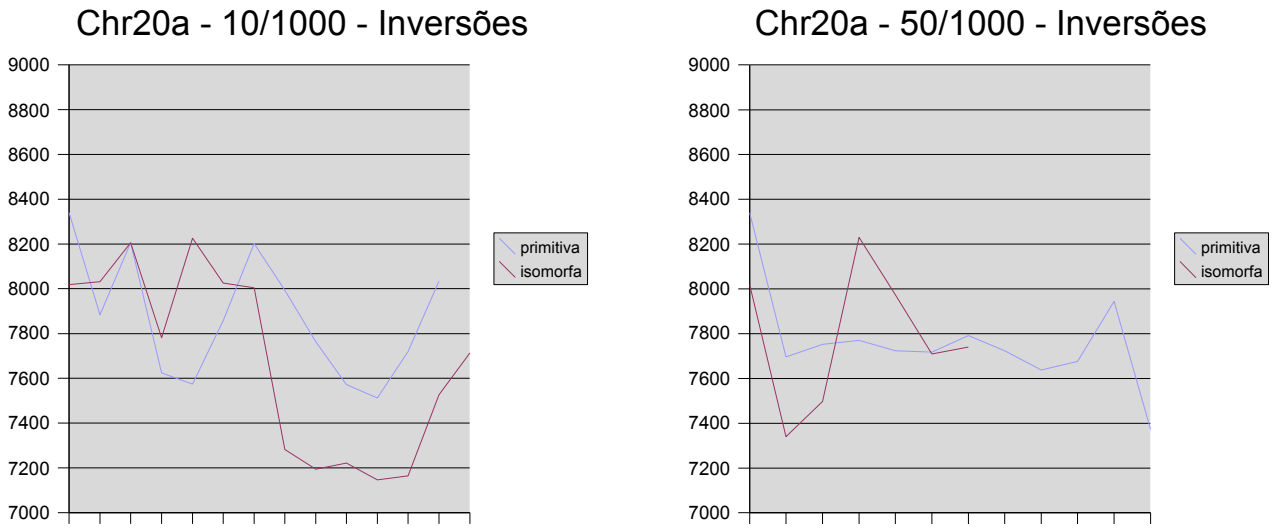


Figura 5.53: Gráfico comparativo do número de inversões para a instância Chr20a nos testes com limite de iterações

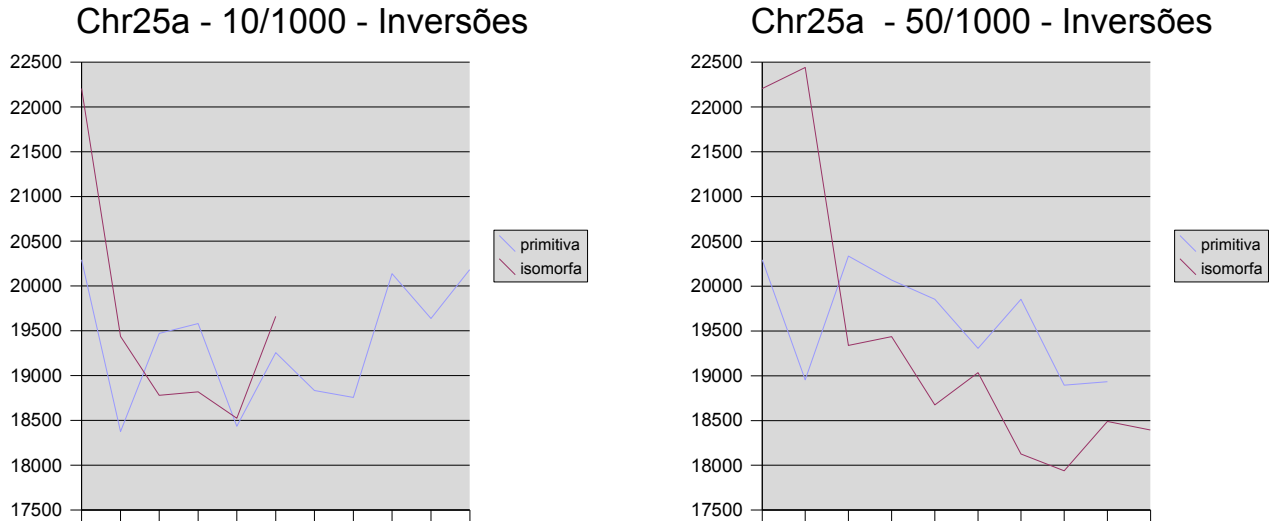


Figura 5.54: Gráfico comparativo do número de inversões para a instância Chr25a nos testes com limite de iterações

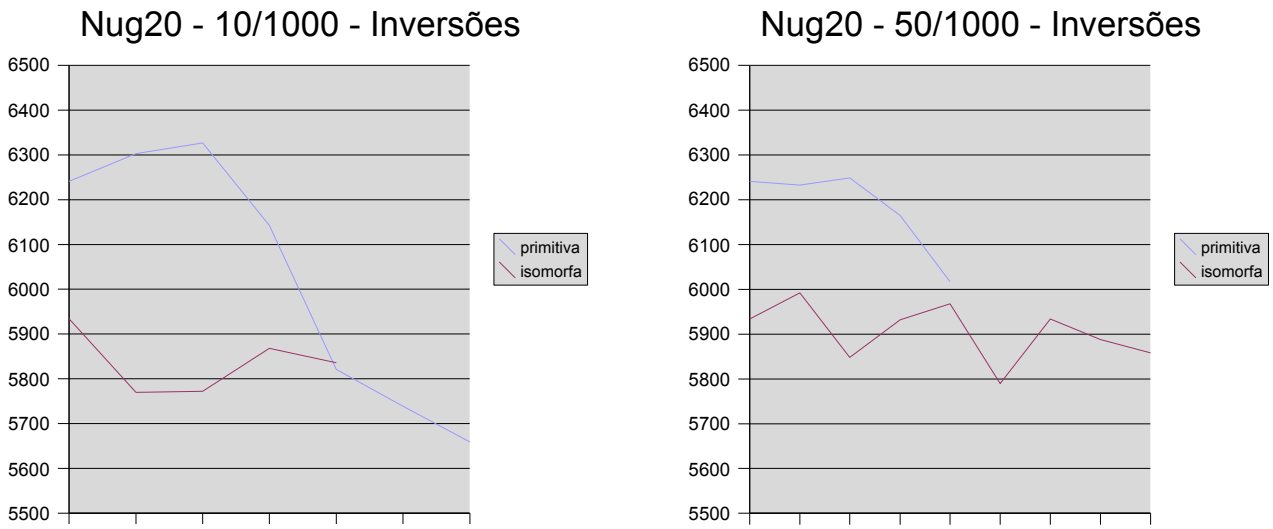


Figura 5.55: Gráfico comparativo do número de inversões para a instância Nug20 nos testes com limite de iterações

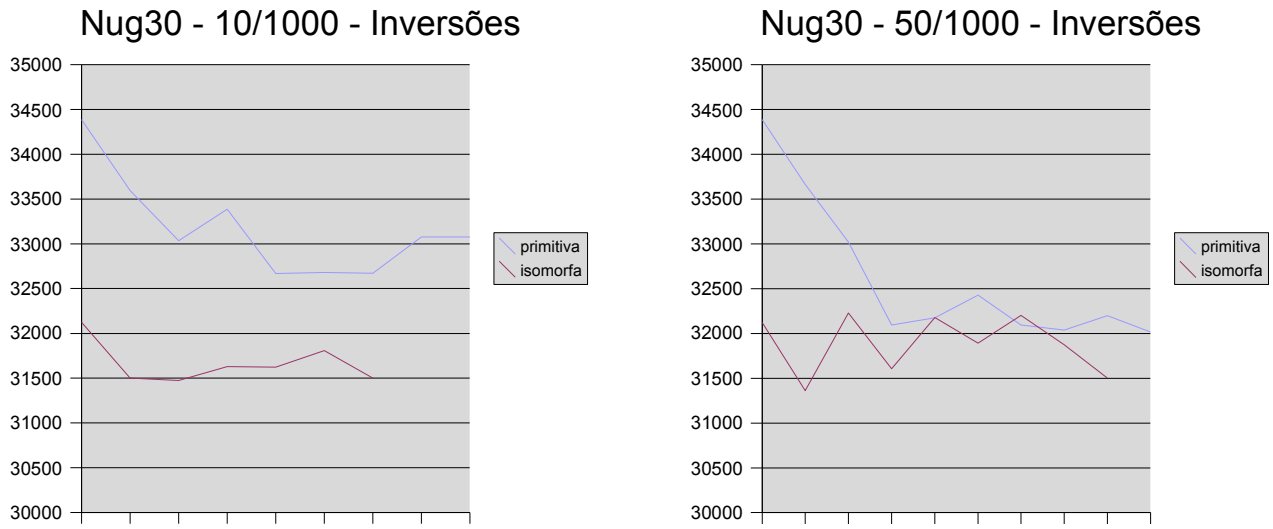


Figura 5.56: Gráfico comparativo do número de inversões para a instância Nug30 nos testes com limite de iterações

Para a Meta-Heurística Colônia de Formigas, a ordenação da instância do problema não se mostrou como uma boa estratégia. A média geral obtida foi de 7,46. Uma exceção a ser destacada ocorreu na segunda bateria de testes, onde o melhor desempenho foi obtido na maioria dos casos para as instâncias isomorfas. Os gráficos de inversões mostram uma curvatura irregular para as instâncias primitivas e isomorfas em todos os casos.

5.2.4 HeuristicHead

Este teste foi feito para realizar uma comparação da resolução do PQA em um programa que realiza um melhoramento nos dados de entrada. Os resultados são apresentados a seguir:

	Custo	Inversões	Tempo
chr12a-prim	9552	904	0,06
chr12a-iso	9916	758	0,06
chr15a-prim	10440	2091	0,27
chr15a-iso	9978	2230	0,30
chr18a-prim	13180	4981	1,00
chr18a-iso	12644	4817	0,98
chr20a-prim	2506	7484	2,02
chr20a-iso	2468	7500	1,95
chr25a-prim	4534	19836	10,38
chr25a-iso	4220	19596	10,83
esc16a-prim	68	2840	0,25
esc16a-iso	68	2918	0,17
esc32a-prim	134	52877	45,44
esc32a-iso	142	51796	41,02
nug12-prim	582	591	0,06
nug12-iso	578	642	0,06
nug15-prim	1152	1542	0,31
nug15-iso	1150	1663	0,28
nug20-prim	2570	5867	2,31
nug20-iso	2570	5662	2,31
nug30-prim	6156	32094	46,69
nug30-iso	6184	31796	44,88
rou20-prim	731348	5513	2,02
rou20-iso	733818	5597	2,13
tai30a-prim	1858226	31512	34,13
tai30a-iso	1859766	31685	33,14
tai60a-prim	7424510	588633	4552,88
tai60a-iso	7397918	586646	4566,11

Tabela 5.5: Resultados obtidos para a Heurística HeuristicHead

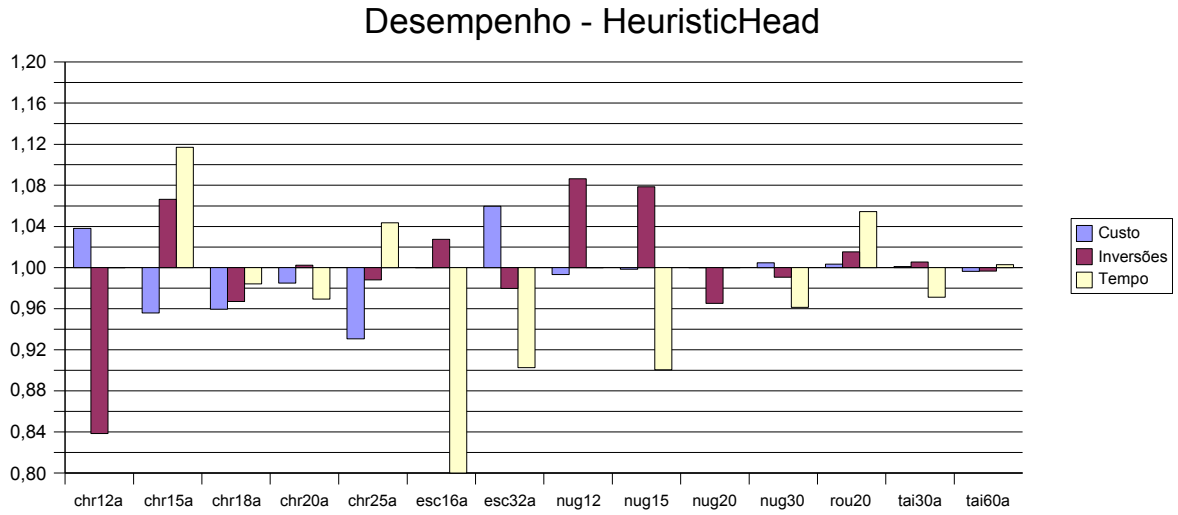


Figura 5.57: Gráfico comparativo do desempenho obtidos pelo HeuristicHead

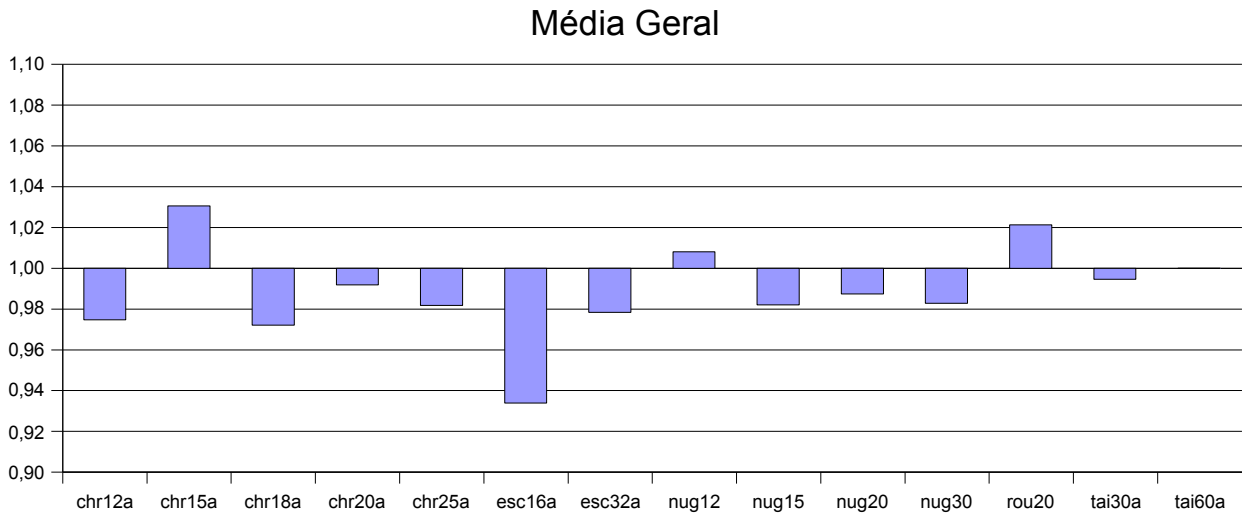


Figura 5.58: Gráfico comparativo da Média Geral obtida pelo HeuristicHead

Como pode ser visto pelo último gráfico, a ordenação das instâncias primitivas resultaram em um ganho de desempenho, mostrando-se uma boa estratégia para ser utilizada em conjunto com esta heurística.

Capítulo 6
Conclusão

Neste trabalho apresentamos uma proposta de resolução do Problema Quadrático de Alocação através da manipulação dos dados de entrada para obtenção de instâncias que fossem mais fáceis de resolver, ou que resultassem em um melhor custo final. Um dos pontos essenciais deste trabalho é derivado do trabalho de Abreu et.al [6], que define a Família de Instâncias cuja a Instância Padrão, PQA($\vec{F}^{-}$, $\vec{D}^{+}$) é resolvido em tempo polinomial. A partir disto foi desenvolvido um algoritmo de ordenação que tivesse como objetivo modificar a instância do problema transformando-a em uma instância com um menor número de inversões.

Um desafio encontrado foi desenvolver um algoritmo que alcançasse o objetivo desejado, e não descaracteriza-se o problema, isto é, a instância do problema seria alterada, mas o problema continuaria o mesmo. Isto foi alcançado através do desenvolvimento do conceito de Isomorfismo Perfeito, que define as seguintes regras: deve ser mantida a quantidade de vértices e arestas, a preservação das arestas, e o agrupamento de arestas em um determinado nó. Apenas com o Isomorfismo Perfeito é possível modificar uma instância do PQA para uma outra instância sem haver a descaracterização do problema.

Após o desenvolvimento do programa de ordenação das matrizes, foi feita uma bateria de testes com as metaheurísticas GRASP, Busca Tabu, Colônia de Formigas e HeuristicHead para se realizar uma comparação do desempenho na resolução do PQA com instâncias sem ordenação e com ordenação, chamadas de primitivas e isomorfas respectivamente. Os resultados obtidos mostraram que a estratégia adotada apresentou bons resultados para as metaheurísticas GRASP, Busca Tabu, HeuristicHead, onde se conseguiu em vários casos tempos de execução melhor, e solução melhor. Isto pode ter ocorrido devido a características que estas metaheurísticas possuem, principalmente na etapa de busca local.

Trabalhos futuros podem ser realizados no algoritmo de ordenação da instância do PQA, visando aumentar ainda mais a ordenação dos elementos. Também podem ser desenvolvidas metaheurísticas que já estejam preparadas para trabalhar com as instâncias ordenadas, com o objetivo de contornar problemas problemas de cair em ótimos locais, dentre outros.

Bibliografia

[1] - Abreu, N. M. M., Querido, T. M., Gouvea, E. F., Boaventura Netto, P. O. "Classes of Quadratic Assignment Problem instances: isomophirsm and difficulty measure using a statistical approach", Discrete Applied Mathematics, vol.124, pp. 103-116, 2002.

[2] - Akkeleş, A., Ünveren, A., Acan, A., “A Fresh View: Quadratic Assignment Problem”, 5th GRACM International Congress on Computational Mechanics, 29 June – 1 July, 2005

[3] – Anjos, M. F., Vanneli, A., "A New Mathematical Programming Framework for Facility Layout", Zentrum für Angewandte Informatik Köln, 2002.

[4] – Borgatti, S. P., "A Statistical Method for Comparing Aggregate Data Across A Priori Groups", Field Methods, Vol. 14, No. 1, pp. 88–107, February 2002.

[5] – Brüngger, A., Marzetta, A., Clausen, J., Perregaard, M., "Solving Large-Scale QAP Problems in Parallel with the Search Library ZRAM", 1998.

[6] – Burkard, R.E., Çela, E., Demidenko, V.M., Metelski, N.N., Woeginger G.J., “Perspectives of easy and hard cases of the quadratic assignment problem”, Optimierung und Kontrolle, 1997.

[7] - Burkard, R.E., Karisch, S.E., Rendl, F., “QAPLIB A Quadratic Assignment Problem Library”, European Journal of Operational Research, v55, pp115-119, 1991.

[8] - Çela, E. “The Quadratic Assignment Problem - Theory and Algorithms”, Kluwer Academic Publisher, 1998.

[9] – Çela, E., "The Quadratic Assignment Problem - Special Cases and Relatives", Technische Universitãt Graz, Institut für Mathematik, 1995.

[10] - Çela, E., Burkard, R.E., Pardalos, P.M., Pitsoulis, L.S. “The Quadratic Assignment Problem”, Universitãt Graz, Institut für Mathematik, 1998.

[11] – Commander, C.W.: "A Survey of the Quadratic Assignment Problem, with Applications", The University of Florida, 2003.

[12] - Costa, F.P. “Programação de Horários em Escolas via GRASP e Busca Tabu”, Monografia de graduação em Engenharia de Produção – UNIVERSIDADE FEDERAL DE OURO PRETO, 2003.

[13] – Dorigo,M., Sttzle, T., “The ant colony optimization metaheuristic: Algorithms, Applications and Advances”, To appear in F. Glover and G. Kochenberger, eds., Handbook of Metaheuristics. Kluwer Academic Publishers, 2002.

[14] - Dorigo, M. “Ant Algorithms Solve Difficult Optimization Problems”, Advances in Artificial Life, Proceedings of the Sixth European Conference on Artificial Life, LNAI 2159, Springer-Verlag, pp. 11–22.

[15] – Feo, T., Resende, M.G.C., Smith, S., “A Greedy Randomized Adaptive Search Procedure for the Maximum Independent Set”, Journal of Operations Research, pp. 860-878, 1994.

[16] – Gambardella, L., Taillard, E., Dorigo, M., “Ant colonies for the quadratic assignment problem”, Journal of the Operational Research Society, pp 167-176, 1999.

[17] - Geoffrion, A.M., G.W. Graves “Scheduling Parallel Production Lines with Changeover Costs” - Practical Application of a Quadratic Assignment (LP) Approach”, Operations Research, vol 24, n. 4, pp. 595-610, 1976.

[18] – Glover, F., Laguna, M., “Tabu Search”, Kluwer Academic Publishers, 1997.

[19] - Glover, F, Laguna, M., “Tabu Search”, http://www.upt.pt/tabusearch, 2005.

[20] – Hahn, P. M., "A Hospital Facility Layout Problem Finally Solved", University of Pennsylvania, 2000.

[21] – Hahn, M.P., Hightower, W.L., Jhonson, T.A., Spielberg, M.G., Roucariol, C., “A Lower Bound for the Quadratic Assignment Problem Based on a Level-2 Reformulation - Linearization Technique”, University of Pensilvania, 2002.

[22] - Hahn, P.M., and Grant, T.L, "Lower Bounds for the Quadratic Assignment Problem Based Upon a Dual Formulation", Operations Research, Vol. 46, No. 6, 1998.

[23] – Hadley,S.W., Rendl,F., Wolkowicz, H., “Bounds for the Quadratic Assignment Problem Using Continuos Optimization Techniques”.

[24] – Hadley,S.W., Rendl,F., Wolkowicz, H., “A new lower bound via projection for the quadratic assignment problem”, Math. of Operations Research, vol 17, pp.727-739, 1992.

[25] – Helmberg, C., “An Interior Point Method for Semidefinite Programming and Max-Cut Bound”, Dissertation zur Erlangung des Titels eines Doktors der techischen Wissenschaften eingereicht an der Tecnisch-Naturwissenchaftlichen Fakultat der Technischen Universitat Graz, 1994.

[26] - Hertz,A., Taillard, É. D., Werra, D., "Tabu Search", Local search in combinatorial optimization, pp. 121-136, 1997.

[27] – Hertz A, Taillard E., Werra D., “A Tutorial on Tabu Search”. http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf.

[28] – Julien, S.L., Taskar, B., Klein, D., “Word Alignment via Quadratic Assignment”. To appear in HLT-NAACL 06, 2006.

[29] - Kaufman, D.E., Nonis, J., Smith, R.L., “A Mixed Integer Linear Programming Model for Dynamic Route Guidance”, University of Michigan, 1997.

[30] – Koopmans, T.C, Beckmann, M., “Assignment Problems and the Location of Economic Activities”, Econometrica, vol. 25, n. 1, 1957.

[31] - Li, Y., Pardalos, P.M., Resende, M.G.C., “A greedy randomized adaptive search procedure for the quadratic assignment problem”, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol 16, pp. 237-261, 1994.

[32] - Li, Y., Pardalos, P. M., Ramarkrishnan. K. G., Resende, M. G. C.: "Lower bounds for the quadratic assignment problem", Annal of Operations Research 50, pp. 387-410, 1994.

[33] - Loiola, E. M., Abreu, N. M. M., Netto, P. B.: "Uma Revisão Comentada das Abordagens do Problema Quadrático de Alocação", Pesquisa Operacional, vol. 24, n. 1, pp. 73-109, 2004.

[34] – Maniezzo, V., Colorni, A., "The Ant System Applied to the Quadratic Assignment Problem", IEEE - Knowledge and Data Engineering 11, pp 769-778.

[35] – Miagkikh, V.V., Punch, W.F., "An Approach to Solving Combinatorial Optimization Problems Using a Population of Reinforcement Learning Agents", Michigan State University.

[36] - Mittelmann, H., Peng,J., Wu, X., "An Integer Linear Program Approach to Quadratic Assignment Problem", Arizona State University, 2006.

[37] – Pardalos P.M., Migdalas,A., Burkard, R., "Tight QAP bounds via linear programming" Combinatorial and Global Optimization, pp. 297-303, 2002.

[38] - Pardalos, P.M., Rendl, F., Wolkowicz, H., "The Quadratic Assignment Problem: A Survey of Recent Developments", Quadratic Assignment and Related Problems, volume 16, pp. 1-42. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1994.

[39] – Pardalos, P.M., Pitsoulis, L., Mavridou, T., Resende, M.G.C, "Parallel search for combinatorial optimization - Genetic algorithms, simulated annealing, tabu search and GRASP", Workshop on Parallel Algorithms for Irregularly Structured Problems Lyon, France, September 4-6, 1995.

[40] - Ramarkrishnan, K.G., Pardalos, P.M., Resende, M.G.C., "A branch and bound algorithm for the quadratic assignment problem using a lower bound based on linear programming", Proceedings of the state of the art in Global Optimization: Computational Methods and Applications, 1995.

[41] – Rangel, M.C., Abreu, N.M.M., Boaventura-Netto, P.O., "GRASP para o PQA – Um Limite de Aceitação para Soluções Iniciais", Pesquisa Operacional, vol. 20, No.1, junho de 2000.

[42] – Rangel M.C. Contribuições Algébricas ao Problema Quadrático de Alocação. Tese de doutorado, Programa de Engenharia de Produção – COPPE/UFRJ, 2000.

[43] – Rangel, M.C., Abreu, N.M.M., “Ordenações Parciais nos Conjuntos das Soluções dos Problemas de Alocação Linear e Quadrático”, Pesquisa Operacional, vol.23, n.2, pp.265-284, 2003.

[44] - Resende, M.G.C., P.M., Pitsoulis, L.S., “Greedy randomized adaptive search procedures”, ATT Labs Research, 2001.

[45] - Resende, M.G.C., RIBEIRO, C.C., “Greedy Randomized Adaptative Search Procedures”, in Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, pp. 219-249, 2003.

[46] - Resendo, L.C., Rangel, M.C., “Um algoritmo construtivo baseado em uma abordagem algébrica do problema quadrático de alocação”. Dissertação de Mestrado, PPGI/UFES, 2004.

[47] – Rosen, J.B., “A quadratic assignment formulation of the molecular conformation problem”, Journal of Global Optimization, vol 4, pp. 229-241, 1994.

[48] – Sahni, S., Gonzalez, T., “P-Complete Approximation Problems”, University of Minesota, Journal ol the Association for Computing Machinery, vol 23, No 3, pp 555-556, I976.

[49] – Schaub, M.A., Mermoud, G., “Ant Colony Optimization for Quadratic Assignment Problem”. Relatório Técnico. Retirado de http://mtc.epfl.ch/~mschaub/swarm_report.pdf

[50] - Simões, R.M. , Rangel, M.C., “Ordenação Parcial as Instancias do Problema Quadrático de Alocação”, 5º Encontro Regional de Matemática Aplicada e Computação, 2005.

[51] – Taillard, E.D., Hahn, P.M., Drezner, Z., "A Study of Quadratic Assignment Problem Instances that are Difficult for Meta-heuristic Methods".

[52] - Toran, J. “On the hardness of graph isomorphism”, 41st Annual Symposium on Foundations of Computer Science, Novembro de 2000.

[53] - Wigderson, A., “P, NP and Mathematics – a computational complexity perspective”, 2000.

[54] - Zhao,Q., Karisch,S., Rendl,F., Wolkowicz,H., "Semidefinite programming relaxation for the quadratic assignment problem", Journal of Combinatorial Optimization, vol. 2, pp. 71-109, 1998.