UM ALGORITMO CONSTRUTIVO BASEADO NO TEOREMA DAS INVERSÕES COM *PATH RELINKING* PARA O PROBLEMA QUADRÁTICO DE ALOCAÇÃO

Carlos Jones Rebello Junior
Universidade Federal do Espírito Santo - CT - Departamento de Informática
Av. Fernando Ferrari, 514 - *Campus* Goiabeiras - Vitória - ES - 29075-910
jones@ifes.edu.br

Maria Cristina Rangel
Universidade Federal do Espírito Santo - CT - Departamento de Informática
Av. Fernando Ferrari, 514 - *Campus* Goiabeiras - Vitória - ES - 29075-910
crangel@inf.ufes.br

RESUMO

O Problema Quadrático de Alocação (PQA) apresentado neste trabalho utiliza uma abordagem algébrica por meio de uma relaxação linear conhecida como Problema de Alocação Linear (PAL). Utiliza-se essa abordagem por existir na literatura o Teorema das Inversões, que associa o custo de uma solução do PQA ao número de inversões de sua correspondente linear no PAL. Para otimizar o processo de busca por soluções viáveis de boa qualidade, aplica-se o rastreamento de soluções através da construção de matrizes de dimensões $n \times (n-1)$ denotadas por $HeadQ$ e $HeadQ^*$. Combinando o rastreamento de soluções com o Teorema das Inversões, é apresentado um algoritmo construtivo que gera soluções iniciais de boa qualidade.

PALAVRAS CHAVE. Problema Quadrático de Alocação, Teorema das Inversões, *Path Relinking*, ÁREA PRINCIPAL. Otimização Combinatória.

ABSTRACT

The Quadratic Assignment Problem (QAP) presented in this paper uses an algebraic approach by a linear relaxation known as Linear Assignment Problem (LAP). We use this approach because the Inversion Theorem, found in literature, associates the cost of a solution to the QAP with the number of inversions of the corresponding linear LAP. To optimize the searching process for feasible solutions of good quality, applies tracking solutions by building dimensions matrices $n \times (n-1)$ denoted by $HeadQ$ e $HeadQ^*$. Combining tracking solutions with the Inversion Theorem we present a constructive algorithm to generate good initial solutions.

KEYWORDS. Quadratic Assignment Problem, Inversion Theorem, Path Relinking, MAIN AREA. Combinatorial Optimization.

1. Introdução

Introduzido por Koopmans e Beckman (1957), no contexto de atividades econômicas, o Problema Quadrático de Alocação (PQA) tem como objetivo determinar a alocação ótima de pares de atividades a pares de localidades, dadas as distâncias entre as n localidades e os fluxos entre as n atividades. O PQA é um dos problemas mais difíceis da classe *NP-hard* por possuir um alto grau de combinatoriedade, a ponto de problemas de dimensão $n \geq 30$ serem considerados de grande porte mesmo para os avançados recursos computacionais existentes hoje. Os estudos sobre o PQA basicamente dividem-se em dois grupos: algoritmos exatos ou ótimos e algoritmos heurísticos ou sub-ótimos.

Quando se fala em métodos exatos, uma grande dificuldade a destacar é a necessidade de se ter disponível plataformas computacionais poderosas e uma grande quantidade de memória para armazenamento. Como exemplos pode-se citar Anstreicher et al. (2002) que apresentam a solução ótima para a instância Kra30b com um tempo computacional de 182 dias com o uso de apenas uma estação de trabalho e também Anstreicher e Brixius (2001) com instância Nug30 resolvida de forma ótima após um período de 7 dias, porém, com um conjunto de mil computadores em rede ao redor do mundo. Dadas as limitações apresentadas pelos algoritmos exatos, uma grande quantidade de pesquisas surgiram voltadas ao estudo dos algoritmos heurísticos. Na literatura pode-se citar como exemplos o GRASP, proposto por Li, Pardalos e Resende (1994), que utilizam algoritmos construtivos associados a técnicas de melhoramento, a busca tabu em paralelo de James, Rego e Glover (2009) com o objetivo de aproveitar o ambiente paralelo para aumentar a intensificação e diversificação estratégica do algoritmo. Existem ainda as meta-heurísticas aplicadas ao PQA que utilizam como estratégia de melhoramento os fenômenos naturais, como o *Simulated Annealing* (SA) desenvolvido por Burkard e Rendl (1983) e também estão na literatura Seyedkashi et al. (2010), que propõem um algoritmo guloso para melhorar o SA, o Algoritmo Genético (AG), aplicado ao PQA por Misevicius(2003), estes implementam um algoritmo híbrido entre o AG e um procedimento chamado *ruin and recreate* (R *and R*), o Sistema da Colônia de Formigas, por Maniezzo, Colorni e Dorigo (1999) e por Gambardella, Taillard e Dorigo (1997) e mais recentemente uma meta-heurística baseada na formação em V de vôo de aves migratórias, proposta por Duman, Uysal e Alkaya (2012), *Migrating Birds Optimization*, que mostra-se muito eficiente quando comparada a várias técnicas meta-heurísticas conhecidas.

Este trabalho apresenta um algoritmo construtivo que utiliza o Teorema das Inversões (Rangel, 2000) e as matrizes propostas por Resendo e Rangel (2006), chamadas de $HeadQ$ e $HeadQ^*$, que mapeiam as soluções quadráticas do PQA no universo das soluções lineares da relaxação do PQA caracterizada pelo Problema de Alocação Linear (PAL). O algoritmo procura gerar soluções iniciais de boa qualidade e, como fase de melhoramento, aplica-se uma busca local e *path relinking*.

O trabalho está organizado da seguinte forma: na seção 2 descreve-se o Problema Quadrático de Alocação e sua relaxação linear na forma do Problema de Alocação Linear com o estudo da viabilidade das soluções quadráticas no universo de soluções lineares, e o Teorema das Inversões. Na seção 3 apresentam-se algumas técnicas utilizadas neste trabalho para a solução do problema e a proposta do algoritmo construtivo que é a contribuição deste trabalho. Na seção 4 mostram-se os resultados computacionais obtidos com os testes do algoritmo desenvolvido tanto serial quanto paralelo. Ao final, a seção 5 apresenta as conclusões e propostas para trabalhos futuros.

2. Definição do Problema

O Problema Quadrático de Alocação (PQA) foi introduzido por Koopmans e Beckmann (1957) no contexto de localização de atividades econômicas com o objetivo de determinar a alocação ótima de pares de atividades a pares de localidades, dados como entrada as distâncias entre as localidades e os fluxos entre as atividades. Para este problema, além do modelo proposto por Koopmans e Beckmann (1957), existem alguns modelos de formulações propostas como o PQA genérico e o PQA linear com o uso de variáveis binárias de Lawler (1963) e o PQA genérico de Burkard e Stratman (1978) que utilizam permutação de elementos no conjunto $\Omega^n = \{1, \ldots, n\}$ para representar as alocações.

Neste trabalho utiliza-se como modelo o PQA proposto por Burkard e Stratman (1978) que descreve matematicamente a busca pela melhor solução da seguinte forma:

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

onde os custos da função objetivo são apresentados segundo o contexto econômico dado por Koopmans e Beckmann pelo produto $f_{ij}d_{\varphi(i)\varphi(j)}$, tal que $1 \leq i, j \leq n$, Π_n é conjunto de todas as permutações φ de $\Omega^n = \{1, \ldots, n\}$, e F = $[f_{ij}]$ e D = $[d_{kl}]$ são matrizes de fluxo e distância respectivamente.

Pode-se dizer então que resolver uma instância PQA dadas as matrizes de ordem n, F e D, de coordenadas reais inteiras não negativas, denotada por PQA(F, D), é resolver a função

$$Z_{\varphi^*} = \min_{\varphi \in \Pi_n} \sum_{i,j=1}^{n} f_{ij}d_{\varphi(i)\varphi(j)}, \tag{2}$$

onde Π_n é conjunto de todas as permutações φ de $\Omega^n = \{1, \ldots, n\}$.

Tomem como exemplo a instância Gavett-Plyter, denotada por GP66, definida pelas matrizes da Figura 1. Chama-se φ uma permutação qualquer das cliques representada por uma sobreposição dos nós. A Figura 2 apresenta uma solução viável (φ) para o GP66, onde o custo é igual a 904, que representa-se por $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix}$, mas que denota-se apenas pela imagem $\varphi = \begin{pmatrix} 4 & 1 & 2 & 3 \end{pmatrix}$. Para este problema a solução ótima é $\varphi = \begin{pmatrix} 4 & 1 & 3 & 2 \end{pmatrix}$ cujo custo é 806.

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

Figura 1. Matrizes F e D para o GP66

2.1. Relaxação Linear do Problema Quadrático de Alocação

Considerando algumas características das matrizes F e D como simetria e diagonal principal nula, pode-se armazenar suas informações em vetores com dimensão $N = C_{n,2}$, pela ordem lexicográfica dos índices das matrizes. Para isso usa-se a bijeção $\psi(i, j) = ((i-1)n - i(i+2)/2) + j$, citada por Rangel (2000), que associa a cada par $(i, j) \in n \times n$ com $i < j$ um natural $z \in \{1, \ldots, N\}$ que representa uma aresta das cliques. Para as matrizes F e D da instância GP66 têm-se os vetores **F** = $\begin{pmatrix} 28 & 25 & 13 & 15 & 4 & 23 \end{pmatrix}$ e **D** = $\begin{pmatrix} 6 & 7 & 2 & 5 & 6 & 1 \end{pmatrix}$.

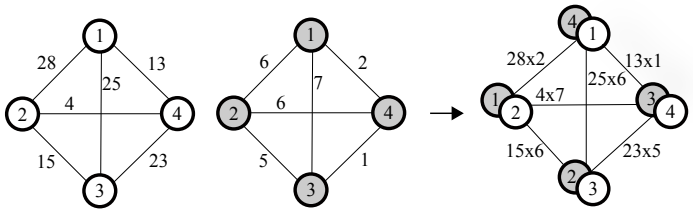

Figura 2. Sobreposição das cliques, uma solução viável para o problema GP66.

Considerando agora apenas as sobreposições de arestas, estabelece-se uma relaxação linear através do Problema de Alocação Linear (PAL), que pode ser definido como:

$$Z_{\xi^*} = \min_{\xi \in \Pi_N} \sum_{i=1}^{N} f_i d_{\xi(i)}, \tag{3}$$

onde N = $C_{n,2}$ e Π_N é o conjunto de todas as permutações ξ de $\Omega^N = \{1, \ldots, N\}$. O PAL é considerado uma relaxação de um PQA no sentido de o conjunto de soluções viáveis do primeiro conter as do segundo. O número de soluções do PAL, $N!$, é bem maior que o número de soluções quadráticas, $n!$, consequentemente, nem sempre uma permutação de arestas pode representar uma permutação de vértices, tais soluções são ditas não-viáveis para o PQA.

Considere as matrizes $Q = \mathbf{F}^t\mathbf{D}$ e $Q^* = (F^-)^t D^+$, Figura 3, onde F^- e D^+ são vetores com a ordenação não-crescente e não-decrescente de **F** e **D** respectivamente. Para encontrar a solução ξ ótima do PAL(Q) é muito simples, visto que, esta solução coincide com o traço da matriz Q^*, soma da diagonal principal, além disso este é um limite inferior para o PQA(Q) definido por **F** e **D**. Vale ressaltar que o problema PAL(Q^*) é um problema isomorfo ao PAL(Q), pois o conjunto de soluções viáveis do PAL(Q^*) é o mesmo que o PAL(Q). Denota-se por ξ a solução do PAL(Q) e por ρ a solução do PAL(Q^*).

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

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

Figura 3. Matrizes Q e Q* para o GP66

A pergunta a ser feita a partir deste momento é, dada uma permutação $\rho \in \Pi_N$, solução do PAL(Q^*), como saber se existe uma permutação de vértices da clique K_F sobre a clique K_D a ela correspondente?

A resposta para esta pergunta está em estabelecer uma relação de bijeção entre ξ e ρ assim como ocorre com φ e ξ na a bijeção ψ. Para isso armazenam-se as trocas feitas nos vetores durante a ordenação dos vetores **F** e **D** em permutações auxiliares denotadas por ϕ_F e ϕ_D, Rangel (2000). Para o exemplo da instância GP66 tem-se $\phi_F = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 5 & 4 & 6 & 3 \end{pmatrix}$ e $\phi_D = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 6 & 2 & 3 & 5 & 1 \end{pmatrix}$. Tendo as permutações auxiliares ϕ_F e ϕ_D obtém-se a relação entre ξ e ρ através da bijeção $\xi = \phi_D^{-1} \circ \rho \circ \phi_F$.

Admitindo uma ρ como sendo o traço da matriz, *tr(Q^*)*, tem-se a permutação identidade ρ = (1 2 3 4 5 6) para se obter a ξ_ρ através da bijeção, assim tem-se,

$\xi = \phi_D^{-1} \circ \rho \circ \phi_F$

$\xi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 3 & 4 & 1 & 5 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 5 & 4 & 6 & 3 \end{pmatrix}$

$\xi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 3 & 5 & 1 & 2 & 4 \end{pmatrix}$

Se dada uma ρ for possível construir φ através do caminho inverso ao da relaxação, diz-se então que há uma solução linear viável para o PQA(F, D). Para $\rho \in \Pi_N$ ser viável é necessário que exista uma $\varphi \in \Pi_n$ que satisfaça a seguinte sequência de igualdades:

$$f_r^- d_{\rho(r)}^+ = f_{\phi_F^{-1}(r)} d_{\phi_D^{-1}(\rho(r))} = f_p d_{\xi(p)} = f_{\psi^{-1}(p)} d_{\psi^{-1}(\xi(p))} = f_{ij} d_{kl}, \quad (4)$$

tal que r = $\{1, \ldots, N\}$; φ(i) = k e φ(j) = l; onde i, j, k, l = $\{1, \ldots, n\}$. Caso contrário, a solução é não-viável.

O algoritmo SolViável, proposto por Rangel (2000) apresentado a seguir verifica a viabilidade de ρ para a geração de uma φ viável.

Algorithm 1 SolViável

1: **Entrada:** ρ, ϕ_F e ϕ_D^{-1}
2: $\xi = \phi_D^{-1} \circ \rho \circ \phi_F$
3: **for** t = 1,... , N **do**
4: $ListaIJ[t] \leftarrow \psi^{-1}(t)$ e $ListaKL[t] \leftarrow \psi^{-1}(\xi(t))$
5: **end for**
6: **if** $\exists$ p $\in$ 1, ... , n tal que $\varphi(1) = p$ para as (n - 1) primeiras combinações **then**
7: $\varphi(1)$ =p
8: **for** i = 2, ... , n **do**
9: $[\varphi(i) = l \neq p] \vee [\varphi(i) = k \neq p]$ para $ListaKL[i-1]$
10: **end for**
11: **if** φ construída possui todas as N combinações compatíveis **then**
12: ρ é viável
13: **else**
14: ρ não é viável
15: **end if**
16: **else**
17: ρ não é viável
18: **end if**

É fácil observar neste algoritmo que para construir uma permutação φ bastam as (n-1) primeiras combinações (linhas 6 e 7). As (n-1) primeiras posições de ξ que satisfazem as linhas 6-10, Rangel (2000), denota-se por cabeça da permutação ξ. Esse conceito é importante, pois as matrizes $HeadQ$ e $HeadQ^*$ trabalham exatamente com essas posições.

2.2. Teorema das Inversões

Na literatura encontra-se o Teorema das Inversões demonstrado por Rangel (2000) que associa o custo de uma solução ρ, denotada por $Z(\rho)$, do PAL(Q^*) ao o seu número de inversões. Para que se possa compreender melhor este teorema deve-se antes conhecer algumas definições. Uma inversão é dada pelo par ($\rho(i)$, $\rho(j)$) tal que $\rho(j) < \rho(i)$ com i < j, i=$\{1, \ldots, N\}$. A quantidade de ocorrências de inversões em uma permutação é o chamado de número de inversões de ρ. Duas permutações ρ_1 e ρ_2 são chamadas de livremente comparáveis quando se pode afirmar que $Z(\rho_1) \leq (ou \geq) Z(\rho_2)$ independentemente dos

valores dos vetores F^- e D^+. Com base nessas definições, o Teorema das Inversões pode ser enunciado da seguinte forma: no conjunto das permutações $\rho \in \Pi_N$ que são livremente comparáveis seus custos crescem ou decrescem diretamente com seus números de inversões.

3. Técnicas de Solução do Problema

Nesta seção apresentam-se algumas técnicas utilizadas neste trabalho para a solução do problema e o algoritmo desenvolvido para este fim.

3.1. A Heurística Construtiva

Apesar do algoritmo SolViável ser capaz de reconhecer a viabilidade de uma solução linear para o problema quadrático, sabe-se que ele seria pouco eficiente, do ponto de vista computacional, se tiver de enumerar todas as soluções do PAL(Q^*) para conseguir reconhecer as suas respectivas soluções no PQA(F, D).

Para resolver este problema, Resendo (2004) propõe o rastreamento de soluções viáveis de boa qualidade através da construção de matrizes específicas de dimensões $n \times (n-1)$, denominadas matrizes $HeadQ$ e $HeadQ^*$.

Na matriz $HeadQ$ cada linha i representa permutações ξ que geram $\varphi(1)$ = i para todo i = $\{1, \ldots, n\}$.

Algorithm 2 ConstróiHeadQ2(N)

1: **for** t = 1, ... , N **do**
2: $ListaIJ[t] \leftarrow \psi^{-1}(t)$
3: $HeadQ[i, j-1] \leftarrow t$
4: $HeadQ[j, i] \leftarrow t$
5: **end for**

A Figura 4 a seguir mostra o formato da matriz $HeadQ$ considerando um $n = 4$.

	$\xi(1)$	$\xi(2)$	$\xi(3)$	
1	1	2	3	$\xi = \begin{pmatrix} 1 & 2 & 3 & \ldots \\ 1 & 2 & 3 & \ldots \end{pmatrix} \rightarrow \varphi(1) = 1 \rightarrow \varphi^1 = (\ 1\ \ 2\ \ 3\ \ 4\ )$
2	1	4	5	$\xi = \begin{pmatrix} 1 & 2 & 3 & \ldots \\ 1 & 4 & 5 & \ldots \end{pmatrix} \rightarrow \varphi(1) = 2 \rightarrow \varphi^2 = (\ 2\ \ 1\ \ 3\ \ 4\ )$
3	2	4	6	$\xi = \begin{pmatrix} 1 & 2 & 3 & \ldots \\ 2 & 4 & 6 & \ldots \end{pmatrix} \rightarrow \varphi(1) = 3 \rightarrow \varphi^3 = (\ 3\ \ 1\ \ 2\ \ 4\ )$
4	3	5	6	$\xi = \begin{pmatrix} 1 & 2 & 3 & \ldots \\ 3 & 5 & 6 & \ldots \end{pmatrix} \rightarrow \varphi(1) = 4 \rightarrow \varphi^4 = (\ 4\ \ 1\ \ 2\ \ 3\ )$

Figura 4. Formato da matriz *HeadQ* para $n = 4$ e geração das $\varphi's$ ordeandas

A partir das linhas 6-10 do Algoritmo 1 (SolViável), pode-se observar que a linha 1 gera a permutação $\varphi^1 = (1\ 2\ 3\ \ldots n)$, a linha 2 gera a permutação $\varphi^2 = (2\ 1\ 3\ \ldots n)$, linha 3 gera a permutação $\varphi^3 = (3\ 1\ 2\ \ldots n)$, e assim por diante até a linha n, que gera $\varphi^n = (n\ 1\ 2\ 3\ \ldots n-1)$. Essas permutações φ^i são chamadas neste trabalho de $\varphi's$ ordenadas. Se fizer uma troca de algumas componentes da linha da matriz, essa nova linha gera uma permutação φ com a mesma troca, lembrando que a primeira posição das $\varphi's$ são fixadas. Exemplo com $n = 4$, considere a linha 1 da $HeadQ$ igual a 1 2 3, faça uma troca qualquer, por exemplo, 3 2 1. A φ gerada por essa linha a partir do Algoritmo 1 (SolViável)

é φ = (1 4 3 2). As observações destacadas acima são importantes para o entendimento do algoritmo proposto neste trabalho.

Sabe-se que toda permutação $\xi \in \Pi_N$ no PAL(Q) possui uma correspondente $\rho \in \Pi_N$ no PAL(Q^*), sendo assim, Resendo (2004) propõe a construção de uma matriz $HeadQ^*$ que monta as permutações $\rho \in \Pi_N$ capazes de gerar uma $\varphi \in \Pi_n$, solução do PQA(F, D). É necessário que se trabalhe com as soluções do PAL(Q^*), pois o Teorema das Inversões é aplicado quando os vetores que definem o problema são ordenados tais como F^- e D^+. Para a construção da $HeadQ^*$, necessita-se novamente das permutações auxiliares ϕ_F e ϕ_D, pois estas permutações contêm uma memória das posições das trocas efetuadas para a geração dos vetores F^- e D^+ para a obtenção da relação entre ξ e ρ através da bijeção. Agora se faz uma bijeção inversa para efetuar a transformação entre essas matrizes.

Como se trabalha somente com cabeças das permutações $\xi \in \Pi_N$, é necessário assegurar que as imagens das aplicações ϕ_D^{-1} serão $1, \ldots, n-1$. Aplica-se ϕ_F nos valores de ξ(i) com i = $1, \ldots, n-1$, indicando em que posição os elementos das linhas da matriz $HeadQ^*$ estão na permutação $\rho \in \Pi_N$. Exemplificando esta situação, analise a instância GP66 dadas as suas permutações $\phi_F = \begin{pmatrix} 1 & 2 & 5 & 4 & 6 & 3 \end{pmatrix}$ e $\phi_D = \begin{pmatrix} 4 & 6 & 2 & 3 & 5 & 1 \end{pmatrix}$:

$$\phi_D \circ \xi \circ \phi_F^{-1} = \rho$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 6 & 2 & 3 & 5 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} \circ \begin{pmatrix} \mathbf{1} & \mathbf{2} & 3 & 4 & \mathbf{5} & 6 \\ \mathbf{1} & \mathbf{2} & 6 & 4 & \mathbf{3} & 5 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{2} & 3 & 4 & \mathbf{5} & 6 \\ \mathbf{4} & \mathbf{6} & & & \mathbf{2} & \end{pmatrix}$$

$$\circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 5 & & & \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{2} & 3 & 4 & \mathbf{5} & 6 \\ \mathbf{4} & \mathbf{3} & & & \mathbf{5} & \end{pmatrix}$$

$$\circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 6 & & & \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{2} & 3 & 4 & \mathbf{5} & 6 \\ \mathbf{6} & \mathbf{3} & & & \mathbf{1} & \end{pmatrix}$$

$$\circ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 6 & & & \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{2} & 3 & 4 & \mathbf{5} & 6 \\ \mathbf{2} & \mathbf{5} & & & \mathbf{1} & \end{pmatrix}$$

Para conhecer a imagem de cada elemento da matriz basta aplicar a função ϕ_D pois, como visto, a construção da matriz $HeadQ$ está baseada na permutação identidade de $\xi \in \Pi_N$ e a imagem da composição das permutações é exatamente ϕ_D. A Figura 5 mostra como fica a matriz $HeadQ^*$ depois de gerada. É importante lembrar que a $HeadQ$ possui o mesmo formato, independente dos valores dos dados de entrada do PQA. Contudo, a $HeadQ^*$ é totalmente dependente desses dados, pois é construída através das permutações que guardam as trocas feitas para as ordenações de F^- e D^+.

	$\phi_F(\xi(1))$	$\phi_F(\xi(2))$	$\phi_F(\xi(3))$
1	$\phi_D(1)$	$\phi_D(2)$	$\phi_D(3)$
2	$\phi_D(1)$	$\phi_D(4)$	$\phi_D(5)$
3	$\phi_D(2)$	$\phi_D(4)$	$\phi_D(6)$
4	$\phi_D(3)$	$\phi_D(5)$	$\phi_D(6)$

	1	2	5
1	4	6	2
2	4	3	5
3	6	3	1
4	2	5	1

Figura 5. Matriz $HeadQ^*$

3.2. *Path Relinking*

A técnica *Path Relinking* (PR), também conhecida como reconexão por caminhos, descrita originalmente por Glover (1996), consiste em explorar o caminho ou trajetória entre duas soluções dadas, uma chamada solução inicial e outra chamada solução guia que

é a solução a qual deseja-se chegar. Queiroz e Mendes (2011) explicam que basicamente a sua aplicação se dá a cada iteração realizando um movimento na solução inicial com o objetivo de aproximá-la da solução guia. Para que isso ocorra deve-se introduzir atributos encontrados na solução guia na solução inicial, até que ambas as soluções se tornem iguais. Como exemplo de aplicação em soluções do PQA observe a Figura 6, em vermelho estão sendo destacadas as trocas dos elementos, um em cada nível, com o objetivo de aproximar a cada passo, a solução inicial da solução guia. O caminho seguido pelo PR é compreendido como um processo de intensificação que permite explorar espaços de solução no caminho de aproximação entre as duas soluções. É possível que ao longo desta trajetória ótimos locais sejam alcançados, produzindo novas soluções viáveis de qualidade. Ainda para esse exemplo a solução marcada com o asterisco é a melhor solução encontrada no trajeto, que no caso do GP66 é a solução ótima do problema.

Quanto as formas de implementação Resende et al. (2010a) discutem a questão de "atualização estática" e "atualização dinâmica". Na "atualização estática" um conjunto elite é criado e atualizado ao longo das iterações do algoritmo, ao final o PR é aplicado como pós-otimização entre as soluções elite, com o intuito de explorar o espaço existente entre cada par destas soluções. Na "atualização dinâmica" para cada solução gerada durante uma iteração, uma solução do conjunto elite é sorteada e o PR é aplicado.

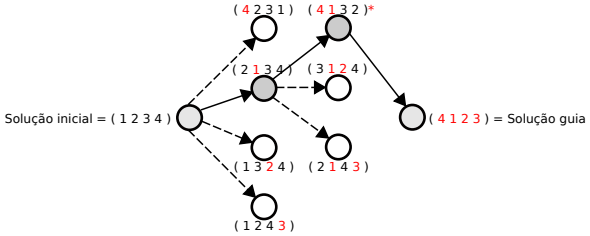

Figura 6. Um exemplo de *Path Relinking* para GP66

Há ainda algumas outras variações de PR na literatura citadas por Resende e Ribeiro (2003), por exemplo, *forward relinking*, Figura 6, partindo da solução inicial até alcançar a solução guia, *backward relinking*, que faz o inverso da técnica anterior e *mixed relinking*, onde as duas trajetórias são simultaneamente exploradas, a primeira iniciando com a solução inicial e a segunda iniciando com a solução guia até que se encontrem.

3.3. Algoritmo Proposto

O algoritmo proposto neste trabalho, Algoritmo Construtivo Baseado no Teorema das Inversões com *Path Relinking* (ACTI-*PR*), é um algoritmo que combina o método construtivo de soluções iniciais proposto por Resendo (2004), enriquecido com uma técnica de refinamento de soluções com forte embasamento no Teorema das Inversões, Rangel (2000), e uma diversificação caracterizada pelo *Path Relinking*.

Criou-se uma função denominada **DiminuirNumInverões** que avalia se uma troca feita na linha da $HeadQ^*$ induz uma diminuição do número de inversões levando-se em consideração as posições ocupadas no vetor ϕ_F, respectivamente. Observe o algoritmo a seguir.

O teste de linha 2 verifica se existe um valor maior, $HeadQ^*[k][i]$, que se encontra em uma posição menor, $\phi_F[i]$, na linha da $HeadQ^*$. De acordo com o Teorema das Inversões, seção 2.2, deve-se efetuar a troca do elemento $HeadQ^*[k][i]$ com o elemento

Algorithm 3 DiminuirNumInversões

1: Entrada de dados $(HeadQ^*, \phi_F, i, j, k)$;
2: **if** $((HeadQ^*[k][i] > HeadQ^*[k][j])$ **and** $(\phi_F[i] < \phi_F[j]))$ **or**
$((HeadQ^*[k][i] < HeadQ^*[k][j])$ **and** $(\phi_F[i] > \phi_F[j]))$ **then**
3: Retornar Verdadeiro
4: **else**
5: Retornar Falso
6: **end if**

$HeadQ^*[k][j]$ pois assim diminui-se o número de inversões da ρ associada a $HeadQ^*$, gerando por consequência uma nova φ de boa qualidade o problema.

O funcionamento do algoritmo proposto, **ACTI-*PR***, é apresentado a seguir em pseudo-código.

Algorithm 4 ACTI-*PR*

1: Entrada de dados (n, F, D);
2: gerar F^-, D^+, ϕ_F, ϕ_D, $HeadQ$, $HeadQ^*$ e φ's ordenadas: $\varphi^1, \varphi^2 \ldots \varphi^n$;
3: $custo_{\varphi_{best}} \leftarrow \infty$; $\varphi_{best} \leftarrow \emptyset$;
4: **for** k = 1, ... , n **do**
5: $\varphi_{bestLocal} \leftarrow$ Busca Local(φ^k);
6: $custo\varphi_{best} \leftarrow$ atualizar custo ($custo\varphi_{bestLocal}$, $custo\varphi_{best}$);
7: $\varphi_{best} \leftarrow$ atualizar ($\varphi_{bestLocal}$, $custo\varphi_{best}$, $custo\varphi_{bestLocal}$);
8: **for** i = 1, . . . , $(n$-1)-1 **do**
9: **for** j = i+1, . . . , n-1 **do**
10: **if** DiminuirNumInversões($HeadQ^*, \phi_F, i, j, k$) **then**
11: $\varphi \leftarrow$ troca($\varphi^k[i+1], \varphi^k[j+1]$);
12: $\varphi_{bestLocal} \leftarrow$ Busca Local(φ);
13: $custo\varphi_{best} \leftarrow$ atualizar custo ($custo\varphi_{bestLocal}$, $custo\varphi_{best}$);
14: $\varphi_{best} \leftarrow$ atualizar ($\varphi_{bestLocal}$, $custo\varphi_{best}$, $custo\varphi_{bestLocal}$);
15: **end if**
16: **end for**
17: **end for**
18: $\varphi_{aleatoria} \leftarrow$ gerar φ aleatória(n);
19: $\varphi_{bestPR} \leftarrow$ *Path Relinking*($\varphi_{aleatoria}$, $\varphi_{bestLocal}$);
20: $custo\varphi_{best} \leftarrow$ atualizar custo ($custo\varphi_{bestPR}$, $custo\varphi_{best}$);
21: $\varphi_{best} \leftarrow$ atualizar (φ_{bestPR}, $custo\varphi_{best}$, $custo\varphi_{bestPR}$);
22: **end for**
23: Retornar φ_{best} e $custo\varphi_{best}$;

Na linha 1 do algoritmo faz-se a leitura dos dados de entrada. Na linha 2 constrói-se os vetores ordenados F^- e D^+, necessários para a formar os vetores ϕ_F eϕ_D, que guardam as posições das trocas efetuadas na construção de F^- e D^+, conforme descrito na seção 2.1. Além disso, são construídas as matrizes $HeadQ$ e $HeadQ^*$ já apresentadas na seção 3.1. A construção da $HeadQ$ é necessária para se gerar as φ's ordenadas (ainda na linha 2). A linha 3 faz a inicialização das variáveis $custo\varphi_{best}$ e φ_{best}, que armazenam o custo da melhor solução encontrada para o problema e a sua permutação φ correspondente. O laço da linha 4 controla a linha k da $HeadQ^*$ e, por consequência, a φ^k ordenada que está sendo avaliada. Para cada valor de k, uma busca local é realizada na φ^k ordenada (linha 5) e as atualizações de $custo\varphi_{best}$ e φ_{best} são feitas nas linhas 6 e 7. Os laços das linhas 8 e 9 do algoritmo controlam os elementos a serem testados dois a dois na linha k da matriz $HeadQ^*$ e suas correspondentes posições no vetor ϕ_F. Tais testes são efetuados pela

função **DiminuirNumInversões**. Se o retorno for verdadeiro, faz-se a troca de elementos na solução φ^k que está sendo avaliada no momento (linhas 10 e 11). As linhas de 12 a 14 são responsáveis pela atualização das variáveis $custo\varphi_{best}$ e φ_{best} a cada troca de posições efetuadas. Antes do término de cada iteração de k é gerada uma $\varphi_{aleatria}$ (linha 18) utilizada no *path relinking* da linha 19 como solução inicial. As linhas 20 e 21 são responsáveis pela atualização das variáveis $custo\varphi_{best}$ e φ_{best} ao final de cada iteração do laço da linha 4. Ao se esgotar as n iterações do laço da linha 4, o algoritmo retorna a melhor solução encontrada para o problema, linha 23.

Neste trabalho optou-se pelo uso do *forward relinking* e a busca local deste algoritmo utiliza a estrutura de vizinhança 2-troca e percorre a árvore de busca em largura e profundidade. Observe que na proposta da técnica *path relinking* deste trabalho, o conjunto de soluções elite não foi construído. As soluções guias são as melhores soluções locais encontradas. Essa escolha se deu pela necessidade de diversificar os caminhos do *path relinking*.

4. Resultados Computacionais

A primeira implementação, em sua versão serial, foi executada em um notebook Intel Core 2 Duo T6500 2,1 GHz $\times$ 2, porém utilizou-se apenas um núcleo, e 3GB de memória RAM DDR2. Nessa etapa foram efetuados testes em instâncias de ordem até 30 disponíveis na QAPLIB (Burkard, 2013), pois o objetivo inicial foi analisar a qualidade das soluções no sentido de alcançar as soluções ótimas para o problema. Comparando os resultados obtidos nesta etapa com o algoritmo proposto por Resendo (2004), percebe-se uma maior eficiência do ACTI-*PR* em termos de aproximação do valor da solução ótima, ou melhor solução conhecida até o momento. A Tabela 1 a seguir apresenta o nome da instância, a melhor solução encontrada disponível na QAPLIB (Burkard, 2013) até o momento, os resultados do ACTI-*PR*, proposto por este trabalho, o resultado da HeuristicHead, proposta por Resendo (2004), e a porcentagem de erro em relação aos melhores resultados existentes.

Tabela 1. Comparação entre os algoritmos HeuristicHead e ACTI-*PR*

INSTÂNCIA	QAPLIB	ACTI-*PR*	HeuristicHead	% ERRO ACTI-*PR*	% ERRO HeuristicHead
nug12	578	578	582	0,00	0,68
rou12	235528	235528	235528	0,00	0,00
nug15	1150	1150	1152	0,00	0,17
tai15a	388214	388214	390782	0,00	0,66
nug20	2570	2570	2596	0,00	1,01
scr20	110030	110030	110058	0,00	0,03
tai30a	1818146	1818146	1858226	0,00	2,20
nug30	6124	6124	6156	0,00	0,52

Ainda com o objetivo de avaliar o algoritmo, além dos testes efetuados para comparação, foram realizadas outras baterias de testes, usando os dados da QAPLIB (Burkard, 2013), com todas as instâncias simétricas conhecidas como Chr, Els, Esc, Had, Kra, Nug, Rou, Scr, Tai, Tho de dimensão até 30. Em um total de 52 instâncias, todos os resultados obtidos alcançaram a melhor solução disponível para o problema, o que mostra a eficiência do algoritmo em termos de qualidade da solução.

Para problemas de grande porte os recursos computacionais, quando se utiliza apenas uma estação de trabalho, tornam-se insuficientes do ponto de vista de resposta com-

putacional. Para tentar resolver tal situação uma versão paralela para este algoritmo foi implementada e está em fase de testes. Em sua execução utilizou-se um cluster da Universidade Federal do Espírito Santo (UFES) disponibilizado pelo Laboratório de Computação de Alto Desempenho (LCAD), que tem como estrutura a seguinte configuração: 16 processadores Intel QUAD CORE de 2.4GHz por núcleo, onde o Master e demais nós possuem 2GB de memória RAM.

Inicialmente os testes computacionais foram feitos numa relação de 1 processador para cada 2 linhas da matriz $HeadQ$. Dependendo da dimensão da instância, esta relação não pode ser aplicada. Neste caso, seguindo a sugestão de Resendo e Rangel (2006), utiliza-se como número de processadores um valor que seja divisor exato da dimensão da instância. Este modelo inicial de teste é interessante pois o *speed up* entre os processos é minimizado. A Tabela 2 a seguir mostra os resultados computacionais obtidos com os primeiros testes da versão paralela deste trabalho. As instâncias Tai35a, Tai40a, Esc64a e Esc128 não foram testadas pela heurística proposta por Resendo e Rangel (2006).

Tabela 2. Resultados dos testes computacionais para a versão paralela do ACTI-*PR*

INSTÂNCIA	QAPLIB	ACTI-*PR*	HeuristicHead	% ERRO ACTI-*PR*	% ERRO HeuristicHead
tai35a	2422002	2435890	-	0,57	-
tai40a	3139370	3157972	-	0,59	-
sko42a	15812	15816	15910	0,03	0,60
tai50a	4938796	5007712	5090380	1,38	3,00
tai60a	7205962	7320012	7424510	1,56	3,00
esc64a	116	116	-	0,00	-
tai80a	13499184	13704816	13853702	1,50	2,50
esc128	64	64	-	0,00	-

5. Conclusão e Trabalhos Futuros

Neste trabalho foi proposto um algoritmo construtivo que aplica as noções do Teorema das Inversões com o objetivo de gerar soluções do PQA de boa qualidade. Pode-se notar através da análise dos resultados que as soluções obtidas pelo ACTI-*PR* alcançam as soluções ótimas ou as melhores disponíveis na QAPLIB (Burkard, 2013). Quando isso não ocorre, o erro médio das soluções do ACTI-*PR* é de 2,275. Sendo assim, a aplicação do Teorema das Inversões foi de grande valia para o sucesso das qualidades das soluções apresentadas.

Como trabalhos futuros pretende-se continuar os testes com a versão paralela do algoritmo em instâncias maiores e juntamente com esta etapa buscar melhorias na geração das soluções iniciais principalmente para uso em instâncias de grande porte. Esta etapa é de suma importância para tornar ACTI-*PR* competitivo em termos de desempenho computacional, visto que em termos de qualidade de solução, o algoritmo é satisfatório.

Referências

Anstreicher, K. M. e Brixius, N. W. (2001), A new bound for the quadratic assignment problem based on convex quadratic programming. Mathematical Programming, 89, 341-357.

Anstreicher, K. M. et al. (2002), Solving large quadratic assignment problem on computation grid. Mathematical Programming, 91, 563-588.

Burkard, R. E. e Rendl, F. (1983), A thermodynamically motivated simulation procedure for combinatorial optimization problems. European Journal of Operations Research, 17, 169-174.
Burkard, R. E. e Stratman, K. H. (1978), Numerical investigations on quadratic assigment problem. Naval Research Logistics Quartely, 148, 25-129.
Burkard, R. E. et al., QAPLIB - A Quadratic Assignment Problem Library <http://www.opt.math.tugraz.at/qaplib/> Última visita: 28/04/2013.
Duman, Uysal e Alkaya (2012), Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem, Information sciences, 217, 65-77.
Gambardella, L. M., Taillard, E. D. e Dorigo, M., Ant Colonies for the QAP. Technical Report IDSIA97-4, IDSIA, Laguno, Switzerland, 1997.
Glover, F. (1996), Tabu search and adaptive memory programming – advances, applications and challenges, Interfaces in Computer Science and Operations Research, 7, 1-75.
James, T., Rego, C. e Glover, F. (2009), A Cooperative Parallel Tabu Search Algorithm for the Quadratic Assignment Problem. European journal of operational research, 195, 810-826.
Koopmans, T. C. e Beckmann, M. J. (1957), Assignment problem and the location of economic activities. Econometrica, 25, 53-75.
Lawler, E. (1963), The quadratic assignment problem. Management Science, 9, 586-599.
Li, Y., Pardalos, P. M. e Resende, M. G. C. (1994), A greedy randomized adaptive search procedures for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16, 237-261.
Maniezzo, V., Colorni, A. e Dorigo, M. (1999), The Ant System Applied to the quadratic assignment problem. Knowledge and Data Engineering, IEEE Transactions, 11, 769-778.
Misevicius, A. (2003), Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. Knowledge-based systems, 16, 261-268.
Queiroz, M. M. e Mendes, A. B. (2011), Metaheurística Grasp com Path Relinking Aplicada ao Problema de Programação de Embarcações PLV. XLIII Simpósio Brasileiro de Pesquisa Operacional, 1723-1734.
Rangel, M. C., Contribuições Algébricas ao Problema Quadrático de Alocação. Tese de Doutorado, Programa de Engenharia de Produção – COPPE/UFRJ, Brasil, 2000.
Resende, M. G. C. e Ribeiro, C. C. (2003), Grasp with path relinking: recent advances and applications. AT & T Labs Technical Report TD-5TU726, 1-24.
Resende, M. G. C. et al. (2010a), Grasp and path relinking for the max-min diversity problem. Computers & Operations Research, 37, 498-508.
Resendo L. C. (2004), Um mapeamento de soluções do Problema Quadrático de Alocação(PQA) no universo de soluções do Problema de Alocação Linear (PAL). Tese de Mestrado, Programa de Pós-Graduação em Informática - UFES/PPGI, Brasil, 2004.
Seyedkashi et al. (2010), New Simulated Annealing Algorithm for Quadratic Assignment Problem . ADVCOMP 2010 : The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences , 105-110.