UM ALGORITMO DUAL DISTRIBUÍDO PARA A FORMULAÇÃO RLT DE NÍVEL 3 APLICADA AO PROBLEMA QUADRÁTICO DE ATRIBUIÇÃO

Alexandre Domigues Gonçalves
Lúcia Maria de Assumpção Drummond
Instituto de Computação - Universidade Federal Fluminense - UFF
Rua Passo da Pátria 156 - Bloco E - 3º andar - São Domingos - Niterói - RJ
{*agoncalves,lucia*}*@ic.uff.br*

Artur Alves Pessoa
Engenharia de Produção - Universidade Federal Fluminense - UFF
Rua Passo da Pátria 156 - Bloco E - 4º andar - São Domingos - Niterói - RJ
artur@producao.uff.br

Peter M. Hahn
Electrical and Systems Engineering, The University of Pennsylvania
Philadelphia, PA 19104-6315, USA
hahn@seas.upenn.edu

RESUMO

A aplicação da técnica de reformulação-linearização (RLT) ao problema quadrático de atribuição (QAP) leva a uma relaxação linear relativamente justa porém de grandes dimensões e difícil resolução. Trabalhos anteriores mostram que a utilização dessas relaxações em métodos de *branch-and-bound* fazem parte do estado-da-arte da resolução exata do QAP. No caso da RLT de nível 3 (RLT3), a utilização dessa relaxação é proibitiva em computadores convecionais para instâncias com mais de 22 locais por limitações de memória. Este artigo apresenta uma versão distribuída de um algoritmo dual de subida para a formulação RLT3 do QAP que permite resolvê-lo de forma aproximada para instâncias com até 30 locais pela primeira vez. Comparações com outros limitantes inferiores da literatura mostram que a nossa versão distribuída gera os melhores limites conhecidos em 26 das 28 instâncias testadas, sendo que 18 destes limites alcançam a solução ótima.

PALAVRAS CHAVE: QAP, RLT, sistemas distribuídos.

ABSTRACT

The application of the reformulation-linearization technique (RLT) to the quadratic assignment problem (QAP) leads to a tight linear relaxation with huge dimensions that is hard to solve. Previous works found in the literature show that these relaxations combined with branch-and-bound algorithms belong to the state-of-the-art of exact methods for the QAP. For the level 3 RLT (RLT3), using this relaxation is prohibitive in conventional machines for instances with more than 22 locations due to memory limitations. This paper presents a distributed version of a dual ascent algorithm for the RLT3 QAP relaxation that approximately solves it for instances with up to 30 locations for the first time. When compared to other lower bounding methods found in the literature, our distributed version generates the best known lower bounds for 26 out of the 28 tested instances, reaching the optimal solution in 18 of them.

KEYWORDS: QAP, RLT, distributed systems.

1 Introdução

Dados N objetos, N locais, um fluxo f_{ik} de cada objeto i para cada objeto k, $k \neq i$, e uma distância d_{jl} de cada local j para cada local l, $l \neq j$, o problema quadrático de atribuição consiste em atribuir cada objeto i a exatamente um local distinto $a(i)$ de modo a minimizar $\sum_{i=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} f_{ik} \times d_{a(i),a(k)}$. Apresentado inicialmente por Koopmans e Beckmann (1957), o QAP tem aplicações práticas em alocação de objetos em departamentos, design de placas de circuitos eletrônicos, problemas de layout, planejamento de construções, entre outros.

O problema quadrático de atribuição, aqui no trabalho referenciado pela sua sigla em inglês (QAP), é um dos mais difíceis e mais estudados problemas de otimização combinatória da literatura, dentre os estudos relacionados às aplicações do QAP temos: Heffley (1980) em problemas econômicos, Francis *et al*. (1992) com um *framework* para atribuição de facilidades e locações, Hubert (1986) em análises estatísticas, Dickey e Hopkins (1972) em alocações em um campus universitário e Elshafei (1977) em um planejamento hospitalar.

Métodos exatos para o QAP levam um tempo demasiadamente longo para resolver instâncias consideradas difíceis. No tabalho de Adams *et al*. (2007), a resolução de uma instância com 30 locais, executada em um computador Dell 7150 PowerEdge server, levou cerca de 1848 dias.

Bons limites inferiores são indispensáveis na resolução exata de instâncias com mais de 15 locais pois permitem descartar um maior número de soluções alternativas na busca pela solução ótima através do emprego de algoritmos de *branch-and-bound*. Um resumo das técnicas de cálculo de limites inferiores pode ser visto em Loiola *et al*. (2007), neste destacamos os limites alcançados por Gilmore (1962) e Lawler (1963), cujas implementações são mais simples e de baixo custo computacional, entretanto o *gap* em relação à solução ótima cresce demasiadamente de acordo com o tamanho do problema.

Uma tabela que compara os melhores limites inferiores alcançados em cada instância pode ser observada no site do QAPLIB, *http://www.seas.upenn.edu/qaplib/lowerbound.html*. Os limites alcançados por Burer e Vandenbussche (2006), Adams *et al*. (2007) e Hahn *et al*. (2012) destacam-se como os mais próximos dos apresentados no site. A proposta de Burer e Vandenbussche (2006) consiste em relaxações *lift-and-project* de programas binários inteiros, Adams *et al*. (2007) apresenta um algoritmo dual Hahn-Hightower RLT2 aplicado ao QAP e a proposta do Hahn *et al*. (2012) que consiste no algoritmo dual de subida Hahn-Zhu RLT3.

O algoritmo de subida para a avaliação dos limites de relaxação linear da formulação RLT3 descrito em Hahn *et al*. (2012) obtém resultados de baixo *gap* ou coincidem com a solução ótima em algumas instâncias. Instâncias maiores que 25 locais não tinham sido avaliadas na formulação RLT3 até então, devido ao tamanho de memória RAM necessária que deverá ser superior aos 173 GB requiridos para a de 25 locais.

Este trabalho apresenta uma versão distribuída baseada no algoritmo dual de subida de Hahn *et al*. (2012) que utiliza a formulação RLT3 na busca do limite inferior. A versão distribuída poderá ser executada em *clusters* compostos de máquinas de uso comercial e permitirá execuções de busca de limites inferiores em instâncias maiores que 25 locais. A nossa proposta difere de Hahn *et al*. (2012) quanto ao uso do tipo de dados *float* nos coeficientes das matrizes em combinação com a média aritmética na transferência de custos entre complementares, o que possibilita alcançar os bons resultados apresentados na seção 5.

2 A formulação do problema quadrático de atribuição - QAP

Consideremos f_{ik} o fluxo entre os objetos i e k e d_{jn} a distância entre as localizações j e n. Deseja-se então calcular:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} \sum_{\substack{n=1 \\ n \neq j}}^{N} f_{ik} d_{jn} x_{ij} x_{kn} : x \in X,\ x \in \{0,1\} \tag{1}$$

$$\text{Onde } X \equiv \left\{ x_{ij} \geq 0 \, , \sum_{i=1}^{N} x_{ij} = 1 \; \forall \; j = 1, ..., N \, , \; \sum_{j=1}^{N} x_{ij} = 1 \; \forall \; i = 1, ..., N \right\} \tag{2}$$

Considerando b_{ij} o custo de alocação do objeto i à posição j e a substituição de $f_{ik}d_{jn}$ por c_{ijkn}, proposta por Lawer(1963), teremos:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} \sum_{\substack{n=1 \\ n \neq j}}^{N} c_{ijkn} x_{ij} x_{kn} + \sum_{i=1}^{N} \sum_{j=1}^{N} b_{ij} x_{ij} : x \in X, \; x \in \{0, 1\} \tag{3}$$

É importante destacar a existência de elementos complementares de custo, ou seja, se um elemento de custo c_{ijkn} $(i \neq k \text{ e } j \neq n)$ fizer parte da solução $(x_{ij}x_{kn} = 1)$, então seu elemento de custo complementar c_{knij} também fará parte da solução.

3 Técnica de reformulação-linearização aplicada ao QAP

A técnica de reformulação-linearização, ou *reformulation-linearization technique* (RLT), foi desenvolvida inicialmente por Adams e Sherali (1986), com o objetivo de gerar relaxações justas para problemas de programação linear convexos e não contínuos.

A RLT consiste de 2 etapas: a reformulação e a linearização. Na etapa de reformulação, um conjunto de fatores variáveis não negativos é definido, produtos entre esses fatores e as restrições originais são formados gerando várias restrições não lineares. Na etapa de linearização, uma técnica de substituição de variáveis é utilizada para linearizar as restrições não lineares.

A RLT estabelece um z-nivel hierárquico de relaxações. Para um dado $z \in \{i, .., n\}$, o nível-z RLT, ou RLTz, constrói vários fatores polinomiais de grau z que consistem do produto das mesmas variáveis binárias x_j ou de seus complementares $(1 - x_j)$. Encontramos na literatura vários níveis de RLT aplicados ao QAP, RLT1 em Hahn e Grant (1998), RLT2 em Adams *et al.* (2007) e RLT3 em Hahn *et al.* (2012).

A reformulação RLT3, apresentada em Hahn *et al.* (2012), consiste em: multiplicar cada uma das $2N$ restrições de alocação por cada uma das N^2 variáveis binárias x_{ij} (aplicação do RLT1); multiplicar cada uma das $2N$ restrições de alocação por cada um dos $N^2(N-1)^2$ produtos $x_{ij}x_{kn}$, $k \neq i \; e \; n \neq j$ (aplicação do RLT2); multiplicar cada uma das $2N$ restrições de alocação por cada um dos $N^2(N-1)^2(N-2)^2$ produtos $x_{ij}x_{kn}x_{pq}$, $p \neq k \neq i$ e $q \neq n \neq j$ (aplicação do RLT3); substituir $x_{ij} = x_{ij}^2$ sempre que tais produtos ou similares aparecerem; remover os produtos $x_{ij}x_{kn}$ se $(k = i \; e \; n \neq j)$ ou $(k \neq i \; e \; n = j)$ em expressões quadráticas; remover todos os produtos $x_{ij}x_{kn}x_{pq}$, se $(p = i \; e \; q \neq j)$, $(p = k \; e \; q \neq n)$, $(p \neq i \; e \; q = j)$ ou $(p \neq k \; e \; q = n)$ em expressões cúbicas; remover todos os produtos $x_{ij}x_{kn}x_{pq}x_{gh}$ se $(g = i \; e \; h \neq j)$, $(g = k \; e \; h \neq n)$, $(g = p \; e \; h \neq q)$, $(g \neq i \; e \; h = j)$, $(g \neq k \; e \; h = n)$ ou $(g \neq p \; e \; h = q)$ em expressões biquadráticas.

A linearização consiste em: substituir cada produto $x_{ij}x_{kn}$ com $i \neq k$ e $j \neq n$, pela variável contínua y_{ijkn}, impondo a restrição $y_{ijkn} = y_{knij}$ (2 complementares) para todo (i, j, k, n) com $i < k$ e $j \neq n$ (aplicação do RLT1); substituir cada produto $x_{ij}x_{kn}x_{pq}$ com $i \neq k \neq p$ e $j \neq n \neq q$ pela variável contínua z_{ijknpq}, impondo a restrição $z_{ijknpq} = z_{ijpqkn} = z_{knijpq} = z_{knpqij} = z_{pqijkn} = z_{pqknij}$ (6 complementares) para todo (i, j, k, n, p, q) com $i < k < p$ e $j \neq n \neq q$ (aplicação do RLT2); substituir cada produto $x_{ij}x_{kn}x_{pq}x_{gh}$ por $v_{ijknpqgh}$, com $i \neq k \neq p \neq g$ e $j \neq n \neq q \neq h$, pela variável contínua $v_{ijknpqgh}$, impondo a restrição $v_{ijknpqgh} = v_{ijknghpq} = ... = v_{ghpqknij}$ (24 complementares) para todo (i, j, k, n, p, q, g, h) com $i < k < p < g$ e $j \neq n \neq q \neq h$ (aplicação do RLT3).

Ao final da reformulação RLT3 alcançaremos a seguinte formulação:

$$min \left\{ \sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}x_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{\substack{n=1\\n\neq j}}^{N} C_{ijkn}y_{ijkn} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{\substack{n=1\\n\neq j}}^{N}\sum_{\substack{p=1\\p\neq i,k}}^{N}\sum_{\substack{q=1\\q\neq j,n}}^{N} D_{ijknpq}z_{ijknpq} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{\substack{n=1\\n\neq j}}^{N}\sum_{\substack{p=1\\p\neq i,k}}^{N}\sum_{\substack{q=1\\q\neq j,n}}^{N}\sum_{\substack{g=1\\g\neq i,k,p}}^{N}\sum_{\substack{h=1\\h\neq j,n,q}}^{N} E_{ijknpqgh}v_{ijknpqgh} \right\} \quad (4)$$

$$\sum_{\substack{g=1\\g\neq i,k,p}}^{N} v_{ijknpqgh} = z_{ijknpq} : (i,j,k,n,p,q,h = 1,..,N)\ ,\ h \neq q \neq n \neq j\ ,\ p \neq k \neq i \quad (5)$$

$$\sum_{\substack{h=1\\h\neq j,n,q}}^{N} v_{ijknpqgh} = z_{ijknpq} : (i,j,k,n,p,q,g = 1,..,N)\ ,\ q \neq n \neq j\ ,\ g \neq p \neq k \neq i \quad (6)$$

$$v_{ijknpqgh} = v_{ijknghpq} = v_{ijpqkngh} = ... = v_{ghpqknij}\ (24\ complementares)\ : \\ (i,j,k,n,p,q,g,h = 1,..,N)\ ,\ i < k < p < g\ ,\ j \neq n \neq q \neq h \quad (7)$$

$$v_{ijknpqgh} \geq 0\ :\ (i,j,k,n,p,q,g,h = 1,..,N)\ ,\ i < k < p < g\ ,\ j \neq n \neq q \neq h \quad (8)$$

$$\sum_{\substack{p=1\\p\neq i,k}}^{N} z_{ijknpq} = y_{ijkn}\ :\ (i,j,k,n,q = 1,..,N)\ ,\ q \neq n \neq j,k \neq i \quad (9)$$

$$\sum_{\substack{q=1\\q\neq j,n}}^{N} z_{ijknpq} = y_{ijkn}\ :\ (i,j,k,n,p = 1,..,N)\ ;\ n \neq j,p \neq k \neq i \quad (10)$$

$$z_{ijknpq} = z_{ijpqkn} = z_{knijpq} = z_{knpqij} = z_{pqijkn} = z_{pqknij}\ (6\ complementares)\ : \\ (i,j,k,n,p,q = 1,..,N)\ ,\ i < k < p\ ,\ j \neq n \neq q \quad (11)$$

$$z_{ijknpq} \geq 0\ :\ (i,j,k,n,p,q = 1,..,N)\ ,\ i < k < p\ ,\ j \neq n \neq q \quad (12)$$

$$\sum_{\substack{k=1\\k\neq i}}^{N} y_{ijkn} = x_{ij}\ :\ (i,j,n = 1,..,N)\ ,\ n \neq j \quad (13)$$

$$\sum_{\substack{n=1\\n\neq j}}^{N} y_{ijkn} = x_{ij}\ :\ (i,j,k = 1,..,N)\ ,\ k \neq i \quad (14)$$

$$y_{ijkn} = y_{knij}\ (2\ complementares)\ :\ (i,j,k,n = 1,..,N)\ ,\ i < k\ ,\ j \neq n \quad (15)$$

$$y_{ijkn} \geq 0\ :\ (i,j,k,n = 1,..,N)\ ,\ i < k\ ,\ j \neq n \quad (16)$$

Na função objetivo, Expressão (4), cada elemento $B_{ij} = 0 \, \forall \, (i,j)$, cada elemento $C_{ijkn} = f_{ik} \times d_{jn} \, \forall \, (i,j,k,n)$ tal que $i \neq k$ e $j \neq n$, cada elemento $D_{ijknpq} = 0 \, \forall \, (i,j,k,n,p,q)$ tal que $i \neq k \neq p$ e $j \neq n \neq q$, cada elemento $E_{ijknpqgh} = 0 \, \forall \, (i,j,k,n,p,q,g,h)$ tal que $i \neq k \neq p \neq g$ e $j \neq n \neq q \neq h$.

O algoritmo dual de subida implementado neste trabalho consiste em modificar o valor do *Lower Bound* (LB) e dos elementos das matrizes B, C, D e E de modo que nenhum valor se torne negativo e o custo de qualquer solução viável para o QAP permaneça inalterado após a modificação de acordo com a formulação acima, Expressões (4) a (16). Como conseqüência desta propriedade, o valor do LB em qualquer instante da execução do algoritmo é um limite inferior válido para o custo da solução ótima. Assim, as seguintes modificações são válidas:

I. **Espalhamento de custos:** consiste nas distribuições de custos da matriz B para C, da matriz C para D e da matriz D para E. Seguem as respectivas descrições:

- Para cada (i,j), o elemento de custo B_{ij} é espalhado nas $(N-1)$ linhas da matriz C, ou seja, cada elemento de custo C_{ijkn} recebe um acréscimo de $B_{ij}/(N-1)$, $\forall \, k \neq i \, e \, n \neq j$. Após atualização, para cada (i,j), $B_{ij} = 0$.
- Para cada (i,j,k,n), o elemento de custo C_{ijkn} é espalhado nas $(N-2)$ linhas da matriz D, ou seja, cada elemento de custo D_{ijknpq} recebe um acréscimo de $C_{ijkn}/(N-2)$, $\forall \, p \neq i,k$ e $q \neq j,n$. Após atualização, para cada (i,j,k,n), $C_{ijkn} = 0 \, \forall \, k \neq i$ e $n \neq j$.
- Para cada (i,j,k,n,p,q), o elemento de custo D_{ijknpq} é espalhado nas $(N-3)$ linhas da matriz E, ou seja, cada elemento de custo $E_{ijknpqgh}$ recebe um acréscimo de $D_{ijknpq}/(N-3)$, $\forall \, g \neq i,k,p$ e $h \neq j,n,q$. Após atualização, para cada (i,j,k,n,p,q), $D_{ijknpq} = 0 \, \forall \, p \neq i,k$ e $q \neq j,n$.

II. **Concentração de custos:** para esta modificação adotamos o uso do Algoritmo Húngaro, ver Munkres (1957). O Algoritmo Húngaro tem o objetivo de minimizar o custo total S de uma atribuição. Tomemos como exemplo uma matriz de custos M de dimensão N^2, onde cada elemento M_{rs} corresponde ao custo de atribuir um recurso r a uma atividade s. A função objetivo fica, então: $min \; S = \sum_r^N \sum_s^N M_{rs} x_{rs}$ onde $x_{rs} \in \{0,1\}$, $\sum_{r=1}^N x_{rs} = 1 \, \forall \, s \in \{1,..,N\}$, $\sum_{s=1}^N x_{rs} = 1 \, \forall \, r \in \{1,..,N\}$. As concentrações são feitas da matriz E para D, da matriz D para C, da matriz C para B, e da matriz B para o limite LB. Seguem as respectivas descrições:

- Seja M uma matriz de dimensão $(N-3)^2$. Para cada (i,j,k,n,p,q), M recebe os $(N-3)^2$ elementos de custo da submatriz E_{ijknpq}. Para cada $(r,s = 1,..,N-3)$, M_{rs} recebe $E_{ijknpqgh} \, \forall \, g \neq i,k,p$ e $h \neq j,n,q$. Obtém-se S a partir da aplicação do algoritmo húngaro em M e armazena em D_{ijknpq}. Os elementos de custos de $E_{ijknpqgh}$, $\forall \, g \neq i,k,p$ e $h \neq j,n,q$ são substituidos pelos correspondentes coeficientes residuais de M, ver Munkres (1957). Representamos esta transferência de custos como: $D_{ijknpq} \leftarrow Hungaro(E_{ijknpq})$.
- Seja M uma matriz de dimensão $(N-2)^2$. Para cada (i,j,k,n), M recebe os $(N-2)^2$ elementos de custos da submatriz D_{ijkn}. Para cada $(r,s = 1,..,N-2)$, M_{rs} recebe $D_{ijknpq} \, \forall \, p \neq i,k$ e $q \neq j,n$. Obtém-se S a partir da aplicação do algoritmo húngaro em M e armazena em C_{ijkn}. Os elementos de custos de D_{ijknpq}, $\forall \, p \neq i,k$ e $q \neq j,n$ são substituidos pelos correspondentes coeficientes residuais de M. Representamos esta transferência de custos como: $C_{ijkn} \leftarrow Hungaro(D_{ijkn})$.
- Seja M uma matriz de dimensão $(N-1)^2$. Para cada (i,j), M recebe os $(N-1)^2$ elementos da submatriz C_{ij}. Para cada $(r,s = 1,..,N-1)$, M_{rs} recebe $C_{ijkn} \, \forall \, k \neq i$

e $n \neq j$. Obtém-se S a partir da aplicação do algoritmo húngaro em M e armazena em B_{ij}. Os custos de C_{ijkn}, $\forall\ i \neq k$ e $j \neq n$ são substituidos pelos correspondentes coeficientes residuais de M. Representamos esta transferência de custos como: $B_{ij} \leftarrow Hungaro(C_{ij})$.

- Seja M uma matriz de dimensão N^2. M recebe os N^2 elementos de B. Para cada $(r, s = 1, .., N)$, M_{rs} recebe B_{ij}. Obtém-se S a partir da aplicação do algoritmo húngaro em M e adiciona em LB. Os custos de B_{ij} são substituidos pelos correspondentes coeficientes residuais de M. Representamos esta transferência de custos como: $LB \leftarrow LB + Hungaro(B)$.

III. **Transferência de custos entre complementares:** As restrições (7), (11) e (15) referenciam os complementares nas matrizes E, D e C respectivamente. Definimos em nossa proposta, que a transferência de custos é feita através da média aritmética entre os elementos complementares de custos em cada matriz. Seguem as descrições:

- Na matriz C, para cada (i, j, k, n), $C_{ijkn} \leftarrow C_{knij} \leftarrow (C_{ijkn} + C_{knij})/2$, onde $i < k$ e $j \neq n$
- Na matriz D, para cada (i, j, k, n, p, q), $D_{ijknpq} \leftarrow D_{ijpqkn} \leftarrow D_{knijpq} \leftarrow D_{knpqij} \leftarrow D_{pqijkn} \leftarrow D_{pqknij} = (D_{ijknpq} + D_{ijpqkn} + D_{knijpq} + D_{knpqij} + D_{pqijkn} + D_{pqknij})/6$, onde $i < k < p$ e $j \neq n \neq q$
- Na matriz E, para cada (i, j, k, n, p, q, g, h), $E_{ijknpqgh} \leftarrow E_{ijknghpq} \leftarrow ... \leftarrow E_{ghpqknij} \leftarrow (E_{ijknpqgh} + E_{ijknghpq} + ... + E_{ghpqknij})/24$, onde $i < k < p < g$ e $j \neq n \neq q \neq h$

4 Algoritmo proposto

Em nossa versão distribuída, consideremos T o conjunto de nós que executam a aplicação, e R_t ($R_t \in T$) a identificação de um nó participante da execução.

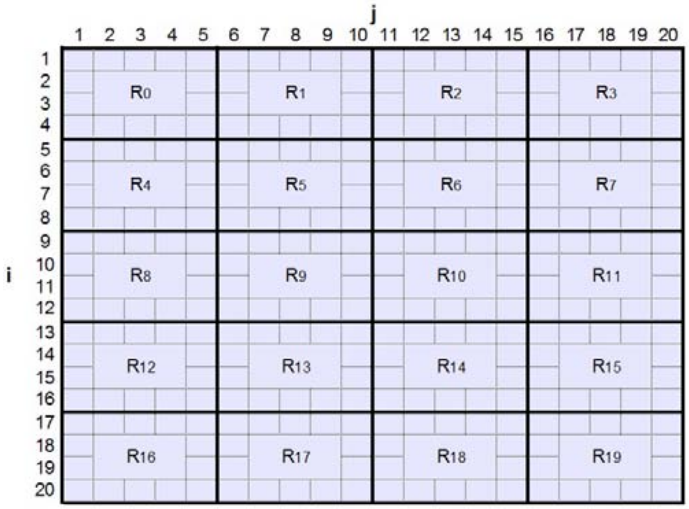

Figura 1: Exemplo de distribuição de conjuntos G_{ij} pelos nós da aplicação

Sejam d_{ik} e f_{jn} as matrizes de distância e de fluxo respectivamente, conforme Expressão (1), LB o limite inferior e B,C,D e E as matrizes com os elementos de custo apresentadas na função objetivo, Expressão (4). Consideremos G_{ij} um conjunto formado pelas submatrizes B, C, D, E de mesmo $\{ij\}$ armazenado e processado no nó R_t.

A distribuição da carga pelos nós é feita alocando-se para cada nó R_t, um ou mais conjuntos G_{ij}. Com o propósito de ter uma execução com cargas similares entre todos os nós, distribui-se a mesma quantidade de G_{ij} para cada R_t. A nossa implementação permite distribuições com quantidades diferentes de conjuntos por nó, no entanto, os que tiverem uma carga menor, certamente ficarão mais tempo no estado de espera pelas mensagens dos nós com maior carga.

Como exemplo, tomemos um cenário com 20 nós participantes da aplicação que executa uma instância com tamanho $N = 20$. A organização dos conjuntos e nós obedece à distribuição mostrada na Figura 1. Observemos que o conjunto $G_{15,7}$ composto das submatrizes $B_{15,7}$, $C_{15,7,k,n}$, $D_{15,7,k,n,p,q}$ e $E_{15,7,k,n,p,q,g,h}$ é armazenado e processado no nó R_{13}. Outras formas de distribuição poderão ser feitas, desde que utilize G_{ij} como unidade de distribuição.

A execução do algoritmo RLT3 aplicado ao QAP exige uma grande quantidade de memória RAM para armazenamento dos coeficientes de suas matrizes. Uma instância $N = 30$, por exemplo, tem a necessidade de uma memória principal de cerca de 1,6 Tbytes apenas para a matriz E. A matriz E é composta de N^2 x $(N-1)^2$ x $(N-2)^2$ x $(N-3)^2$ posições, cada uma armazenando um coeficiente do tipo inteiro ou float (4 bytes). Podemos reduzir a memória necessária alocando complementares em uma mesma posição na memória. Na matriz E, cada coeficiente faz parte de um grupo de 24 complementares, na matriz D são 6 complementares e na matriz C são 2 complementares. Com o compartilhamento de posição, a memória necessária para matriz E, matriz D e matriz C passa a ser respectivamente $\frac{1}{24}$, $\frac{1}{6}$ e $\frac{1}{2}$ da memória necessária sem compartilhamento. Essa redução foi utilizada por Hahn *et al*. (2012) em seus testes.

Algoritmo 1: *Versão Distribuída - Algoritmo a ser executado em* R_t

início
  $LB \leftarrow 0$; *cont* $\leftarrow 1$; $lim \leftarrow$ *limite de iterações*; $otimo \leftarrow$ *ótimo conhecido*
  Para cada (i,j,k,n,p,q,g,h), $E_{ijknpqgh} \leftarrow 0$, $D_{ijknpq} \leftarrow 0$, $B_{ij} \leftarrow 0$
  Para cada $(i,j,k,n), i \neq k$ *e* $j \neq n$, $C_{ijkn} \leftarrow f_{ik} \times d_{jn}$
  Para cada $R_s \in T$ *e* $R_s \neq R_t$ *enviar* $Comp(C)_{ts}$
  Para cada G_{ij} *em* R_t, $C_{ijkn} \leftarrow (C_{ijkn} + C_{knij})/2$
  Para cada G_{ij} *em* R_t, $B_{ij} \leftarrow Hungaro(C_{ij})$
  Para cada $R_s \in T, R_s \neq R_t$ *enviar* $Mens(B)_{ts}$
  $LB \leftarrow LB + Hungaro(B)$
  loop *Enquanto* $(LB \neq otimo)$ *e* $(cont < lim)$
    Para cada G_{ij} *em* R_t, $C_{ijkn} \leftarrow C_{ijkn} + B_{ij}/(N-1)$
    Para cada G_{ij} *em* R_t, $D_{ijknpq} \leftarrow D_{ijknpq} + C_{ijkn}/(N-2)$
    Para cada G_{ij} *em* R_t, $E_{ijknpqgh} \leftarrow E_{ijknpqgh} + D_{ijknpq}/(N-3)$
    Para cada $R_s \in T$ *e* $R_s \neq R_t$, *enviar* $Comp(E)_{ts}$
    Para cada G_{ij} *em* R_t, $E_{ijknpqgh} \leftarrow (E_{ijknpqgh} + E_{ijknghpq} + ... + E_{ghpqknij})/24$
    Para cada G_{ij} *em* R_t, $D_{ijknpq} \leftarrow Hungaro(E_{ijknpq})$
    Para cada $R_s \in T$ *e* $R_s \neq R_t$, *enviar* $Comp(D)_{ts}$
    Para cada G_{ij} *em* R_t, $D_{ijknpq} \leftarrow (D_{ijknpq} + D_{ijpqkn} + ... + D_{pqknij})/6$
    Para cada G_{ij} *em* R_t, $C_{ijkn} \leftarrow Hungaro(D_{ijkn})$
    Para cada $R_s \in T$ *e* $R_s \neq R_t$, *enviar* $Comp(C)_{ts}$
    Para cada G_{ij} *em* R_t, $C_{ijkn} \leftarrow (C_{ijkn} + C_{knij})/2$
    Para cada G_{ij} *em* R_t, $B_{ij} \leftarrow Hungaro(C_{ij})$
    Para cada $R_s \in T$ *e* $R_s \neq R_t$, *enviar* $Mens(B)_{ts}$
    $LB \leftarrow LB + Hungaro(B)$
    $cont \leftarrow cont + 1$
  fim
fim

Na versão distribuída, complementares que pertençam a conjuntos diferentes podem estar alocados em nós distintos. Em consequência disso, não é possível que os complementares compartilhem a mesma posição de memória. Estabelecemos, nesta versão, que a utilização da mesma posição de memória é feita apenas pelos complementares de um mesmo conjunto G_{ij}. Por causa desta restrição, não conseguimos uma grande redução de memória quando comparamos com as execuções feitas em Hahn *et al*. (2012). Na matriz C, os complementares ficam em conjuntos diferentes, portanto não podem ocupar a mesma posição na memória. Na matriz D, apenas 2 complementares podem ocupar a mesma posição da memória. E na matriz E apenas os 6 complementares de mesmo $\{ij\}$ compartilham posição.

Para a transferência de custos entre complementares que estejam em nós distintos, há a necessidade de troca de mensagens contendo seus custos. A cada ciclo do loop principal do Algoritmo 1 é necessária a troca dos complementares entre os nós, e isto é feito em 4 etapas. A primeira etapa consiste na troca, entre todos os nós, dos complementares da matriz E. O conjunto dos complementares armazenados em R_x, necessários em R_z e transferidos através de mensagens de R_x para R_z, denotamos como $Comp(E)_{xz}$. Nas 2 etapas seguintes há a troca dos complementares das matrizes D e C, envio e recebimento de $Comp(D)_{xz}$ e $Comp(C)_{xz}$ respectivamente. Na 4a. e última etapa, há a troca de coeficientes da matriz B, onde cada processo envia seus coeficientes. A matriz B não possui complementares. Definimos esta mensagem como $Mens(B)_{xz}$.

O tempo gasto na troca de mensagens $Comp(E)_{xz}$ não impacta na execução de instâncias de pequeno tamanho, como $N = 12$, mas alcança cerca de 70% do tempo total da execução em instâncias de tamanho $N = 30$.

Os coeficientes das matrizes podem ser do tipo de dados inteiro ou *float*. A escolha do tipo de dados influência na distribuição dos conteúdos dos coeficientes. Vemos isso ao final da etapa de "transferência de custos entre complementares", ver Seção 3. Logo após a média aritmética entre os complementeres do tipo *float*, todos passam a ter o mesmo valor. Já ao final da mesma etapa, quando no uso do tipo de dados inteiro, os complementares podem não ter o mesmo conteúdo, visto que se a média não for uma divisão inteira perfeita teremos um resto que deverá ser distribuído unidade a unidade pelos complementares, como nem todos complementares recebem uma unidade adicional, passam a diferir assim seus conteúdos. Optamos, na versão distribuída, em utilizar o tipo de dados *float*.

Segue a descrição do Algoritmo 1 a ser executado em cada processo R_t:

Linhas 2, 3 e 4 - Inicialização: $LB \leftarrow 0$, $B_{ij} \leftarrow 0 \; \forall \; (i,j)$, $C_{ijkn} \leftarrow f_{ik} \times d_{jn} \;\; \forall \;\; (i,j,k,n)$ com $i \neq k$ e $j \neq n$, $D_{ijknpq} \leftarrow 0 \;\; \forall \;\; (i,j,k,n,p,q)$ com $i \neq k \neq p$ e $j \neq n \neq q$, $E_{ijknpqgh} \leftarrow 0 \;\; \forall \;\; (i,j,k,n,p,q,g,h)$ com $i \neq k \neq p \neq g$ e $j \neq n \neq q \neq h$, $cont \leftarrow 1$, $lim \leftarrow$ total de iterações e $ótimo \leftarrow$ solução ótima conhecida ou maior valor positivo possível para aquela variável.

Linhas 5 e 6 - Transferência de custos entre complementares da matriz C: Para cada $R_s \in T$ e $R_s \neq R_t$, e para cada (i,j,k,n) tal que G_{ij} alocado em R_t e G_{kn} alocado em R_s, incluir os coeficientes C_{ijkn} em $Comp(C)_{ts}$. Enviar mensagem contendo $Comp(C)_{ts}$. Após receber mensagens de todos os outros nós, para cada G_{ij} alocado em R_t, $i < k$ e $j \neq n$, $C_{ijkn} \leftarrow (C_{ijkn} + C_{knij})/2$.

Limha 7 - Concentração de custos da matriz C para B: Para cada G_{ij} alocado em R_t, concentrar os coeficientes das submatrizes de C para B, ou seja, $B_{ij} \leftarrow Hungaro(C_{ij})$.

Linha 8 - Transferência de custos da matriz B: Para cada $R_s \in T$ e $R_s \neq R_t$, e para cada (i,j) tal que G_{ij} alocado em R_t, incluir os coeficientes B_{ij} em $Mens(B)_{ts}$. Enviar mensagem contendo $Mens(B)_{ts}$. Após receber as mensagens de todos os outros nós, transferir os coeficientes recebidos para a matriz B local.

Linha 9 - Concentração de custos da matriz B para LB: $LB \leftarrow LB + Hungaro(B)$.

Linha 10 - Início do loop principal:. A condição de término do loop está em alcançar o total de iterações previamente definido, $cont = lim$, ou o ótimo, $LB = ótimo$.

Linha 11 - Espalhamento de custos da matriz B para C: Para cada (i,j) tal que G_{ij} alocado

em R_t, espalhar B_{ij} pelas $(N-1)$ linhas da submatriz C_{ij}. Cada elemento de custo C_{ijkn} receberá o acréscimo de B_{ij} / $(N-1)$ $\forall$ $k \neq i$ e $n \neq j$.

Linha 12 - Espalhamento de custos da matriz C para D: Para cada (i,j,k,n) tal que G_{ij} alocado em R_t , $i \neq j$ e $k \neq n$, espalhar C_{ijkn} pelas $(N-2)$ linhas da submatriz D_{ijkn}. Cada elemento de custo D_{ijknpq} receberá o acréscimo de C_{ijkn} / $(N-2)$ $\forall$ $p \neq i,k$ e $q \neq j,n$. Os coeficientes D_{ijknpq} e D_{ijpqkn} ocupam a mesma posição da memória.

Linha 13 - Espalhamento de custos da matriz D para E: Para cada (i,j,k,n,p,q) tal que G_{ij} alocado em R_t , $i \neq j,p$ e $k \neq n,q$, espalhar D_{ijknpq} pelas $(N-3)$ linhas da submatriz E_{ijknpq}. Cada elemento de custo $E_{ijknpqgh}$ receberá o acréscimo de D_{ijknpq} / $(N-3)$ $\forall$ $g \neq i,k,p$ e $h \neq j,n,q$. Os coeficientes $E_{ijknpqgh}$,$E_{ijknghpq}$,$E_{ijpqkngh}$, $E_{ijpqghkn}$, $E_{ijghknpq}$ e $E_{ijghpqkn}$ ocupam a mesma posição da memória.

Linhas 14 e 15 - Transferência de custos entre complementares da matriz E: Para cada $R_s \in T$ e $R_s \neq R_t$, para cada (i,j,k,n,p,q,g,h) tal que G_{ij} alocado em R_t e $(G_{kn}, G_{pq}$ ou $G_{gh})$ alocado em R_s, incluir os coeficientes $E_{ijknpqgh}$ em $Comp(E)_{ts}$. Enviar mensagem contendo $Comp(E)_{ts}$. Após receber mensagens de todos os outros nós, para cada (i,j,k,n,p,q,g,h) tal que G_{ij} alocado em R_t e $i < k < p < g$ e $j \neq n \neq q \neq h$, $E_{ijknpqgh} \leftarrow E_{ijknghpq} \leftarrow E_{ijpqkngh} \leftarrow E_{ijpqghkn} \leftarrow E_{ijghknpq} \leftarrow E_{ijghpqkn} \leftarrow (E_{ijknpqgh} + E_{ijknghpq} + \ ... \ + E_{ghpqknij})/24$.

Linha 16 - Concentração de custos da matriz E para D: Para cada (i,j,k,n,p,q) tal que G_{ij} alocado em R_t, concentrar as submatrizes de E para D, ou seja, $D_{ijknpq} \leftarrow Hungaro(E_{ijknpq})$.

Linhas 17 e 18 - Transferência de custos entre complementares da matriz D: Para cada $R_s \in T$ e $R_s \neq R_t$, e para cada (i,j,k,n,p,q) tal que G_{ij} alocado em R_t e $(G_{kn}$ ou $G_{pq})$ alocado em R_s incluir os coeficientes D_{ijknpq} em $Comp(D)_{ts}$. Enviar mensagem contendo $Comp(D)_{ts}$. Após receber mensagens de todos os outros nós, para cada (i,j,k,n,p,q) tal que $G_{ij} \in R_t$, $i < k < p$ e $j \neq n \neq q$, $D_{ijknpq} \leftarrow D_{ijpqkn} \leftarrow (D_{ijknpq} + D_{ijpqkn} + D_{knijpq} + D_{knpqij} + D_{pqijkn} + D_{pqknij})/6$.

Linha 19 - Concentração de custos da matriz D para C: Para cada (i,j,k,n) tal que G_{ij} alocado em R_t, concentrar as submatrizes de D para C, ou seja, $C_{ijkn} \leftarrow Hungaro(D_{ijkn})$.

Linhas 20 e 21 - Transferência de custos entre complementares da matriz C: Para cada $R_s \in T$ e $R_s \neq R_t$, e para cada (i,j,k,n) tal que G_{ij} alocado em R_r e G_{kn} alocado em R_s, incluir os coeficientes C_{ijkn} em $Comp(C)_{ts}$. Enviar mensagem contendo $Comp(C)_{ts}$. Após receber mensagens de todos os outros nós, para cada (i,j,k,n) tal que G_{ij} alocado em R_t, $i < k$ e $j \neq n$, $C_{ijkn} \leftarrow (C_{ijkn} + C_{knij})/2$.

Linha 22 - Concentração de custos da matriz C para B: Para cada (i,j) tal que G_{ij} alocado em R_t, concentrar as submatrizes de C para B, $B_{ij} \leftarrow Hungaro(C_{ij})$.

Linha 23 - Transferência de custos da matriz B: Para cada $R_s \in T$ e $R_s \neq R_t$, e para cada (i,j) tal que G_{ij} alocado em R_t, incluir os coeficientes B_{ij} em $Mens(B)_{ts}$. Enviar mensagem contendo $Mens(B)_{ts}$. Após receber as mensagens de todos os outros nós, transferir os coeficientes recebidos para matriz B local.

Linha 24 - Concentração de custos da matriz B para LB: $LB \leftarrow LB + Hungaro(B)$.

Linha 25 - Fim do loop: Incrementar o $cont$ e retornar ao início do loop principal (linha 10).

5 Resultados alcançados

A aplicação foi implementada usando a liguagem C++ em conjunto com a biblioteca IntelMPI. Os experimentos foram executados no Supercomputador Netuno, ver Silva *et al.* (2011), um *cluster* composto de 256 nós. Cada nó contém dois processadores Intel Xeon E5430 2.66GHz Quad core com 12MB de cache L2 cada e 16 GB de memória RAM por nó,

Em cada nó é executado um processo de forma exclusiva com o objetivo de utilizar o máximo da memória disponível da máquina, e limitar contenções de recursos em função da concorrência. A mesma denominação R_t serve para identificar o processo ou o nó que o executa. Neste trabalho, adotamos o uso apenas de um núcleo por nó. Propostas com o uso dos outros núcleos, seja para

execução de threads ou para execução de outros processos no mesmo nó, serão abordadas em um próximo trabalho cujo objetivo é a busca da otimização desta versão distribuida.

Para avaliação preliminar do algoritmo distribuído proposto, o término da execução foi condicionado ao alcance do valor ótimo, este fornecido no início da execução, ou ao número total de iterações. Cada iteração corresponde a um ciclo do *loop* principal do algoritmo (ver Algoritmo 1). O valor pré-determinado para o total de iterações foi de 300 iterações. As instâncias de tamanho $N = 30$ tiveram seus testes interrompidos devido a eventualidades, como problemas técnicos ocorridos com o *cluster*. Nestes casos, o valor apresentado foi o obtido até aquele momento.

Tabela 1: Limites alcançados pela versão distribuída e as demais técnicas do site do QAPLIB

Instância	Ótimo	BV04	HH01	HZ07	Versão Distribuída		
					LB	*gap*	tempo(s)
had14	2724	**2724** *			**2724** *	-	79
had16	3720	3672	**3720** *	3719.1	**3720** *	-	742
had18	5358	5299	**5358** *	5357	**5358** *	-	6636
had20	6922	6811	**6922** *	6920	**6922** *	-	16118
kra30a	88900	86678	86247		**88424**	0.54%	196835
nug12	578	568	**578** *	577.2	**578** *	-	73
nug15	1150	1141	**1150** *	1149.1	**1150** *	-	371
nug16a	1610	1597			**1610** *	-	1132
nug16b	1240	1219.0			**1240** *	-	1326
nug18	1930	1892.9		**1930** *	**1930** *	-	7353
nug20	2570	2506	2508	2568.1	**2570** *	-	27605
nug22	3596	3511.9	3511	3594.04	**3596** *	-	41616
nug24	3488	3397.0			**3478**	0.28%	171216
nug25	3744	3620.8			**3689**	1.44%	125012
nug28	5166	5018.3			**5038**	2.48%	171783
nug30	6124	5934	5770		**5940**	3.00%	229583
rou15	354210	350207	**354210** *	**354210** *	**354210** *	-	321
rou20	725520	695123	699390	**725314.4**	720343	0.71%	44694
tai15a	388214	377111.1			**388214** *	-	301
tai17a	491812	476516.6			**491812** *	-	1624
tai20a	703482	671685	675870	**703482** *	698549	0.70%	45720
tai25a	1167256	1112862	1091653		**1122090**	3.87%	98743
tai30a	1818146	1706875	1686290		**1724509**	5.15%	112085
tho30	149936	142814	136708		**142990**	4.63%	145713
chr18a	11098	**11098** *			**11098** *	-	1892
chr20a	2192	2188.1			**2192** *	-	5914
chr20b	2298	2295.0			**2298** *	-	3708
chr22a	6156	6154.2			**6156** *	-	5321

A Tabela 1 apresenta os resultados alcançados para diferentes instâncias e tamanhos do QAP. A maior parte das instâncias desta tabela pode ser encontrada no site do QAPLIB. Nós acrescentamos outras instâncias que são: had14, nug16a, nug16b, nug24, nug25, tai15a, tai17a, chr18a, chr20a, chr20b e chr22a.

Na primeira coluna da Tabela 1 estão as instâncias utilizadas e suas dimensões, por exemplo, a *nug20* corresponde a instância *nug* de Nugent *et al.* (1968) com tamanho $N = 20$. Na segunda coluna estão os valores ótimos de cada instância. Na terceira coluna (BV04) estão os resultados obtidos

pelo relaxamento *lift-and-bound* proposto por Burer e Vandenbussche (2006). Na quarta coluna (HH01), os resultados obtidos pelo algoritmo dual Hahn-Hightower RLT2 aplicado ao problema de QAP por Adams *et al*. (2007). Na quinta coluna (HZ07), os resultados obtidos pelo algoritmo dual de subida Hahn-Zhu RLT3 de Hahn *et al*. (2012). Na sexta coluna, os resultados obtidos na versão RLT3 distribuída proposta neste artigo. Na sétima coluna, os *gaps* dos resultados da versão distribuída em relação ao ótimo, e na última coluna, o tempo total de execução em segundos.

Ainda na Tabela 1, observa-se que os resultados que alcançaram a solução ótima tem o indicativo (*) ao lado do valor, e os melhores resultados na instância foram destacados em negrito.

A Tabela 2 apresenta os tempos de execução, a quantidade de nós e o tamanho das mensagens da versão distribuída para a instâncias *nug* que possui uma quantidade maior de tamanhos dentre as apresentadas na Tabela 1. Na primeira coluna, temos as instâncias. Na segunda coluna, o tempo total em segundos na execução da aplicação. Na terceira coluna, a quantidade de nós participantes na versão distribuída. Na quarta e quinta coluna desta tabela, o tamanho médio de cada mensagem de troca de complementares das maiores matrizes, $Comp(D)$ e $Comp(E)$ respectivamente. Na sexta coluna temos o número de mensagens trocadas em cada etapa de transferência de complementares ou coeficientes. A cada ciclo do loop principal do algoritmo temos 4 etapas, uma para cada matriz: B, C, D e E. E na útlima coluna, o total de dados trocados pelos nós por ciclo do loop principal.

Tabela 2: Tempo, quantidade de nós e tamanho das mensagens entre os nós na versão distribuída

Instância	Versão Distribuída					
	Tempo (s)	Nós	$Comp(D)$ (Kbytes)	$Comp(E)$ (Mbytes)	Quant. Msg	Total (Gbytes)
nug12	73	4	411.0	14.4	12	0.17
nug15	371	9	355.7	23.9	72	1.70
nug16a	1132	8	654.5	51.1	56	2.83
nug16b	1326	8	654.5	51.1	56	2.83
nug18	7353	9	1135.2	119.2	72	8.46
nug20	27605	20	423.8	58.9	380	22.0
nug22	41616	22	601.6	105.0	462	47.6
nug24	171216	24	925.6	196.7	552	106.5
nug25	125012	25	1146.7	267.4	600	157.3
nug28	171783	49	588.0	178.5	2352	411.3
nug30	229583	100	219.5	97.7	9900	946.6

6 Conclusão e trabalhos futuros

A versão distribuída alcançou ótimos resultados em relação aos demais, ver Tabela 1, e que permitiu a execução baseada na formulação RLT3 de instâncias como as de tamanho $N = 28$ e $N = 30$, antes proibitivas devido ao tamanho necessário de memória RAM. Observamos que o algoritmo proposto gera os melhores limites conhecidos em 26 das 28 instâncias testadas, sendo que 18 destes limites alcançam o ótimo. Os bons resultados alcançados se devem ao fato da aplicação ser distribuída e uma combinação do uso da média dos complementares com o tipo de dados *float*.

Observamos na Tabela 2 que o volume de dados transferidos através de mensagens cresce excessivamente conforme o aumento do tamanho da instância, influenciando fortemente no tempo total de execução. Uma instância com tamanho $N = 30$ tem cerca de 70% de seu tempo gasto apenas na troca de mensagens. Técnicas de compactação aplicadas aos conteúdos das mensagens estão sendo avaliadas e os resultados serão apresentados em trabalho posterior.

Instâncias maiores $(N > 30)$ serão avaliadas conforme a diaponibilidade do *cluster* e apresentaremos seus resultados em trabalho posterior. Ainda em trabalhos futuros, otimizaremos a

versão distribuída e possivelmente iremos utilizar placas gráficas (*GPUs*) para o processamento do algoritmo húngaro com o objetivo de reduzir o tempo de execução.

Referências

Adams, W. P. e Sherali, H. D. (1986), A tight linearization and an algorithm for zero-one quadratic programming problems. *Manage. Sci.*, v. 32, n. 10, p. 1274–1290.

Adams, W. P., Guignard, M., Hahn, P. M. e Hightower, W. L. (2007), A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *European Journal of Operational Research*, v. 180, n. 3, p. 983–996.

Burer, S. e Vandenbussche, D. (2006), Solving lift-and-project relaxations of binary integer programs. *SIAM Journal on Optimization*, v. 16, p. 726–750.

Dickey, J. e Hopkins, J. (1972), Campus building arrangement using topaz. *Transportation Research*, v. 6, p. 59–68.

Elshafei, A. N. (1977), Hospital layout as a quadratic assignment problem. *Journal of The Operational Research Society*, v. 28, n. 1, p. 167–179.

Francis, R. L., McGinnis, L. F. e White, J. A. *Facility layout and location: an analytical approach*. Prentice Hall, Englewood Cliffs, NJ, 1992.

Gilmore, P. C. (1962), Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society of Industrial and Applied Mathematics*, v. 10, n. 2, p. 305–313.

Hahn, P. M. e Grant, T. (1998), Lower bounds for the quadratic assignment problem based upon a dual formulation. *Oper. Res.*, v. 46, n. 6, p. 912–922.

Hahn, P. M., Zhu, Y.-R., Guignard, M., Higthower, W. L. e Saltzman, M. (2012), A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS Journal on Computing*, v. 24, n. 2, p. 202–209.

Heffley, D. R. (1980), Decomposition of the koopmans-beckmann problem. *Regional Science and Urban Economics*, v. 10, n. 4, p. 571–580.

Hubert, L. *Assignment methods in combinatorial data analysis*. M. Dekker, New York, N.Y. ISBN 0824776178 9780824776176, 1986.

Koopmans, T. C. e Beckmann, M. (1957), Assignment problems and the location of economic activities. v. 25, n. 1, p. 53–76.

Lawler, E. L. (1963), The quadratic assignment problem. *Management Science*, v. 9, n. 4, p. 586–599.

Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P. M. e Querido, T. (2007), A survey for the quadratic assignment problem. *European Journal of Operational Research*, v. 176, n. 2, p. 657–690.

Munkres, J. (1957), Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, v. 5, n. 1, p. 32–38.

Nugent, C. E., Vollmann, T. E. e Ruml, J. (1968), An experimental comparison of techniques for the assignment of facilities to locations. v. 16, n. 1, p. 150–173.

Silva, G. P., Correa, J., Bentes, C., Guedes, S. e Gabioux, M. (2011), The experience in designing and building the high performance cluster netuno. *Computer Architecture and High Performance Computing, Symposium on*, v. 0, p. 144–151.