# UM NOVO ALGORITMO HÍBRIDO PARA RESOLUÇÃO DO PROBLEMA GENERALIZADO DE ATRIBUIÇÃO

**Robert Fabrício Subtil[1], Sérgio Ricardo de Souza[1]**

[1]Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG
Av. Amazonas, 7675 – Belo Horizonte, MG, CEP 30.510-000
rsubtil@yahoo.com.br, sergio@dppg.cefetmg.br

**Resumo:** Este trabalho apresenta um algoritmo híbrido, que combina os procedimentos heurísticos *Simulated Annealing*, Descida em Vizinhança Variável, Busca Tabu com Relaxação Adaptativa e Reconexão por Caminhos para resolver o Problema Generalizado de Atribuição. Este problema consiste em encontrar o menor custo para a atribuição de *n* tarefas a *m* agentes, sendo que cada tarefa deve ser atribuída a apenas um agente, sujeito à capacidade dos mesmos. Foram realizados testes computacionais com instâncias de tamanho médio, os quais mostraram que a abordagem proposta conseguiu encontrar alguns resultados melhores em relação a outras estratégias da literatura.

**Palavras-chave:** Problema Generalizado de Atribuição, *Simulated Annealing*, Descida em Vizinhança Variável, Busca Tabu, Relaxação Adaptativa, Reconexão por Caminhos.

**Abstract:** This paper presents the a hybrid algorithm, which combines the heuristics procedures Simulated Annealing, Variable Neighborhood Descent, Tabu Search with Adaptive Relaxation and Path Relinking to solve the Generalized Assignment Problem. This problem consists in finding the lowest cost to the assignment of *n* tasks to *m* agents, and each task should be assigned to only one agent, subject to their capabilities. We performed computational tests with instances of medium size, which showed that the approach could find some better results compared with other strategies proposed in the literature.

**Keywords:** Generalized Assignment Problem, Simulated Annealing, Variable Neighborhood Descent, Tabu Search, Adaptive Relaxation, Path Relinking.

## 1 Introdução

O Problema Generalizado de Atribuição (PGA) é um problema combinatório que consiste na atribuição de *n* diferentes tarefas a *m* diferentes agentes, assegurando que cada tarefa seja atribuída a um único agente cuja capacidade deve ser respeitada (Balachandran, 1976). Aplicações deste problema surgem em diversos contextos práticos, tais como a atribuição de tarefas em redes de computadores (Balachandran, 1976), localização de facilidades (Ross & Soland, 1977), carregamento de contêineres (Hung & Fisk, 1979), escalonamento de máquinas paralelas (Lenstra *et al*., 1990), roteamento de veículos (Fisher & Jaikumar, 2006), comparação de estoque/documentos (Dawande & Kalagnamam, 1998), escalonamento de recursos (Mazzola *et al*., 1989) e escalonamento de multiprocessadores (Turek *et al*., 1992).

O PGA é um problema NP-Difícil (Sahni & Gonzalez, 1976, Chu & Beasley, 1997), o que significa que ainda não existem algoritmos de complexidade polinomial para resolvê-lo na otimalidade. A importância do PGA e as dificuldades relacionadas à busca de suas soluções têm justificado o emprego de vários métodos de otimização para lidar com ele: heurísticas baseadas em teoria de conjuntos (Cattrysse *et al*., 1994, Wilson, 2005), heurísticas aproximadas desenvolvidas com base na estrutura do problema (Nauss, 2003), Path-relinking (Alfandari *et al.*, 2001, Yagiura *et al.*, 2006), relaxação Lagrangeana (Narciso & Lorena, 1999, Guignard & Rosenwein, 1989), Busca Tabu (Osman, 1995, Diaz & Fernandez, 2001, Laguna *et al.*, 1995), *Simulated Annealing* (Osman, 1995), Algoritmos Genéticos (Chu & Beasley, 1997) e muitos outros.

Para a resolução do PGA, este trabalho apresenta um algoritmo heurístico híbrido que combina os métodos *Simulated Annealing*, Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND), Busca Tabu com Relaxação Adaptativa e Reconexão por Caminhos. O algoritmo foi testado em instâncias disponíveis na literatura, bem como teve seus resultados comparados com outras abordagens propostas ao problema.

Na próxima seção é apresentada a formulação do problema, na Seção 3 é descrito o algoritmo proposto para resolver o PGA. Os resultados são apresentados na Seção 4 e a Seção 5 conclui o trabalho e aponta os trabalhos futuros.

## 2 O Problema Generalizado de Atribuição

Dado A o conjunto de agentes, $T$ o conjunto de tarefas, $c_{ij}$ é o custo de atribuir a tarefa $j \in T$ ao agente $i \in A$ , $a_{ij}$ são os recursos necessários ao agente $i \in A$ para desempenhar a tarefa $j \in T$, $b_i$ é a capacidade do agente $i \in A$ e $x_{ij}$ é a variável binária de decisão a qual assume o valor "1" se a tarefa $j \in T$ é atribuída ao agente $i \in A$ ou "0" em caso contrário. Desta maneira, o PGA pode ser apresentado como (Yagiura *et al.*, 2006):

$$f_C(x) = \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{T}} c_{ij} \cdot x_{ij} \qquad (1)$$

$$\text{subject to :} \begin{cases} \sum_{j \in \mathcal{T}} a_{ij} \cdot x_{ij} \leq b_i & , \forall i \in \mathcal{A} \\ \sum_{i \in \mathcal{A}} x_{ij} = 1 & , \forall j \in \mathcal{T} \\ x_{ij} \in \{0,1\} & , \forall i \in \mathcal{A},\ \forall j \in \mathcal{T} \end{cases} \qquad (2)$$

Nesta formulação, a Eq. (1) é a função de custo que deve ser minimizada. Na Eq. (2), a primeira restrição assegura que a capacidade dos agentes não é violada, a segunda restrição

assegura que cada tarefa é atribuída a apenas um agente e a terceira restrição define que cada variável $x_{ij}$ é binária.

## 3 Metodologia

Para resolução do PGA propõe-se neste trabalho um algoritmo heurístico híbrido combinando os métodos *Simulated Annealing* (SA), Descida em Vizinhança Variável (*Variable Neighborhood Descent* - VND), Busca Tabu com Relaxação Adaptativa (BTRA) e Reconexão por Caminhos (RC).

No algoritmo proposto, denominado BTRA-RC, a solução inicial é construída pela aplicação do procedimento heurístico SA. Esse procedimento inicia sua busca com uma solução aleatória e aplica o procedimento VND para fazer buscas locais em cada solução de melhora encontrada durante o processo de otimização. Construída a solução inicial, ela é refinada pelo procedimento BTRA. Durante sua execução, a BTRA armazena um conjunto de soluções elite e aplica uma estratégia de intensificação com o procedimento RC. Este último procedimento é aplicado para fazer reconexões nos caminhos entre uma solução qualquer do Conjunto Elite e a melhor solução encontrada durante *N* iterações em que este conjunto ficou sem atualizações.

A seção 3.1 descreve a forma de representação da solução do problema; na seção 3.2 são estabelecidas as estruturas de vizinhança; a seção 3.3 apresenta a função de avaliação; as seções 3.4, 3.5 e 3.6 descrevem o procedimento de geração da solução inicial; na seção 3.7 é descrita a aplicação da Busca Tabu com Relaxação Adaptativa ao PGA e a seção 3.8 apresenta o procedimento de Reconexão por Caminhos e sua aplicação ao problema; na seção 3.9 é introduzido os conceitos em torno da Relaxação Adaptativa.

### 3.1 Representação

Uma solução do PGA foi codificada como um vetor de inteiros $1 \times |T|$, em que $|T|$ é o número de tarefas. Cada posição desse vetor pode assumir valores inteiros no intervalo $[1, |A|]$, sendo $|A|$ o número de agentes. Um exemplo de codificação para uma instância PGA $4{\times}10$ , isto é, com 4 agentes e 10 tarefas, é ilustrado na Figura 1.

| 3 | 4 | 4 | 2 | 1 | 4 | 3 | 2 | 2 | 4 |
|---|---|---|---|---|---|---|---|---|---|

**Figura 1:** Exemplo de representação: Nesta solução candidata a tarefa 5 é atribuída ao agente A1, as tarefas 4, 8 e 9 são atribuídas ao agente A2, as tarefas 1 e 7 são atribuídas ao agente A3 e as tarefas 2, 3, 6 e 10 são atribuídas ao agente A4.

Observe que, pela representação adotada, o segundo conjunto de restrições (2) é automaticamente satisfeito, já que cada célula do vetor solução *s* recebe um único agente.

### 3.2 Estruturas de Vizinhança

Foram usados dois tipos de movimentos para explorar o espaço de soluções do problema. A Figura 2 ilustra esses movimentos.

O primeiro movimento consiste em substituir um agente alocado a uma tarefa por outro agente qualquer que não faz parte da solução. O conjunto de todas as soluções *s'* geradas a partir de $s$ através de movimentos de substituição define a vizinhança $N^{(S)}(s)$.

O segundo movimento é caracterizado pela troca de agentes entre duas tarefas. O conjunto de todas as soluções *s'* geradas a partir de $s$ através de movimentos de troca define a vizinhança $N^{(T)}(s)$.

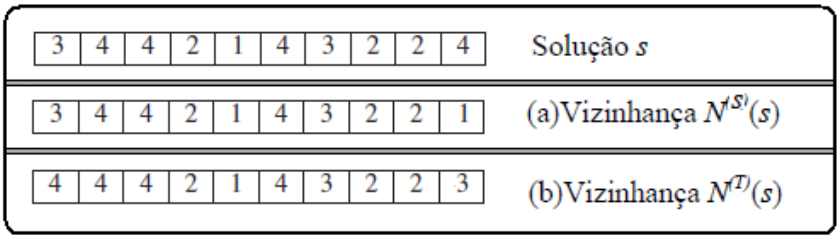


**Figura 2**: Estruturas de vizinhança: Neste exemplo com uma instância PGA A4×10 (4 agentes × 10 tarefas), o movimento (a) de substituição resulta na solução *s'* tendo o agente A4 substituído pelo agente A1 na tarefa 10. O movimento (b) de troca resulta na solução *s'* com as trocas dos agentes A3 e A4 anteriormente alocados às tarefas 1 e 10, respectivamente

A vizinhança $N(s)$ de uma dada solução $s$ é definida pela união dessas duas vizinhanças, isto é, $N(s) = N^{(S)}(s) \cup N^{(T)}(s)$.

### 3.3 Função de Avaliação

Uma solução $s$ é avaliada com base na função $f_c(s)$, dada pela Eq. (3). Essa função, que deve ser minimizada, considera os custos das atribuições dos agentes às tarefas bem como penaliza o uso de recursos indisponíveis de cada agente.

$$f_c^m(s) = f_c(s) + r_{inv} \qquad (3)$$

Na Eq. (3), $f_c(s)$ é calculada conforme a Eq. (1) e $r_{inv} = \sum_{i \in A} \rho_{i \times max\{0, \sum_{j \in T}(a_{ij} \cdot s_j) b_i\}}$. A segunda parcela dessa função de avaliação penaliza as soluções em que a capacidade de cada agente i foi ultrapassada. Essa quantidade de violação referente ao primeiro conjunto de restrições (2) é multiplicada por um fator $\rho_i$, dado por $\rho_i = \sum_{j \in T} a_{ij}$.

### 3.4 Geração da Solução Inicial

A solução inicial é obtida pela aplicação do procedimento *Simulated Annealing* (descrito na seção 3.5) a uma solução construída aleatoriamente. O pseudocódigo do procedimento de construção é apresentado no Algoritmo 1.

**Algoritmo 1:** Procedimento gera_solucao_inicial()

```
s ← ∅;
para j = 1, 2, ··· , |T| faça
    s[j] ← agente i ∈ A escolhido aleatoriamente;
fim
T_0 ← TemperaturaInicial();
s ← SA(f_C^m(s), s, T_0);
Retorna s;
```

Observe que antes da aplicação do *Simulated Annealing* à solução aleatória é determinada a temperatura inicial $T_0$ deste procedimento. A forma como essa temperatura é calculada é descrita na próxima seção.

### 3.5 *Simulated Annealing*

O *Simulated Annealing* (SA) é um método de busca local probabilístico, que se fundamenta em uma analogia com a termodinâmica ao simular o resfriamento de um conjunto de átomos aquecidos, operação conhecida como recozimento.

O procedimento principal do SA consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho *s'* da solução corrente *s*. Os parâmetros de controle do procedimento são a razão de resfriamento $\alpha$, o número de iterações para cada temperatura (*SAMax*) e a temperatura inicial $T_0$. Mais detalhes sobre o método podem ser vistos em Kirkpatrick *et al*. (1983).

Neste trabalho, o algoritmo SA básico trabalha em conjunto com o procedimento VND, cuja descrição é feita na seção 3.6. Mais especificamente, o VND é acionado sempre que o SA melhora a solução global.

Os vizinhos aleatórios foram gerados com o movimento de substituição. A temperatura inicial $T_0$ foi obtida de forma auto-adaptativa por meio de simulação, utilizando o mesmo princípio do método *Simulated Annealing*. A descrição dessa forma de calcular a temperatura está descrita em Souza, 2009. Os parâmetros usados foram: taxa de aumento da temperatura $\beta = 10\%$ , taxa mínima de aceitação de soluções vizinhas $\gamma = 5\%$ e temperatura de partida $T_0 = 10$.

### 3.6 Descida em Vizinhança Variável (VND)

A Descida em Vizinhança Variável (*Variable Neighborhood Descent* – VND), proposto por Mladenovic & Hansen (1997), é um procedimento de busca local que explora o espaço de soluções por meio da troca sistemática de estruturas de vizinhança.

O procedimento parte de uma solução e de um número pré-definido de estruturas de vizinhança em uma dada ordem de exploração. A busca inicia na primeira vizinhança $N^{(1)}$ . Para cada vizinhança $N^{(k)}$ é procurada a melhor solução vizinha da solução corrente. Se o vizinho encontrado for melhor que a solução corrente, então esse vizinho passa a ser a nova solução corrente e o procedimento continua a exploração do espaço de soluções usando a primeira estrutura de vizinhança; caso contrário, o procedimento passa para a próxima vizinhança. O procedimento é interrompido quando é atingida a última estrutura de vizinhança sem melhora na solução corrente.

Para a resolução do PGA, foram utilizadas três estruturas de vizinhança para o VND: substituição do agente de uma tarefa, substituição de dois agentes de duas tarefas e a troca dos agentes entre duas tarefas. A ordem descrita também representa a sequência usada de vizinhanças na exploração do espaço de soluções.

### 3.7 Busca Tabu com Relaxação Adaptativa

A Busca Tabu (Glover, 1986; Hansen, 1986) é um procedimento de busca local que explora o espaço de soluções utilizando uma estrutura de memória. Ele consiste em avaliar a vizinhança de uma solução e mover para o seu melhor vizinho. Após este movimento, o vizinho ou atributos que o melhor represente são armazenados em uma estrutura de memória, chamada lista tabu. Esta estrutura é utilizada para impedir por um tempo determinado o retorno a soluções previamente visitadas, diminuindo a possibilidade de o algoritmo ficar preso em ótimos locais. O método também conta com uma função de aspiração, na qual movimentos com *status* tabu podem ser aceitos se a nova solução produzida satisfizer um determinado critério.

Neste trabalho foi implementada uma matriz tabu $n \times m$ , em que $n$ é número de agentes e $m$ a quantidade de tarefas, para armazenar os movimentos proibidos. O tempo de duração tabu é sorteado dentro de uma faixa $[T_{min}, T_{max}]$ em que $T_{min} = 0{,}2 \times n - 7$ e $T_{max} = 0{,}2 \times n + 7$.

Durante a execução da Busca Tabu, um conjunto de soluções de boa qualidade é formado, sendo este conjunto conhecido na literatura como Conjunto Elite (CE). Esse conjunto possui tamanho fixo e para fazer parte dele a solução candidata deve obedecer a um dos dois seguintes critérios: 1) ser melhor que a melhor solução do CE; 2) ser melhor que a pior solução do CE e ainda se diferenciar de todas as outras soluções em um certo percentual dos atributos. Tal conjunto é utilizado no algoritmo proposto para viabilizar a estratégia de intensificação feita pela Reconexão por Caminhos, cujo procedimento encontra-se descrito na seção 3.8.

Adotou-se o critério de aspiração por objetivo global. Dessa forma, uma solução *s* tabu é aceita se seu valor for menor que a melhor solução global.

O pseudocódigo da Busca Tabu aplicada ao PGA é apresentado no Algoritmo 2. Os parâmetros do método são o número de iterações sem melhora (*BTmax*), o tamanho do conjunto elite (|CE|), o número de iterações sem atualização do CE (n*MaxIterSemElite*) e os limites inferior e superior da duração tabu, $T_{min}$ e $T_{max}$, respectivamente.

### 3.8 Reconexão por Caminhos

A Reconexão por Caminhos – RC (*Path Relinking*) é uma estratégia que faz um balanço entre intensificação e diversificação (Rosseti, 2003). Foi proposta por Glover (1996) para ser utilizada em conjunto com a Busca Tabu. Considerando um par de soluções, sendo uma delas a solução base e a outra solução guia, o objetivo deste método é partir da solução base e caminhar para a solução guia através da inserção gradativa de atributos da solução guia na solução base.

Neste trabalho, tal método é utilizado em dois momentos. No primeiro, a RC é acionada a cada *nMaxIterSemElite* iterações sem atualização do CE. Nesta situação, o par de soluções a ser considerado é formado pela melhor solução encontrada dentro desse intervalo sem melhoria e alguma solução aleatoriamente selecionada do CE. A RC é aplicada em um segundo momento como estratégia de pós-otimização, ao término da Busca Tabu. Neste caso, a Reconexão por Caminhos é aplicada a todos os pares possíveis de soluções do CE, sendo retornada a melhor solução encontrada. Em ambos os momentos, a Reconexão é aplicada de forma bidirecional, isto é, tanto da melhor solução para a pior quanto da pior para a melhor dentre o par de soluções do conjunto elite.

**Algoritmo 2:** TABU($s$, $f_C^m()$, $N()$, *CE*, *BTmax*,$nMaxIterSemElite$,$T_{\min}$, $T_{\max}$)

```
início
    v* ← v ;                                              // Melhor solução
    T ← ∅;                                                    // Lista Tabu
    IterCorrente ← 0 ;                                  // Iteração corrente
    MelhorIter ← 0 ;                          // Iteração da melhor solução
    nIterSemElite ← 0 ;          // Iterações sem atualização do CE
    enquanto (IterCorrente − MelhorIter ≤ BTmax) faça
        Selecione a vizinhança corrente CN, tal que CN ∈ {N^(R)(s), N^(T)(s)};
        Defina um subconjunto V ⊆ CN;
        melhorMovimento ← movimentoRandomico(V);
        melhorCustoDinamico ← ∞;
        para todo Movimento m ∈ V faça
            se ( f(s ⊕ m) < f(s*) ) então
                melhorMovimento ← m;
                interromper;
            senão
                se (m ∉ ListaTabu) então
                    se ( f(s ⊕ m) < f(s) ) então
                        melhorMovimento ← m;
                        interromper;
                    senão
                        se (CustoPorPesosDinamicos(s ⊕ m) < melhorCustoDinamico)
                            então
                            melhorMovimento ← m;
                            melhorCustoDinamico ← CustoPorPesosDinamicos(s ⊕ m);
                        fim se
                    fim se
                fim se
            fim se
        fim para
        Atualize a matriz de duração tabu T;
        s ← s';
        Atualize o Conjunto Elite CE;
        se Conjunto Elite não foi atualizado então
            nIterSemElite ← nIterSemElite + 1;
            se nIterSemElite = nMaxIterSemElite então
                Selecione aleatoriamente uma solução do Conjunto Elite;
                Aplique o método de Reconexão por Caminhos entre essa solução e a
                melhor encontrada enquanto nIterSemElite < nMaxIterSemElite;
                Atualize s após a saída do método Reconexão por Caminhos;
                nIterSemElite ← 0;
            senão
                Atualize a melhor solução encontrada enquanto nIterSemElite <
                nMaxIterSemElite;
            fim
        fim
        se (f_C^m(s) < f_C^m(s*) então
            s* ← s;
            MelhorIter ← IterCorrente;
        fim
        IterCorrente ← IterCorrente + 1;
        se IteracaoAtualizacao() então AtualizarPesosDinamicos();
    fim
    retorne s*;
fim
```

### 3.9 Relaxação Adaptativa

Após dez iterações sem melhora na solução global durante a fase de Busca Tabu, o algoritmo proposto aplica um mecanismo de relaxação adaptativa. O objetivo desse mecanismo é o de redirecionar a busca para outras regiões mais promissoras do espaço de soluções. O mecanismo adotado é o mesmo de Schaerf (1996), em que os pesos atribuídos para cada fonte de inviabilidade são ajustados dinamicamente, como proposto em Gendreau *et al*. (1994). Dessa forma, o peso de inviabilidade $\rho_i$ da Eq. (3) é multiplicado por um fator $\sigma_i$ que varia de acordo com o seguinte esquema:

1. No início da busca $\sigma_i = 1$;
2. A cada $r$ movimentos:
    a. Se todas as $r$ soluções visitadas são factíveis então $\sigma_i = \sigma_i / \gamma$;
    b. Se todas as $r$ soluções visitadas são infactíveis então $\sigma_i = \sigma_i \times \gamma$;
    c. Se algumas soluções são factíveis e outras são infactíveis $\sigma_i$ permanece inalterado;

O parâmetro $\gamma$ é aleatoriamente escolhido, a cada $r$ movimentos, no intervalo [1,8; 2,2], conforme proposto por Schaerf (1996).

O valor de $\sigma_i$ é limitado por duas constantes $\sigma_{min}$ e $\sigma_{max}$, de forma a evitar que a relaxação adaptativa aumente ou diminua indefinidamente o peso dado à violação de capacidade de um agente.

## 4 Resultados Computacionais

Para a realização dos experimentos computacionais, foi utilizado um conjunto de 18 instâncias dos tipos C, D e E obtidos da literatura em http://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/. Tais instâncias são consideradas de tamanho médio e possuem as dimensões de 5, 10 e 20 agentes, com 100 e 200 tarefas.

Os parâmetros adotados no algoritmo BTRA-RC proposto foram: Número de iterações sem melhora (*BTmax*) = 2000, número máximo de iterações sem atualização no Conjunto Elite (*nMaxIterSemElite*) = 0,20 × *BTmax*, tamanho do Conjunto Elite (CE) = 5; taxa de resfriamento do *Simulated Annealing* ($\alpha$) = 0,998; número de iterações feitas em cada temperatura (*SAmax*) = $k \times m \times n$, sendo a constante $k$ = 3 obtida experimentalmente. A Relaxação Adaptativa é acionada após $u$ = 10 iterações sem melhora da Busca Tabu e usa os seguintes parâmetros: $r$ = 10, $\sigma_{min}$ = 0,0001, $\sigma_{max}$ = 5 e $\gamma \in$ [1,8; 2,2].

Na Tabela 1 são apresentados os resultados da comparação entre os algoritmos BTRA-RC, proposto neste trabalho, e os algoritmos BTRC (Subtil *et al.*, 2010), PREC (Yagiura *et al*., 2006), CB (Chu & Beasley, 1997), TSBB (Woodcock & Wilson, 2010), Bvdsj (Yagiura *et al*., 1998), RA (Amini & Racer, 1995), LKGG (Laguna *et al*., 1995), DF (Diaz & Fernandez, 2001). Nesta tabela os melhores resultados, isto é, os com os menores valores para a função de custo dada pela Eq. (1) estão destacados em negrito.

**Tabela 1:** Comparação de resultados entre BTRA-RC e algoritmos da literatura

| Tipo | Agentes | Tarefas | BTRC | BTRA-RC | TSBB | PREC | Bvds-j | RA | LKGG | CB | DF |
|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 5 | 100 | 1937 | **1931** | **1931** | **1931** | **1931** | 1938 | **1931** | **1931** | **1931** |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 100 | 1403 | 1403 | **1402** | **1402** | 1403 | 1405 | 1403 | 1403 | **1402** |
| | 20 | 100 | 1250 | 1248 | **1243** | **1243** | 1244 | 1250 | 1245 | 1244 | **1243** |
| | 5 | 200 | 3460 | 3459 | **3456** | **3456** | 3457 | 3469 | 3457 | 3458 | 3457 |
| | 10 | 200 | 2820 | 2818 | **2806** | 2807 | 2808 | 2835 | 2812 | 2814 | 2807 |
| | 20 | 200 | 2399 | 2400 | **2391** | **2391** | 2400 | 2419 | 2396 | 2397 | **2391** |
| D | 5 | 100 | 6403 | 6394 | 6356 | **6353** | 6362 | ND | 6386 | 6373 | 6357 |
| | 10 | 100 | 6430 | 6425 | 6366 | 6356 | 6370 | 6532 | 6406 | 6379 | **6355** |
| | 20 | 100 | 6317 | 6318 | 6254 | **6211** | 6245 | 6428 | 6297 | 6269 | 6220 |
| | 5 | 200 | 12816 | 12802 | 12745 | **12744** | 12755 | ND | 12788 | 12796 | 12747 |
| | 10 | 200 | 12596 | 12570 | 12456 | **12438** | 12473 | 12799 | 12537 | 12601 | 12457 |
| | 20 | 200 | 12468 | 12449 | 12332 | **12269** | 12318 | 12665 | 12436 | 12452 | 12351 |
| E | 5 | 100 | 12748 | 12696 | **12681** | **12681** | 12682 | 12917 | 12687 | ND | 12681 |
| | 10 | 100 | 11772 | 11622 | 11584 | **11577** | 11599 | 12047 | 11641 | ND | 11581 |
| | 20 | 100 | 8646 | 8569 | 8479 | **8444** | 8484 | 9004 | 8522 | ND | 8460 |
| | 5 | 200 | 25181 | 24975 | 24933 | 24933 | 24933 | 25649 | 25147 | ND | **24931** |
| | 10 | 200 | 23609 | 23366 | **23307** | 23310 | 23348 | 24717 | 23567 | ND | 23318 |
| | 20 | 200 | 23050 | 22602 | 22393 | **22379** | 22437 | 24117 | 22659 | ND | 22422 |

A Tabela 1 mostra que o algoritmo proposto conseguiu encontrar resultados iguais e até melhores que outras abordagens da literatura, os quais estão destacados em vermelho na tabela. Para todas as instâncias, o BTRA-RC foi superior ao RA, com destaque nas instâncias dos grupos D e E, as quais são mais complexas (Yagiura *et* al., 2006). Nessas instâncias, o algoritmo proposto apresentou uma melhora significativa em relação ao algoritmo RA. O mesmo pode ser observado ao comparar com o método BTRC, exceto para as instâncias C20x200 e D20x100, nas quais o BTRA-RC não conseguiu alcançar resultados melhores. O algoritmo BTRA-RC alcançou e, em alguns casos, também superou alguns resultados dos algoritmos BVDSj, LKGG e CB.

O algoritmo proposto encontrou o melhor resultado conhecido da literatura para a instância C5x100 e encontrou valores próximos aos melhores conhecidos para as demais instâncias do tipo C.

Como experimento adicional, foi aplicado um teste de probabilidade empírica, seguindo a metodologia de Aiex *et al*. (2002), para verificar a capacidade dos algoritmos BTRC e BTRA-RC de alcançarem um valor alvo. Para execução dos experimentos, utilizou-se as instâncias C5x100 e E20x200, tendo como alvo os valores de custo 1945 e 23386, respectivamente. Cada algoritmo foi executado 100 vezes para cada instância, sendo que em cada rodada ele era interrompido após alcançar o alvo. Não foram permitidos tempos de execução repetidos; assim, os tempos repetidos foram descartados e uma nova execução foi feita. Após determinados os tempos para as 100 execuções, estes foram ordenados de forma crescente e, para cada tempo $t_i$, foi associada uma probabilidade $p_i = (i - 0{,}05)/100$. Os resultados dos gráficos $t_i \times p_i$ são apresentados na Figura 3.

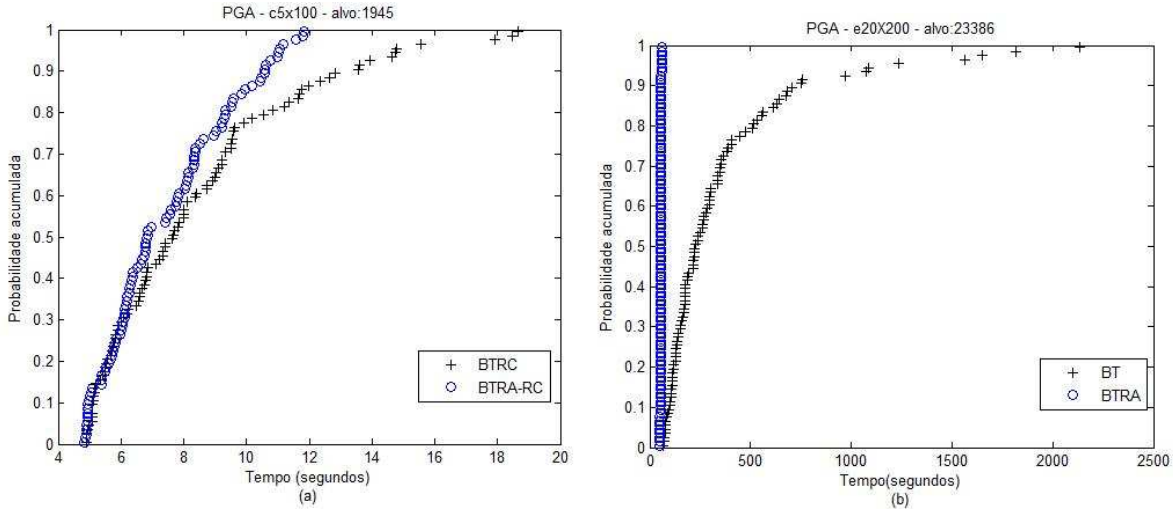


**Figura 3:** Teste de Probabilidade Empírica

É possível observar na Figura 3(a) e (b) que o algoritmo BTRA-RC demandou menos tempo que o BTRC para alcançar os valores alvo. De acordo com o gráfico da Figura 3(a), ele necessita de aproximadamente 12 segundos para encontrar o respectivo valor com quase 100% de probabilidade, enquanto que o algoritmo BTRC necessita de cerca de 19 segundos. A vantagem do algoritmo BTRA-RC é ainda maior para a instância E20x200, como pode ser analisado pelo gráfico da Figura 3(b). Essa figura mostra que o BTRA-RC necessita de um tempo significativamente menor do que o algoritmo BTRC para encontrar o valor alvo.

## 5 Conclusões e trabalhos futuros

Este trabalho propôs uma abordagem híbrida de duas fases, denominada BTRA-RC, ao Problema Generalizado de Atribuição, a qual combina os procedimentos heurísticos *Simulated Annealing* e Descida em Vizinhança Variável para a geração de solução inicial e Busca Tabu para refinamento dessa solução, utilizando de Relaxação Adaptativa e do procedimento de Reconexão por Caminhos como métodos de intensificação e diversificação.

O algoritmo foi testado em um conjunto de instâncias e os resultados obtidos foram comparados com outras abordagens da literatura, incluindo a versão anterior do algoritmo proposto, denominada BTRC. Os resultados obtidos mostraram que o algoritmo se apresentou mais eficiente que alguns algoritmos da literatura. Isso se deveu ao fato de que o BTRA-RC foi capaz de encontrar soluções melhores em alguns casos, além de requerer menos tempo para alcançar valores alvo em duas instâncias quando comparado com sua versão anterior.

No entanto, ficou claro que ainda é preciso aprimorar o algoritmo proposto para que seja mais competitivo em relação às melhores abordagens da literatura. Desta maneira, é necessário um estudo mais aprofundado sobre o método de modo que sejam experimentadas novas adaptações que conduzam o algoritmo à melhores resultados.

## Agradecimentos

Os autores agradecem ao CEFET-MG e à FAPEMIG pelo apoio ao desenvolvimento desta pesquisa. O primeiro autor agradece, também, ao Prof. Marcone Jamilson Freitas Souza, da Universidade Federal de Ouro Preto, pela co-orientação do presente trabalho.

**Referências**


Aiex, R. M.; Resende, M. G. C. e Ribeiro, C. C. (2002). Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, v. 8, p. 343–373.

Alfandari, L., Plateau, A. & Tolla P., 2001. A two-phase path-relinking algorithm for the generalized assignment problem. Proceedings of the 4th Metaheuristics International Conference, Porto, Portugal, 175–179.

Amini, M.M. & Racer, M., 1995. A hybrid heuristic for the generalized assignment problem. *European Journal of Operational Research*, vol. 87, 343–348.

Balachandran, V., 1976. An integer generalized transportation model for optimal job assignment in computer networks. *Operations Research*, vol. 24, 742–759.

Cattrysse, D. G., Salomom, M. & Wassenhove, L. N. V., 1994. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, vol. 72, 167–174.

Chu, P. C. & Beasley, J. E., 1997. A genetic algorithm for the generalized assignment problem. *Computers and Operations Research*, vol. 24, 17–23.

Dawande, M. & Kalagnanam, J., 1998. The multiple knapsack problem with color constraints. *IBM T. J. Watson Research, Tech. Rep*.

Diaz, J. A., Fernandez, E., 2001. A Tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, vol 132, 22–38.

Fisher, M. L. & Jaikumar, R., 2006. A generalized assignment heuristic for vehicle routing. *Networks*, vol. 11, 109–124.

Gendreau, M.; Hertz, A.; Laporte, G. A tabu search heuristic for the vehicle routing problem. Management science, Linthicum, v. 40, n. 10, p. 1276{1290, 1994

Glover, F.,1986. Future paths for integer programming and artificial intelligence. *Computers & Operations Research*, vol. 13, 533–549.

Glover, F.,1996. Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, vol. 65, 223–253.

Guignard, M. & Rosenwein, M., 1989. An improved dual-based algorithm for the generalized assignment problem. *Operations Research*, vol. 37, 658–663.

Hansen, P., 1986. The steepest ascent mildest descent heuristic for combinatorial programming. *Proceedings of the Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy.

Hung, M. S. & Fisk, J. C., 1979. A heuristic routine for solving large loading problems. *Naval Research Logistic Quarterly*, vol. 26, 643–650.

Kirkpatrick, S., GELLAT, D.C. & VECCHI, M.P., 1983. Optimization by Simulated Annealing. *Science*, vol. 220, 671–680.

Lenstra, J. K., Shmoys, D. B. & Tardos, E., 1990. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming: Series A and B*, vol. 46, 259–271.

Laguna, M., Kelly, J. P., Gonzalez-Velarde, J. I. & Glover, F., 1995. Tabu search for the multilevel generalized assignment problem. *European Journal of Operations Research*, vol. 42, 677–687.

Mazzola, J. B., Neebe, A. W. & Dunn, C. V. R., 1989. Production planning of a flexible manufacturing system in a material requirements planning environment. *International Journal of Flexible Manufacturing Systems*, vol. 1, 115–142.

Mladenovic, N. & Hansen, P., 1997. Variable Neighborhood Search. *Computers and Operations Research*, vol. 24, 1097–1100.

Narciso, M. G. & Lorena, L. A. N., 1999. Lagrangean / surrogate relaxation for generalized

assignment problems. *European Journal of Operational Research*, vol. 114, 165–177.
Nauss, R. M., 2003. Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing*, vol. 15, 249–266.
Osman, I. H., 1995. Heuristics for the generalized assignment problem: Simulated Annealing and Tabu Search approaches. *OR Spektrum*, vol. 17, 211–225.
Ross, G. T. & Soland, R. M., 1977. Modeling facility location problems as generalized assignment problems. *Management Science*, vol. 24, 345–357.
Rosseti, I. C. M., 2003. Estratégias sequenciais e paralelas de GRASP com reconexão por caminhos para o problema de síntese de redes a 2-caminhos. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro.
Sahni, S. & Gonzalez, T., 1976. P-complete approximation problems. *Journal of the ACM*, vol. 23, 555–565.
Schaerf, A. Tabu search techniques for large high-school timetabling problems. Amsterdam, 1996.
Souza, M. J. F., 2009. *Inteligência Computacional para Otimização*. Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/ InteligenciaComputacional.pdf.
Subtil, R. F.; Souza, M. J. F.; SOUZA, S. R. Uma abordagem híbrida aplicada ao Problema Generalizado de Atribuição. In: XIII Encontro de Modelagem Matemática e Computacional, 2010, Nova Friburgo (RJ). Anais do XIII EMC. Nova Friburgo (RJ): UERJ, 2010. p. 1-10.
Turek, J.,Wolf, J. L. & Yu, P. S., 1992. Approximate algorithms scheduling parallelizable tasks. *Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, San Diego, USA, 323-332.
Wilson, J. M., 2005. An algorithm for the generalized assignment problem with special ordered sets. *Journal of Heuristics*, vol. 11, 337–350.
Woodcock, A. J. & Wilson J. M., 2010. A hybrid tabu search/branch & bound approach to solving the generalized. assignment problem. *European Journal of Operational Research*, vol. 207, 566–578.
Yagiura, M., Yamaguchi, T., Ibaraki, T., 1998. A variable depth search algorithm with branching search for the generalised assignment problem. *Optimisation Methods & Software*, vol. 10, 419–441.
Yagiura, M., Glover F. & Ibaraki, T., 2006. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, vol. 169, 548–569.