HEURÍSTICAS PARA O PROBLEMA DA ATRIBUIÇÃO CONEXA EM VETORES

Joel Cruz Soares
Mestrado e Doutorado em Ciência da Computação, Universidade Federal do Ceará
Campus do Pici, Bloco 910, Pici, Fortaleza, Ceará, CEP 60440-900
joelcruz@lia.ufc.br

Manoel Campêlo
Departamento de Estatística e Matemática Aplicada, Universidade Federal do Ceará
Campus do Pici, Bloco 910, Pici, Fortaleza, Ceará, CEP 60440-900
mcampelo@lia.ufc.br

Tarcisio Ferreira Maciel
Departamento de Engenharia de Teleinformática, Universidade Federal do Ceará
Campus do Pici, Bloco 725, Pici, Fortaleza, Ceará, CEP 60455-970
maciel@gtel.ufc.br

RESUMO

O Problema da Atribuição Conexa em Vetores (ACV) possui aplicação em alocação de recursos em redes de comunicações móveis. Este problema tem como entrada um conjunto de símbolos $I = \{1, 2, \ldots, M\}$, um vetor v indexado por $J = \{1, 2, \ldots, N\}$, e um valor de ganho ρ_{ij} ao alocar $i \in I$ à posição j de v. Desejamos encontrar uma atribuição de exatamente um símbolo a cada posição do vetor, de modo a obter o maior ganho possível, sob a restrição de que símbolos repetidos formem um único bloco de posições adjacentes no vetor. Demonstramos que ACV é um problema NP-Difícil a partir de uma redução do Problema de Recoloração Convexa de Caminhos (RCC), apresentamos duas formulações de Programação Inteira, propomos quatro heurísticas para o problema e as comparamos através de resultados de experimentos computacionais.

PALAVRAS CHAVE. Complexidade Computacional, Programação Inteira, Heurística.

Tópicos: Otimização Combinatória, Programação Matemática.

ABSTRACT

The Connected Assignment in Arrays (CAA) has applications in resource allocation in mobile communications networks. This problem takes as input a set of symbols $I = \{1, 2, \ldots, M\}$, an array v indexed by $J = \{1, 2, \ldots, N\}$, and a gain value ρ_{ij} for allocating $i \in I$ to position j of v. We want to find an assignment of a unique symbol to each array position so as to maximize the gain, under the constraint that repeated symbols form a single block of adjacent positions in the array. We demonstrate that CAA is an NP-Hard problem by a reduction from the Convex Path Recoloring Problem (CPR), we present two IP formulations, we propose four heuristics for the problem and we compare them by results of computational experiments.

KEYWORDS. Computational complexity, Integer Programming, Heuristic.

Paper topics: Combinatorial Optimization, Mathematical Programming.

1. Introdução

O Problema da Atribuição Conexa em Vetores, ACV, que definimos no presente trabalho, consiste de um problema de otimização combinatória com aplicação dentro do contexto de alocação de recursos de rádio em sistemas de telecomunicações. Uma das técnicas utilizadas nesse contexto considera blocos de frequência, ou faixas de frequência, de mesmo tamanho que são disponibilizados aos clientes. Por particularidades relacionadas ao projeto do sistema, se vários blocos são alocados a um mesmo cliente, então eles devem ser adjacentes. O intuito principal é alocar os blocos de frequência respeitando tal restrição de alocação de modo a maximizar a taxa de transmissão total.

Resumidamente, ACV tem como entrada um conjunto de símbolos $I = \{1, 2, \ldots, M\}$, que representa os clientes, um vetor v indexado por $J = \{1, 2, \ldots, N\}$, que indica os blocos de frequência, e uma função ganho ρ tal que $\rho(i, j)$ (ou simplesmente ρ_{ij}) define o ganho ao alocar $i \in I$ à posição j de v. Desejamos encontrar uma atribuição conexa que tenha o maior ganho possível. Informalmente, uma atribuição conexa aloca exatamente um símbolo a cada posição, de tal modo que um símbolo repetido define um único bloco (subintervalo) do vetor, composto exclusivamente por este símbolo. Iremos nos referir a esta restrição, como restrição de adjacência ou restrição de conexidade.

A Figura 1 mostra uma entrada do Problema ACV. A matriz mostrada representa a função ganho ρ da instância. Cada linha i define os ganhos do símbolo i nas posições de v, e cada entrada (i, j) especifica o ganho de i na posição j de v. Dessa maneira, podemos concluir que $I = \{1, 2, 3, 4\}$ e $J = \{1, 2, 3, 4, 5, 6\}$, isto é, consideramos a alocação de 4 símbolos em um vetor de 6 posições.

i\j	***1***	***2***	***3***	***4***	***5***	***6***
1	1	2	1	2	0	0
2	0	0	3	0	1	0
3	2	2	0	0	2	1
4	2	0	0	2	0	4

Figura 1: Matriz de ganho que representa uma instância do Problema ACV.

Observe as atribuições da Figura 2. A atribuição A_1 aloca o símbolo 1 nas 4 primeiras posições, o símbolo 3 na quinta e o símbolo 4 na sexta. O ganho desta atribuição é a soma dos ganhos dos símbolos nas respectivas posições em que foram alocados. Portanto, A_1 tem ganho correspondente à soma $1 + 2 + 1 + 2 + 2 + 4$, isto é, tal atribuição tem ganho igual a 12. Além disso, notamos que esta respeita a restrição de adjacência imposta pelo problema e, sendo assim, representa uma solução viável para a instância considerada. A atribuição A_2 é uma solução inviável, pois viola a restrição de adjacência, e tem ganho de valor 15; as atribuições A_3 e A_4 tem valor 13 e são soluções ótimas para a instância considerada.

$A_1 \rightarrow$	1	1	1	1	3	4	$A_2 \rightarrow$	3	3	2	1	3	4
$A_3 \rightarrow$	3	3	2	4	4	4	$A_4 \rightarrow$	3	1	2	4	4	4

Figura 2: Exemplos de atribuição para a instância da Figura 1.

2. O Problema da Atribuição Conexa em Vetores

Nesta seção, definimos formalmente o problema ACV e provamos que sua versão de decisão é NP-Completo. Ao final, mostraremos duas formulações matemáticas de Programação Inteira para este problema.

Definição 1 (Atribuição Conexa) *Dados um conjunto $I = \{1, 2, \ldots, M\}$ de símbolos e um conjunto $J = \{1, 2, \ldots, N\}$ de posições, uma atribuição conexa é uma função $A : J \to I$ tal que, para todo $i \in I$, existe um intervalo $[a_i, b_i] \subseteq J$, onde $A(k) = i$ se, e somente se, $k \in [a_i, b_i]$. Além disso, dizemos que $i \in I$ é atribuído por A se, e somente se, existe $j \in J$ tal que $A(j) = i$.*

Problema 1 (Problema ACV) *O Problema da Atribuição Conexa em Vetores, **ACV**, consiste em, dado um vetor v indexado por $J = \{1, 2, \ldots, N\}$, um conjunto $I = \{1, 2, \ldots, M\}$ e uma função $\rho : I \times J \Rightarrow \mathbb{Z}_+$, que define o ganho ao inserir $i \in I$ na posição $j \in J$ do vetor v, encontrar uma atribuição conexa $A : J \to I$ dos símbolos de I em v que maximize $val(A)$, onde $val(A) = \sum_{j=1}^{N} \rho_{A(j)j}$.*

Problema 2 (Problema ACV-D) *O Problema de Decisão da Atribuição Conexa em Vetores, **ACV-D**, consiste em, dado um vetor v indexado por $J = \{1, 2, \ldots, N\}$, um conjunto $I = \{1, 2, \ldots, M\}$, uma função $\rho : I \times J \Rightarrow \mathbb{Z}_+$ que define o ganho ao inserir i na posição j do vetor v e um valor inteiro k, saber se existe uma atribuição conexa $A : J \to I$ dos elementos de I em v tal que $val(A) \geq k$.*

2.1. NP-Completude

Mostraremos que ACV-D é NP-Completo a partir de uma redução da versão de decisão do Problema da Recoloração Convexa de Caminhas (RCC) [Lima, 2011; Lima e Wakabayashi, 2014]; como consequência a versão de otimização, ACV, será NP-Difícil.

O Problema RCC é um problema conhecidamente NP-Difícil, e a demonstração de NP-Completude pode ser encontrada em [Moran e Snir, 2008]. Dado um grafo que é um caminho e uma coloração arbitrária nos vértices (não necessariamente própria), tal problema consiste em encontrar uma recoloração convexa que troca o mínimo de cores possível. Uma recoloração é convexa se o subgrafo induzido por cada cor é conexo. A Figura 3 apresenta um exemplo de uma instância de RCC e em 4 temos sua recoloração ótima.

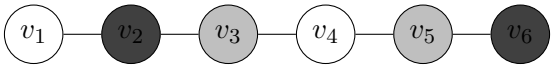

Figura 3: Exemplo de uma instância de RCC.

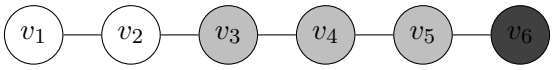

Figura 4: Uma recoloração ótima da instância apresentada na Figura 3.

Observe que minimizar o número de cores modificadas consiste em maximizar o número de cores mantidas. A definição seguinte expressa formalmente a versão de decisão do Problema RCC.

Definição 2 (Problema RCC-D) *Seja k' um valor inteiro e (P, C) um caminho P com coloração arbitrária C. O Problema de Decisão da Recoloração Convexa de Caminhos, **RCC-D**, consiste em encontrar uma recoloração convexa C' de (P, C) tal que $mantem(C') \geq k'$, onde $mantem(C') = |\{v \in V(P) : C(v) = C'(v)\}|$ e representa o número de cores mantidas em C'.*

Observe a similaridade entre os problemas RCC-D e ACV-D. A restrição de convexidade da recoloração implica que cores repetidas sejam todas adjacentes umas às outras, que é uma representação direta da restrição de conexidade em ACV-D. Dessa maneira, associamos os vértices com as posições do vetor e as cores com os símbolos. Assim, a cor de um vértice v_j em RCC-D representa o símbolo alocado na posição j correspondente do vetor no Problema ACV-D. Com base na coloração inicial C, definimos a seguinte função de ganho ρ:

$$\rho_{ij} = \begin{cases} 1, & \text{se } i = C(v_j), \\ 0, & \text{caso contrário.} \end{cases}$$

A função ganho ρ especificada na redução determina que, para cada posição do vetor, exista um único símbolo de ganho não nulo e igual a 1 (o ganho dos demais é igual a zero). Este símbolo i, de ganho não nulo na posição j, é aquele associado à cor inicial do vértice v_j. Portanto, qualquer unidade de ganho na função objetivo da instância definida pela redução deve manter uma cor na instância associada em RCC-D. Pela associação direta, se A é uma atribuição conexa de ACV-D, então a solução da instância de RCC-D será:

$$C'(v_j) = A(j)$$

Observe que a recoloração obtida deve ser convexa, pois a atribuição A, por hipótese, é conexa. Além disso, $val(A) \geq k$ se, e somente se, $mantem(C') \geq k'$, com $k' = k$. Portanto, a redução é válida. E como ACV-D está claramente em NP, podemos concluir que este problema é NP-Completo.

2.2. Formulações Matemáticas

As formulações que apresentaremos utilizam uma matriz de atribuições T, cujo propósito é garantir a condição de adjacência de símbolos de mesmo valor (garantir a conexidade da atribuição) em v. Cada coluna p da matriz T define um padrão de alocação (ou padrão de atribuição) e cada linha se refere a uma posição j do vetor v. A entrada $t_{jp} \in \{0, 1\}$ indica se a posição j é utilizada ou não pelo padrão p, isto é, se p utiliza ou não j.

Um padrão p da matriz de padrões T define um intervalo não vazio $[a, b]$ de alocação no vetor v. Para um instância de ACV onde $N = 4$, temos a seguinte matriz:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Observe que a sequência de padrões de T inicia com os N padrões que utilizam uma única posição, seguindo com os $N - 1$ padrões de 2 posições, $N - 2$ padrões de 3 posições, chegando ao único padrão que aloca todas as posições.

Sendo $N \times P$ a ordem da matrix T, observe que P é dado por:

$$P = \sum_{k=1}^{N} (N - k + 1) = \frac{1}{2}N^2 + \frac{1}{2}N = \binom{N+1}{2}$$

O termo do somatório refere-se à existência de $N - k + 1$ padrões de tamanho k. Consideramos $Q = \{1, 2, \ldots, P\}$ o conjunto de índices dos padrões.

Para cada $i \in I$ e $p \in Q$, definimos uma variável de decisão binária x_{ip} com o seguinte significado:

$$x_{ip} = \begin{cases} 1, & \text{se } i \text{ segue o padrão de atribuição } p, \\ 0, & \text{caso contrário.} \end{cases}$$

Para cada padrão p, definimos uma variável binária y_p tal que:

$$y_p = \begin{cases} 1, & \text{se o padrão } p \text{ é utilizado,} \\ 0, & \text{caso contrário.} \end{cases}$$

Usando essas variáveis, apresentamos duas formulações de programação inteira para ACV. A primeira, dada por (1)-(6), será denotada por Formulação XY. A restrição (2) garante que todo símbolo i pode utilizar, no máximo, um padrão p. A restrição (3) permite a utilização de um padrão p para alocação de algum símbolo i somente se este padrão for escolhido, ou seja, $y_p = 1$. Por fim, a restrição (4) garante que não serão utilizados padrões conflitantes, isto é, padrões com interseção. Dizemos que dois padrões são conflitantes se ambos ocupam alguma posição em comum. Formalmente, p e p' são conflitantes se $\sum_{j=1}^{N} t_{jp}t_{jp'} \geq 1$. A função objetivo desta formulação calcula o valor do ganho da atribuição, que deve ser maximizada.

$$\max \sum_{i=1}^{M}\sum_{j=1}^{N}\rho_{ij}\sum_{p=1}^{P} t_{jp}x_{ip} \tag{1}$$

Sujeito a:

$$\sum_{p=1}^{P} x_{ip} \leq 1, \forall i \in I \tag{2}$$

$$\sum_{i=1}^{M} x_{ip} \leq y_p, \forall p \in Q \tag{3}$$

$$y_p + y_{p'} \leq 1, \forall p, p' \in Q\text{: } p \text{ é conflitante com } p' \tag{4}$$

$$x_{ip} \in \{0,1\}, \forall i \in I \text{ e } \forall p \in Q \tag{5}$$

$$y_p \in \{0,1\}, \forall p \in Q \tag{6}$$

Note que a Formulação XY apresenta dois grupos de restrições, a saber (2) e (4), separáveis, que são acopladas pela restrição (3). Exploramos este fato na Seção 3.

A segunda formulação, a ser chamada Formulação X, foi proposta em [Lima, 2012] e utiliza apenas as variáveis de decisão x_{ip}. Embora sua relaxação seja mais forte que a de XY, o modelo não é mais separável. A restrição (7) estabelece que todo símbolo $i \in I$ deve seguir no máximo um padrão de atribuição. A restrição (8) garante que haverá no máximo um símbolo alocado em cada posição do vetor v. A função objetivo calcula o valor da atribuição, a ser maximizado.

$$\max \sum_{i=1}^{M}\sum_{j=1}^{N}\rho_{ij}\sum_{p=1}^{P} t_{jp}x_{ip}$$

Sujeito a:

$$\sum_{p=1}^{P} x_{ip} \leq 1, \forall i \in I \tag{7}$$

$$\sum_{i=1}^{M}\sum_{p=1}^{P} t_{jp}x_{ip} \leq 1, \forall j \in J \tag{8}$$

$$x_{ip} \in \{0,1\}, \forall i \in I \text{ e } \forall p \in Q \tag{9}$$

3. Heurísticas

Neste seção, apresentamos 4 heurísticas para o problema ACV. As duas primeiras são heurísticas de baixo custo computacional, a terceira aplica conceitos de Relaxação Lagrangeana e a última utiliza uma solução da relaxação linear da Formulação X.

3.1. Uma Heurística Baseada em Multa - H_1

A Heurística H_1 é inspirada no algoritmo 2-aproximativo para o problema RCC [Moran e Snir, 2008]. A partir de uma atribuição possivelmente desconexa A de ganho máximo, a heurística H_1 procura obter uma solução conexa $\hat{A}$ com $val(\hat{A})$ o mais próximo possível de $val(A)$. Primeiramente, definimos uma função de multa relacionada à atribuição inicial A, que toma como parâmetros um símbolo i e um subintervalo $B \subseteq J$ de v. O objetivo desta função é indicar uma estimativa da máxima quantidade perdida em $val(A)$ ao tornar o intervalo B preenchido apenas pelo símbolo i.

Definimos a função multa da seguinte forma:

$$multa_A(i, B) = \rho_A(\overline{J_A(i)} \cap B) - \rho_i(\overline{J_A(i)} \cap B) + \rho_i(J_A(i) \cap \overline{B})$$

onde $J_A(i) = \{j \in J : A(j) = i\}$, $\overline{J_A(i)} = \{j \in J : A(j) \neq i\}$, $\rho_A(W) = \sum_{j \in W} \rho_{A(j)j}$, $\rho_i(W) = \sum_{j \in W} \rho_{ij}$ e $\bar{B} = J \setminus B$. Observe que $\rho_i(J_A(i) \cap \overline{B}) = \rho_i(J_A(i)) - \rho_i(J_A(i) \cap B))$. Então a multa se resume em

$$multa_A(i, B) = \rho_i(J_A(i)) - [\rho_i(B) - \rho_A(\overline{J_A(i)} \cap B))]$$

Portanto, considerando um símbolo $i \in I$, encontrar o subintervalo B que minimiza $multa_A(i, B)$ significa encontrar um subintervalo B que maximiza $\rho_i(B) - \rho_A(\overline{J_A(i)} \cap B))$. Podemos encontrar este intervalo em $O(N)$. De fato, considerando a sequência $S_i = \langle s_1^i, s_2^i, \ldots, s_j^i, \ldots, s_N^i \rangle$, associada ao símbolo i, onde $s_j^i = \rho_{ij} - \rho_{A(j)j}$, se $A(j) \neq i$, e $s_j^i = \rho_{ij}$, se $A(j) = i$, tal problema consiste em encontrar a subsequência de soma máxima de S_i [Chao e Zhang, 2008; Kad].

Dessa maneira, a Heurística H_1 se resume em: (i) calcular uma atribuição (possivelmente não conexa) A de ganho máximo – onde todo $A(j)$ corresponde ao símbolo i de maior ganho na posição j; (ii) a partir de A, determinar, para cada símbolo $i \in I$, o subintervalo de multa mínima B_i; (iii) seguindo a ordem decrescente de $\rho_i(B_i)$, alocar iterativamente i em B_i, caso este bloco esteja totalmente disponível; (iv) preencher posições ainda vazias, mantendo conectividade.

3.2. Uma Heurística Baseada na Diferença para o Ganho Máximo por Posição - H_2

A Heurística H_2 assemelha-se à H_1. Para cada símbolo $i \in I$, definimos uma outra sequência $\langle s_1^i, s_2^i, \ldots, s_N^i \rangle$, associada a i, e encontramos o subintervalo de soma máxima B_i desta sequência. Com isso, aplicamos uma estratégia similar a de H_1 para alocar os símbolos. Sendo I' o subconjunto de símbolos ainda não aloacados e J' o subconjunto de posições livres, a sequência de cada símbolo $i \in I'$ é calculada iterativamente como $s_j^i = \rho_{ij} - \max_{i' \in I' \setminus \{i\}}\{\rho_{i'j}\}, \forall j \in J'$ e $s_j^i = -\infty, \forall j \in J \setminus J'$.

3.3. Uma Heurística Baseada em Relaxação Lagrangeana - H_3

Mostraremos, a seguir, uma heurística baseada nos conceitos de Relaxação Lagrangeana [Nemhauser e Wolsey, 1988]. Considere a Formulação XY. Observe que as únicas restrições que relacionam as variáveis x e y são as restrições (3). Vamos relaxá-las e penalizá-las na função objetivo através de multiplicadores $\lambda_p \geq 0, \forall p \in Q$. Dessa maneira, a Relaxação Lagrangeana do modelo anterior será:

$$\max \sum_{i=1}^{M} \sum_{p=1}^{P} g_{ip} x_{ip} + \sum_{p=1}^{P} \lambda_p y_p$$

Sujeito a:

$$\sum_{p=1}^{P} x_{ip} \leq 1, \forall i \in I \tag{10}$$

$$y_p + y_{p'} \leq 1, \forall p, p' \in Q, \text{conflitantes} \tag{11}$$

$$x_{ip} \in \{0, 1\}, \forall i \in I \text{ e } \forall p \in Q \tag{12}$$

$$y_p \in \{0, 1\}, \forall p \in Q \tag{13}$$

onde $g_{ip} = \sum_{j=1}^{N} \rho_{ij} t_{jp} - \lambda_p$.

Observe que neste modelo relaxado as duas variáveis x e y são independentes e, desta forma, podemos dividi-lo em dois:

$$(P_1) \begin{cases} \max \sum_{i=1}^{M} \sum_{p=1}^{P} g_{ip} x_{ip} \\ \text{Sujeito a:} \\ \sum_{p=1}^{P} x_{ip} \leq 1, \forall i \in I \\ x_{ip} \in \{0,1\}, \forall i \in I \text{ e } \forall p \in Q \end{cases} \qquad (P_2) \begin{cases} \max \sum_{p=1}^{P} \lambda_p y_p \\ \text{Sujeito a:} \\ y_p + y_{p'} \leq 1, \forall p, p' \in Q, \text{conflitantes} \\ y_p \in \{0,1\}, \forall p \in Q \end{cases}$$

Observe ainda que qualquer combinação de duas soluções ótimas para (P_1) e (P_2) corresponde a uma solução ótima para a Relaxação Lagrangeana.

Perceba que ambos os problemas são fáceis de resolver. A resolução de (P_1) pode ser feita a partir da determinação de $p_i \in \arg\max_{p \in Q} g_{ip}, \forall i \in I$. De fato, uma solução ótima para (P_1) tem valor $\sum_{i \in I} \max\{0, g_{ip_i}\}$, ocorrendo no ponto x onde $x_{ip} = 1$ se e somente se $p = p_i$ e $g_{ip_i} > 0$. Uma solução ótima para (P_2) corresponde a encontrar um conjunto independente de peso máximo no grafo G, com conjunto de vértices Q, conjunto de arestas dadas pelos pares de padrões conflitantes e peso do vértice p igual a λ_p. Observe que G é um grafo de intervalo, pois possui representação em intervalo. Sabemos que o problema de encontrar conjuntos independentes de peso máximo em grafos gerais é NP-Difícil, no entanto, para grafos de intervalos, o problema é fácil e se conhecem algoritmos polinomiais para resolvê-lo [Hsiao et al., 1992].

Seja $L(\lambda)$ o valor da relaxação Lagrangeana para um dado vetor de multiplicadores λ, ou seja, $L(\lambda)$ é a soma dos valores ótimos de (P_1) e (P_2) para λ. Sabemos que $L(\lambda)$ é um limite superior para o valor ótimo do problema original, $\forall \lambda \geq 0$. Sendo assim, desejamos encontrar $\lambda \geq 0$ tal que $L(\lambda)$ seja o menor possível (Problema Lagrangeano Dual). Com este objetivo utilizamos o método de subgradientes. Este método faz uso de um procedimento iterativo para definir λ. A partir da escolha de um multiplicador inicial λ^0, o valor da componente i de λ na iteração k é calculado como:

$$\lambda_i^k = \max\{0, \lambda_i^{k-1} - t^{k-1} G(x^{k-1}, y^{k-1})\}, \text{para } k > 0 \tag{14}$$

onde t^{k-1} é um escalar, que representa o tamanho de passo utilizado na iteração $k-1$, e $G(x^{k-1}, y^{k-1})$ é um vetor subgradiente de $L(\lambda^{k-1})$, dados por:

$$G(x^k, y^k) = (g_1, \ldots, g_p, \ldots, g_P), \text{ com } g_p = y_p^k - \sum_{i=1}^{M} x_{ip}^k, \forall p \in Q, \tag{15}$$

$$t^k = \frac{\epsilon \cdot (L(\lambda^{k-1}) - LB)}{\|G(x^{k-1}, y^{k-1})\|^2}, \tag{16}$$

sendo x^k, y^k as soluções ótimas dos problemas (P_1) e (P_2), respectivamente, para λ^k.

O tamanho de passo t^k é ajustado em toda iteração k, e o parâmetro LB é um limite inferior. O valor ϵ é um escalar entre 0 e 2 e tem influência na velocidade de convergência do método.

O Algoritmo 1 expressa a Heurística H_3. Basicamente, a ideia é transformar os pontos (x^k, y^k) obtidos ao longo da resolução do Dual Lagrangenano em soluções viáveis para problema original. Dentre as várias alternativas possíveis, utilizamos o Algoritmo 2. Ele determina a melhor solução, considerando apenas os padrões (não-coflitantes) escolhidos em cada solucão y^k do problema (P_2).

Algoritmo 1: Heurística H_3

Entrada: M,N e a função ganho ρ
Saída: Uma atribuição conexa
início
  $\lambda \leftarrow \lambda^0$;
  para *i=1 até* γ **faça**
    Sejam x e y soluções ótimas de (P_1) e (P_2) para λ, respectivamente;
    $A_i \leftarrow calcularAtribuicaoViavel(x, y)$;
    obtenha LB e compute t^i;
    atualize o parâmetro λ;
  fim
  retorna $\arg\max_{A\in\{A_1,\ldots,A_\gamma\}}\{val(A)\}$;
fim

Algoritmo 2: calcularAtribuicaoViavel

Entrada: A solução y do problema (P_2)
Saída: Uma atribuição conexa
início
  Seja $G = (V_1 \cup V_2, V_1 \times V_2)$ um grafo bipartido completo tal que $V_1 = I$, $V_2 = \{p : y_p = 1\}$;
  Calcule um emparelhamento $M = \{i_1p_1, \ldots, i_kp_k\}$ de peso máximo nas arestas de G, onde o peso de $ip \in V_1 \times V_2$ é dado por $w_{ip} = \sum_{j=1}^{N} \rho_{ij} \cdot t_{jp}$;
  retorna A tal que $A(j) = i, \forall i = i_1, i_2, \ldots, i_k, \forall j \in p_i$;
fim

3.4. Uma Heurística Baseada em Relaxação Linear - H_4

A Heurística H_4 começa calculando uma solução ótima da relaxação linear da Formulação X. Com a solução encontrada, para cada símbolo $i \in I$, é definido o conjunto $S_i = \{p \in Q : x_{ip} > 0\}$, que consiste dos padrões que foram alocados para i na solução relaxada. Baseado nos conjuntos S_i's a heurística aplica uma estratégia gulosa para alocar os símbolos. A cada iteração, é escolhido o par (i, P_i), $P_i \in S_i$, de maior ganho ainda disponível para alocação, ou seja, tal que i não foi alocado e toda posição de P_i está livre. O Algoritmo 3 apresenta em detalhes a Heurística H_4.

4. Resultados Computacionais

Os resultados computacionais apresentados nesta seção foram obtidos num ambiente computacional Ubuntu 14.04 LTS de 64 bits com hardware equipado com processador intel® Core™ i7 com *clock* de 3.4 GHz e 16 GB de memória DDR3. Os algoritmos foram implementados na linguagem Java versão 1.8 e foi utilizado o CPLEX 12.6.1 como ferramenta para resolução dos problemas de programação linear e inteira.

4.1. Instâncias

Utilizamos duas classes de instâncias. A primeira, refere-se àquelas obtidas através do simulador desenvolvido em [Lima, 2012]. Este simulador permite a especificação de um cenário de comunicação celular, através da definição de parâmetros relacionados ao sistema, como, por exemplo, o número de usuários, taxas de transmissão e serviços do sistema. Baseado nestes parâmetros, o software faz uma simulação de um ambiente celular moderno com as características definidas e, através de modelos matemáticos, pode-se calcular as taxas de transmissão dos usuários que representam a função ganho ρ de ACV. Chamaremos genericamente as instâncias desta classe de Instâncias Simuladas.

Algoritmo 3: Heurística H_4

Entrada: M,N e a função ganho ρ
Saída: Uma atribuição conexa
início
  Calcule uma solução ótima A_R para a relaxação linear da Formulação X;
  $\forall i \in I$, seja $S_i \leftarrow \{p : x_{ip} > 0\}$, onde x_{ip} é uma variável da solução A_R;
  $I' \leftarrow \{1, 2, \ldots, M\}$;
  Seja $J(p) \subseteq J$ o conjunto de posições ocupadas por p, $\forall p \in Q$;
  enquanto $I' \neq \emptyset$ *e há posições não alocadas* **faça**
    $\forall i \in I'$, seja P_i, tal que $P_i \in \arg\max_{p \in S_i}\{\sum_{j \in J(p)} \rho_{ij}\}$;
    Seja P_e, tal que $e \in \arg\max_{i \in I'}\{\sum_{j \in J(p)} \rho_{ij}\}$;
    Aloque e com o padrão P_e;
    $I' \leftarrow I' \setminus \{e\}$;
    para todo $i \in I'$ **faça**
      para todo $p \in S_i$ **faça**
        se $J(P_e) \cap J(p) \neq \emptyset$ **então**
          $S_i \leftarrow S_i \setminus \{p\}$;
          Acrescente a S_i os possíveis dois padrões associados $J(p) \setminus J(P_e)$;
        fim
      fim
    fim
  fim
  Se há posições não alocadas, atribua símbolos mantendo a conexidade;
fim

A segunda classe de instâncias compreende aquelas obtidas aleatoriamente com valores de ganhos uniformemente distribuídos no intervalo $[0, 200]$. Iremos nos referir a estas instâncias como Instâncias Aleatórias.

Para cada classe, dividimos as instâncias em 3 grandes grupos: $N = 20$, $N = 60$ e $N = 100$. Tais tamanhos são compatíveis com a aplicação prática. Para um dado N fixo, definimos 6 valores para M: $N/5$, $2N/5$, $3N/5$, $4N/5$, $5N/5$ e $6N/5$. Para cada par (N, M) foram geradas 30 instâncias simuladas e 30 instâncias aleatórias. Assim, cada grupo com N fixo tem um total de 180 instâncias de cada classe.

Usando as instâncias, desejamos verificar o desempenho das heurísticas propostas em termos da qualidade de solução e de tempo de computação, tanto para situações mais próximas da realidade (através de instâncias simuladas) como também estudá-las num contexto mais amplo (através de instâncias aleatórias). Definimos a Razão de Aproximação (RA) de uma solução como sendo a razão (Valor da Solução / Valor Ótimo). Chamaremos de Heurística H_0 a heurística proposta em [Lima et al., 2016], que foi desenvolvida para o contexto prático de aplicação do problema. Esta será utilizada como referência para avaliar os resultados obtidos para as heurísticas propostas. A solução ótima foi obtida usando a Formulação X.

4.2. Resultados Para $N = 20$

As tabelas 1 e 2 sumarizam os resultados obtidos para as 180 instâncias simuladas e as 180 instâncias aleatórias, respectivamente, para $N = 20$. O tempo médio de computação da solução ótima para as instâncias simuladas foi de $68,387$ milissegundos, que é $3,9$ vezes maior que o tempo médio da heurística mais lenta, H_4, e 1.179 vezes maior que o tempo médio da heurística

mais rápida, H_1. Notamos que houve instâncias em que H_1 e H_2 obtiveram soluções com razão de aproximação bem próximas de 75%. Por outro lado, H_3 obteve RA mínima de 91%.

Para as instâncias aleatórias, o tempo de computação da solução ótima foi em média $184,34$ milissegundos. Este valor é $4,78$ vezes maior que o maior tempo médio, de H_4, e 1.645 vezes maior que o menor tempo médio, de H_1. Observamos que houve instâncias aleatórias onde a heurística H_0 obteve soluções com razão de aproximação próxima de 46% e algumas fizeram H_1 obter solução com 72% de proximidade da solução ótima.

Tabela 1: Resultados Médios Gerais Para $N = 20$ (Instâncias Simuladas).

$N = 20$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,988$	$0,972$	$0,975$	$0,992$	**0,998**
RA Min.	$0,896$	$0,750$	$0,757$	**0,907**	$0,887$
RA Max.	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$
Tempo Médio (ms)	$0,059$	**0,058**	$0,110$	$10,222$	$17,359$
Tempo Min. (ms)	**0,009**	$0,019$	$0,020$	$2,244$	$4,547$
Tempo Max. (ms)	$0,686$	**0,200**	$0,649$	$23,416$	$33,762$

Tabela 2: Resultados Médios Gerais Para $N = 20$ (Instâncias Aleatórias).

$N = 20$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,818$	$0,930$	$0,962$	$0,987$	**0,996**
RA Min.	$0,469$	$0,722$	$0,856$	**0,921**	$0,915$
RA Max.	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$
Tempo Médio (ms)	$0,254$	**0,112**	$0,727$	$16,173$	$38,529$
Tempo Min. (ms)	$0,084$	**0,046**	$0,055$	$3,051$	$6,326$
Tempo Max. (ms)	$1,195$	**0,458**	$2,568$	$38,171$	$89,805$

4.3. Resultados Para $N = 60$

As tabelas 3 e 4 resumem os resultados obtidos para as 180 instâncias simuladas e as 180 instâncias aleatórias, respectivamente, com $N = 60$. O tempo médio de computação da solução ótima para o conjunto de instâncias simuladas foi de 4.768 milissegundos. Este valor é $3,5$ vezes maior que o tempo médio da heurística de maior tempo, H_4, e 47.207 vezes maior que o da heurística de menor tempo, H_1. Observamos que, para algumas instâncias, H_2 obteve soluções cuja razão de aproximação foi em torno de 64%. Destacamos o desempenho médio das outras heurísticas propostas, com RA média de 99%

O tempo de computação médio da solução ótima para as instâncias aleatórias com $N = 60$ foi de 10.898 milissegundos, que é $3,8$ vezes maior que o tempo médio de computação da heurística de maior tempo, H_4, e 32.924 vezes maior que o tempo médio de computação da de menor tempo, H_1. Podemos notar que a razão de aproximação média para a heurística de referência H_0 foi $62,5\%$, que é consideravelmente menor que a das demais heurísticas, e para algumas instâncias esta heurística obteve soluções cuja aproximação é $33,8\%$ da ótima.

Tabela 3: Resultados Médios Gerais para $N = 60$ (Instâncias Simuladas).

$N = 60$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,991$	$0,985$	$0,960$	$0,988$	**0,999**
RA Min.	$0,900$	$0,854$	$0,641$	**0,921**	$0,901$
RA Máx.	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$
Tempo Médio (ms)	$0,281$	**0,101**	$2,991$	$160,786$	$1346,800$
Tempo Min. (ms)	**0,025**	$0,029$	$0,040$	$34,072$	$253,777$
Tempo Max. (ms)	$4,880$	**0,434**	$26,548$	$480,116$	$3659,490$

Tabela 4: Resultados Médios Gerais para $N = 60$ (Instâncias Aleatórias).

$N = 60$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,625$	$0,884$	$0,948$	$0,945$	**0,992**
RA Min.	$0,338$	$0,789$	$0,808$	$0,874$	**0,903**
RA Máx.	$0,917$	$0,998$	$0,999$	$0,999$	**1,000**
Tempo Médio (ms)	$2,316$	**0,331**	$32,980$	$286,017$	$2854,650$
Tempo Min. (ms)	$0,789$	**0,069**	$0,642$	$57,302$	$440,368$
Tempo Max. (ms)	$24,637$	**1,318**	$139,187$	$858,880$	$5738,542$

4.4. Resultados Para $N = 100$

As tabela 5 e 6 resumem os resultados médios obtidos para as 180 instâncias simuladas e para as 180 instâncias aleatórias, respectivamente, com $N = 100$. O tempo médio de computação da solução ótima para o conjunto de instâncias simuladas foi de 53.529 milissegundos. Este valor é $3,4$ vezes maior que o tempo médio da heurística de maior tempo, H_4, e 262.397 vezes maior que o da heurística de menor tempo, H_1. Observamos que, para algumas instâncias, H_2 obteve soluções cuja razão de aproximação foi em torno de 60%. Todas as demais encontram solução com no mínimo 92% do valor da solução ótima.

O tempo de computação médio da solução ótima para as instâncias aleatórias com $N = 100$ foi de 74.108 milissegundos, que é $3,6$ vezes maior que o tempo médio de computação da heurística de maior tempo, H_4, e 129.786 vezes maior que o tempo médio de computação da de menor tempo, H_1. Podemos notar que a razão de aproximação média para a heurística de referência H_0 foi $58,6\%$, que é consideravelmente menor que a das demais heurísticas, e para algumas instâncias esta heurística obteve soluções cuja aproximação é $30,5\%$ da ótima.

Tabela 5: Resultados Médios Gerais para $N = 100$ (Instâncias Simuladas).

$N = 100$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,995$	$0,992$	$0,941$	$0,993$	**0,999**
RA Min.	$0,917$	$0,926$	$0,600$	$0,949$	**0,986**
RA Máx.	$1,000$	$1,000$	$1,000$	$1,000$	$1,000$
Tempo Médio (ms)	$0,741$	**0,204**	$18,574$	$862,743$	$15456,465$
Tempo Min. (ms)	$0,047$	**0,036**	$0,102$	$105,438$	$2453,095$
Tempo Max. (ms)	$5,881$	**0,638**	$155,905$	$3110,478$	$39427,780$

Tabela 6: Resultados Médios Gerais para $N = 100$ (Instâncias Aleatórias).

$N = 100$	H_0	H_1	H_2	H_3	H_4
RA Média	$0,586$	$0,870$	$0,945$	$0,900$	**0,988**
RA Min.	$0,305$	$0,790$	$0,844$	$0,829$	**0,878**
RA Máx.	$0,825$	$0,960$	$0,997$	$0,961$	**1,000**
Tempo Médio (ms)	$5,349$	**0,571**	$164,469$	$888,455$	$20255,474$
Tempo Min. (ms)	$2,151$	**0,074**	$2,493$	$139,364$	$3249,167$
Tempo Max. (ms)	$21,902$	**4,237**	$908,099$	$3101,344$	$41586,062$

5. Conclusões

Por estes resultados, podemos observar que a heurística H_4 sempre apresentou melhores resultados que a heurística encontrada na literatura, H_0, quando comparamos a qualidade de solução obtida.

Em relação ao tempo de computação, notamos que a heurística H_1 é a de menor custo computacional dentre todas, apresentando resultados próximos aos da heurística H_0 para instâncias simuladas e a superando nas instâncias aleatórias. Por outro lado, a heurística H_4 apresenta os maiores tempos de computação.

Para o contexto da aplicação, onde se exige um tempo de resposta da ordem de milissegundos, a heurísica H_1 parece constituir uma boa alternativa à H_0. Vale notar que, embora as formulações consigam encontram as soluções ótimas para os tamanhos de instâncias testadas, seus tempos de computação são proibitivos para a aplicação prática.

No momento não sabemos se alguma das heurísticas é um algoritmo de aproximação. Mais precisamente, não sabemos se o problema ACV é aproximável por um fator maior que $1/N$. No entanto, podemos mostrar um algoritmo $1/k$-aproximativo para o caso particular onde cada símbolo i possui ganho não nulo em no máximo k posições do vetor (Problema k-ACV).

Referências

Kadane's algorithm. `http://www.algorithmist.com/index.php/Kadane's_Algorithm`. Acessado: 12-04-2017.

Chao, K.-M. e Zhang, L. (2008). *Sequence Comparison: Theory and Methods*. Springer Publishing Company, Incorporated, 1 edition. ISBN 1848003196, 9781848003194.

Hsiao, J. Y., Tang, C. Y., e Chang, R. S. (1992). An efficient algorithm for finding a maximum weight 2-independent set on interval graphs. *Inf. Process. Lett.*, 43(5):229–235.

Lima, F. R. M. (2012). *Maximizing spectral efficiency under minimum satisfaction constraints on multiservice wireless networks*. PhD thesis, Universidade Federal do Ceará.

Lima, F. R. M., Maciel, T. F., e Cavalcanti, F. R. P. (2016). Radio resource allocation in sc-fdma uplink with resource adjacency constraints. *Journal of Communication and Information Systems*, 31(1):272 – 289.

Lima, K. R. e Wakabayashi, Y. (2014). Convex recoloring of paths. *Discrete Applied Mathematics*, 164, Part 2:450 – 459.

Lima, K. R. P. S. (2011). *Recoloração convexa de caminhos*. PhD thesis, Universidade de São Paulo.

Moran, S. e Snir, S. (2008). Convex recolorings of strings and trees: Definitions, hardness results and algorithms. *Journal of Computer and System Sciences*, 74(5):850 – 869. ISSN 0022-0000.

Nemhauser, G. L. e Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA. ISBN 0-471-82819-X.