O Problema de Programação de Sessões Técnicas de Conferências: o caso do SBPO

Rubens Correia, Anand Subramanian, Teobaldo Bulhões
Universidade Federal da Paraíba – Centro de Informática
Rua dos Escoteiros, s/n – Mangabeira, João Pessoa – PB, 58055-000
rubens.jp.br@gmail.com, anand@ci.ufpb.br, tbulhoes@ci.ufpb.br

Puca Huachi Vaz Penna
Universidade Federal de Ouro Preto – Departamento de Computação
Campus Universitário – Morro do Cruzeiro, Ouro Preto – MG, 35400-000
puca@iceb.ufop.br

RESUMO

Este trabalho trata do problema de programação de sessões técnicas de conferências no contexto do SBPO. O objetivo é maximizar o benefício de alocar trabalhos com temas em comum em uma mesma sessão, satisfazendo varias restrições de recurso bem como aquelas impostas pelos organizadores. Duas formulações matemáticas baseadas em programação linear inteira são apresentadas para o problema, que por sua vez é mostrado ser NP-Difícil. Dado o limite de escalabilidade dos modelos, um algoritmo híbrido que combina uma meta-heurística baseada em busca local e um procedimento baseado em programação matemática é proposto para resolver instâncias realísticas de grande porte. Experimentos computacionais foram conduzidos em instâncias de pequeno porte, nas quais a heurística desenvolvida foi capaz de encontrar todos os ótimos conhecidos. Além disso, foram realizados testes em instâncias de duas edições passadas do evento, para as quais o algoritmo obteve resultados superiores se comparado com a solução gerada manualmente.

PALAVRAS CHAVE. Programação de sessões técnicas de conferencias, Programação Inteira, *Matheuristic*, SBPO.

Tópicos: OC – Otimização Combinatória

ABSTRACT

This paper deals with the conference scheduling problem arising in the context of SBPO. The objective is to maximize the benefit of clustering papers with common topics in the same session, while satisfying several resource constraints as well as those imposed by the organizers. Two mathematical formulations based on integer linear programming are proposed for the problem, which in turn is shown to be NP-hard. Given the limited scalability of the models, a hybrid algorithm that combines a local search-based metaheuristic with a mathematical programming-based procedure is proposed to solve realistic large-size instances. Computational results were conducted for small-size instances where the developed algorithm was capable of finding all known optimal solutions. Moreover, experiments were also carried out for real-life large-size instances of the last two editions of the event, for which the algorithm obtained superior results when compared to those manually generated.

KEYWORDS. Conference scheduling, Integer programming, Matheuristic, SBPO.

Paper topics: CO - Combinatorial Optimization

1. Introdução

Anualmente, vários eventos acadêmicos, das mais diversas áreas, são organizados com o intuito de compartilhar e difundir o conhecimento. Organizar eventos desse tipo demanda tempo e esforço por parte de seus organizadores. Durante a organização, uma atividade comum é a elaboração da programação das sessões técnicas do evento. Geralmente, vários trabalhos, dentro de um conjunto pré-determinado de temas, são submetidos. Por conseguinte, organizá-los manualmente de forma a construir sessões que irão compor o evento é uma tarefa árdua, complexa e, por vezes, leva a soluções insatisfatórias. A complexidade dessa atividade advém da natureza combinatória do problema, dado o grande número de soluções possíveis e o número de variáveis envolvidas. [Silva et al., 2013].

Problemas envolvendo programação de sessões de eventos têm sido estudados há pelo menos três décadas. Na literatura, é possível encontrar basicamente duas abordagens para o problema, como distingue Thompson [2002]. A primeira consiste em uma abordagem baseada na perspectiva dos palestrantes, na qual podem ser levadas em consideração suas preferências e disponibilidade. A segunda baseia-se na perspectiva dos participantes do evento, na qual leva-se em consideração, principalmente, as preferências desses por certas palestras disponibilizadas no evento. Um dos trabalhos pioneiros nesse problema foi o de Eglese e Rand [1987], que gerou uma solução para um pequeno evento levando em consideração as preferências dos participantes. Posteriormente, Sampson e Weiss [1995] estenderam o trabalho de Eglese e Rand [1987] para levar em consideração a capacidade das salas.

Ao longo dos anos, outros trabalhos acerca dessa temática foram publicados. No entanto, apesar de tais trabalhos possuírem um propósito em comum, ou seja, montar a programação das sessões, tanto as abordagens do problema quanto os meios adotados para resolvê-lo são diferentes. Eles propõem outras formas de se efetuar a alocação dos trabalhos às sessões levando em consideração diversos fatores. Por exemplo, em [Ibrahim et al., 2008] uma sessão contém trabalhos de três áreas diferentes, já em [Tanaka et al., 2002] são utilizadas as palavras-chave dos trabalhos para que as sessões contenham trabalhos com temas semelhantes. Outro exemplo pode ser encontrado em [Vangerven et al., 2017], no qual a alocação é feita com o intuito de minimizar a alternância dos participantes entre sessões diferentes com base em suas preferências. Nota-se que parte dessa divergência surge de necessidades específicas de cada evento.

Um evento que ocorre anualmente no Brasil é o Simpósio Brasileiro de Pesquisa Operacional (SBPO). Trata-se de um evento bem consolidado e conta com 49 edições realizadas por todo o país. Atualmente, a programação de suas sessões técnicas do SBPO é organizada de forma manual e por esse motivo costuma-se levar dias para se obter uma solução viável e de qualidade aceitável. Não foi identificada uma abordagem na literatura que atenda as características do SBPO. Portanto, nesse contexto, o presente trabalho visa desenvolver uma solução para o Problema de Programação de Sessões Técnicas de Conferências (PPSTC) no contexto do SBPO ($\text{PPSTC}_{\text{SBPO}}$).

Duas formulações matemáticas são propostas para resolver o problema em questão. Porém, dada a sua complexidade, o emprego de tais formulações se limitam a instâncias com um número limitado de artigos, tipicamente bem inferior ao número de trabalhos apresentados no SBPO. Desta forma, para resolver instâncias de porte maior, propõe-se um algoritmo híbrido que combina elementos das meta-heurísticas *Iterated Local Search* (ILS) e *Simulated Annealing* (SA), além de fazer uso de modelos de programação inteira baseado em particionamento de conjuntos e alocação generalizada. Os resultados obtidos mostram que o algoritmo desenvolvido é capaz de encontrar soluções de qualidade bastante superior àquelas geradas manualmente, além de alcançar as soluções ótimas provadas por ao menos um dos modelos em instâncias menores geradas artificialmente com base do cenário real.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 apresenta a descrição do problema. A Seção 3 introduz as formulações matemáticas propostas para o problema abordado. A Seção 4 descreve o algoritmo híbrido proposto. A Seção 5 contém os resultados dos

experimentos realizados. A Seção 6 apresenta as considerações finais.

2. Descrição do problema

Devido aos diferentes aspectos e objetivos tratados pelos trabalhos na literatura, é difícil encontrar uma definição formal unificada para o problema. Dessa forma, será apresentada a seguir uma definição para a versão do problema abordada neste trabalho.

Considere um conjunto de trabalhos $P = \{1, \ldots, n\}$, um conjunto de dias D, um conjunto de horários H não sobrepostos (e.g., 14–16h e 16–18h são dois horários distintos e sem sobreposição), um conjunto de sessões S, um conjunto de temas T e um conjunto de autores A. Além disso, sejam $P_a \subseteq P$ o conjunto de trabalhos do autor $a \in A$ e $P_t \subseteq P$ o conjunto de trabalhos que possuem o tema $t \in T$. Uma sessão $s \in S$ acontece no dia $d_s \in D$, no horário $h_s \in H$, e possui capacidades mínima m_s e máxima M_s, as quais estão associadas ao número de trabalhos que ela comporta. Para $d \in D$ e $h \in H$, defina: $S_d = \{s \in S : d_s = d\}$, o conjunto de sessões do dia d; $S_h = \{s \in S : h_s = h\}$, o conjunto de sessões do horário h; e $S_d^h = \{s \in S : d_s = d \wedge h_s = h\}$, o conjunto de sessões do dia d e horário h.

O PPSTC$_{\text{SBPO}}$ consiste em alocar os trabalhos, elementos do conjunto P, às sessões, elementos do conjunto S, respeitando os limites de capacidade de cada sessão. O objetivo é encontrar a alocação que tenha um benefício máximo, sendo o benefício de dois trabalhos $i, j \in P$ estarem juntos em uma mesma sessão denotado por b_{ij}. Esse valor é calculado de acordo com a quantidade de temas que os trabalhos i e j possuem em comum. As informações sobre horário, dia e capacidade de cada sessão são dados pré-definidas pelo evento.

Para evitar a apresentação paralela de muitos trabalhos de um mesmo tema $t \in T$, define-se um limite superior M_h para a quantidade de trabalhos do tema t alocados na faixa de horário $h \in H$ em um mesmo dia, independente da sessão. Da mesma forma, para evitar que as apresentações de um determinado tema $t \in T$ sejam esgotadas em um único dia e para promover um maior balanceamento de temas, define-se um limite superior M_d para a quantidade de artigos de um mesmo tema que pode ser alocada em um mesmo dia $d \in D$.

O PPSTC$_{\text{SBPO}}$ é um problema NP-Difícil, pois o Problema de Particionamento de Grafos em *Cliques* (PPGC) estudado por Grötschel e Wakabayashi [1990] pode ser a ele reduzido. No PPGC, é dado um grafo completo $G = (V, E)$ no qual cada aresta $\{i, j\}$ possui um peso w_{ij}. O objetivo é encontrar um particionamento de V tal que a soma dos pesos das arestas de cada partição seja maximizada. Na redução do PPGC para o PPSTC$_{\text{SBPO}}$, cria-se uma instância I do PPSTC$_{\text{SBPO}}$ na qual existem $|V|$ sessões, cada uma representando uma partição (possivelmente vazia), e $|V|$ trabalhos, cada um representando um vértice distinto de G. As sessões são mutuamente não simultâneas e possuem capacidades não restritivas (1 e $|V|$). A cada trabalho, atribui-se um tema e um autor distinto. Dessa forma, qualquer alocação respeita as restrições relativas aos autores e aos limites de trabalhos de um certa tema. Por fim, o benefício de se alocar dois trabalhos i e j em uma mesma sessão é igual ao peso da aresta de G cujos extremos são os vértices representados por i e j. Sendo assim, a solução ótima de I contém uma alocação de trabalhos a sessões que é equivalente a uma solução ótima do PPGC para G.

3. Formulações Matemáticas

A presente seção descreve as duas formulações matemáticas, aqui denominadas de F1 e F2, propostas neste trabalho.

3.1. Formulação F1

Para a formulação F1, foram utilizados três tipos de variáveis de decisão. Defina x_{ij}^s como sendo uma variável binária que assumirá o valor 1 quando dois trabalhos $i, j \in P$, $i < j$, estiverem juntos em uma mesma sessão $s \in S$ e y_{is} como sendo uma variável binária que assumirá o valor 1 apenas quando o trabalho $i \in P$ estiver na sessão $s \in S$. Finalmente, seja u_s uma variável binária que terá o valor 1 quando a sessão $s \in S$ for utilizada. O modelo de programação linear inteira implementado é apresentado a seguir.

$$\text{Max} \sum_{i \in P} \sum_{\substack{j \in P \\ i<j}} \sum_{s \in S} b_{ij} x_{ij}^{s} \tag{1}$$

Sujeito a:

$$\sum_{s \in S} y_{is} = 1, \qquad \forall i \in P \tag{2}$$

$$u_s m_s \leq \sum_{i \in P} y_{is} \leq M_s, \qquad \forall s \in S \tag{3}$$

$$\sum_{i \in P_a} \sum_{s \in S_d^h} y_{is} \leq 1, \qquad \forall a \in A, \forall h \in H, \forall d \in D \tag{4}$$

$$\sum_{i \in P_t} \sum_{s \in S_d^h} y_{is} \leq M_h, \qquad \forall t \in T, \forall h \in H, \forall d \in D \tag{5}$$

$$\sum_{i \in P_t} \sum_{s \in S_d} y_{is} \leq M_d, \qquad \forall t \in T, \forall d \in D \tag{6}$$

$$2x_{ij}^{s} \leq y_{is} + y_{js}, \qquad \forall i \in P, \forall j \in P, i < j, \forall s \in S \tag{7}$$

$$u_s \geq y_{is}, \qquad \forall s \in S, \forall i \in P \tag{8}$$

$$x_{ij}^{s} \in \{0, 1\}, \qquad \forall i \in P, \forall j \in P, i < j, \forall s \in S \tag{9}$$

$$y_{is} \in \{0, 1\}, \qquad \forall i \in P, \forall s \in S \tag{10}$$

$$u_s \in \{0, 1\}, \qquad \forall s \in S. \tag{11}$$

A função objetivo (1) maximiza o benefício de dois trabalhos estarem juntos em uma mesma sessão. As restrições (2) obrigam todos os trabalhos a serem alocados em alguma sessão. As restrições (3) limitam a capacidade da sessão. As restrições (4) evitam que trabalhos de um mesmo autor sejam alocados em sessões paralelas. As restrições (5) e (6) limitam a quantidade de artigos de um mesmo tema em sessões paralelas e em um mesmo dia, respectivamente. As restrições (7) e (8) fazem a relação entre as variáveis de decisão do problema. As restrições (9)–(11) determinam a natureza das variáveis de decisão.

É possível calcular o número mínimo L_i e o máximo L_s de arestas em uma solução em função do número de artigos e das capacidades das sessões disponíveis, de uma maneira similar à empregada por [Bulhões et al., 2017]. Com isso, pode-se incluir a seguinte restrição com o intuito de fortalecer a relaxação linear da formulação:

$$L_i \leq \sum_{i \in P} \sum_{\substack{j \in P \\ i<j}} \sum_{s \in S} x_{ij}^{s} \leq L_s. \tag{12}$$

3.2. Formulação F2

Considere K um conjunto de tipos de *clusters*, $S_k \subseteq S$ o conjunto das sessões do tipo $k \in K$. O limites m_k e M_k indicam os tamanhos mínimo e máximo que um *cluster* do tipo $k \in K$ pode assumir. Seja z_{ij} uma variável binária que assumirá o valor 1 quando os trabalhos $i, j \in P, i < j$ estiverem juntos em um mesmo *cluster*. Defina ainda w_{ijk} como sendo outra variável binária que assumirá o valor 1 quando os trabalhos $i, j \in P, i < j$, estiverem juntos em um *cluster* do tipo $k \in K$ que seja liderado pelo trabalho i. O líder de um *cluster* é o trabalho de menor índice nele contido. A formulação F2 pode ser escrita da seguinte forma.

$$\text{Max} \sum_{i \in P} \sum_{\substack{j \in P \\ i<j}} b_{ij} z_{ij} \tag{13}$$

Sujeito a:

$$z_{ij} + z_{jv} - z_{iv} \leq 1, \qquad \forall i \in P, \forall j \in P, \forall v \in P, i < j < v, \tag{14}$$

$$z_{ij} - z_{jv} + z_{iv} \leq 1, \qquad \forall i \in P, \forall j \in P, \forall v \in P, i < j < v \tag{15}$$

$$-z_{ij} + z_{jv} + z_{iv} + \sum_{k \in K} w_{vvk} \leq 1, \qquad \forall i \in P, \forall j \in P, \forall v \in P, i < j < v \tag{16}$$

$$\sum_{k \in K} \sum_{\substack{i \in P \\ i \leq j}} w_{ijk} = 1, \qquad \forall j \in P \tag{17}$$

$$\sum_{j \in P} w_{jjk} \leq |S_k|, \qquad \forall k \in K \tag{18}$$

$$m_K w_{iik} \leq \sum_{\substack{j \in P \\ i \leq j}} w_{ijk} \leq k w_{iik}, \qquad \forall k \in K, \forall i \in P \tag{19}$$

$$w_{ijk} \leq w_{iik}, \qquad \forall k \in K, \forall i \in P, \forall j \in P, i < j \tag{20}$$

$$L_i \leq \sum_{i \in P} \sum_{\substack{j \in P \\ i < j}} z_{ij} \leq L_s \tag{21}$$

$$\sum_{k \in K} w_{jjk} + z_{ij} \leq 1, \qquad \forall i \in P, \forall j \in P, i < j \tag{22}$$

$$\sum_{k \in K} w_{jjk} + \sum_{\substack{i \in P \\ i < j}} z_{ij} \geq 1, \qquad \forall j \in P \tag{23}$$

$$z_{ij} \geq \sum_{k \in K} w_{ijk}, \qquad \forall i \in P, \forall j \in P, i < j \tag{24}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in P, \forall j \in P, i < j \tag{25}$$

$$w_{ijk} \in \{0, 1\}, \qquad \forall i \in P, \forall j \in P, i < j, \forall k \in K. \tag{26}$$

A função objetivo (13) busca maximizar o benefício total da alocação dos trabalhos. As restrições (14)–(16) proíbem a existência de um caminho induzido formado por três vértices. Em outras palavras, elas garantem que, dado que os trabalhos i e j e os trabalhos j e v estão na mesma sessão, então os trabalhos i e v também devem estar na mesma sessão. A inserção do termo $\sum_{k \in K} w_{vvk}$ nas restrições (16) é um fortalecimento que segue o trabalho de Bulhões et al. [2017]. As restrições (17) obrigam que todos os trabalhos sejam alocados em exatamente um *cluster*. As restrições (18) determinam que existem, no máximo, $|S_k|$ *clusters* de tamanho k, enquanto que as restrições (19) garantem que os tamanhos máximo e mínimo de um determinado tipo de *cluster* sejam respeitados. As restrições (20) asseguram que o trabalho i só pode representar o trabalho j caso i seja líder de *cluster*. As restrições (21) limitam a quantidade de arestas da solução. As restrições (22)–(23) determinam quando um trabalho é líder de *cluster*. As restrições (24) forçam a aresta $\{i, j\}$ a existir caso o trabalho i seja o líder do *cluster* que contém o trabalho j. As restrições (25)–(26) estão associadas ao domínio das variáveis de decisão.

A formulação dada por (13)–(26) é incompleta pois permite inviabilidades. Todavia, verificou-se empiricamente que a relaxação linear da formulação tende a ser mais forte que aquela obtida por F1 nas instâncias testadas. Para garantir a viabilidade da solução, implementou-se um verificador que checa em tempo de execução se a solução incumbente encontrada pelo modelo acima é viável ou não. Em caso de inviabilidade, a solução é descartada. Este tipo de abordagem é conhecida como *branch-and-reject* [Müller-Merbach, 1975]. O verificador encontra-se descrito a seguir.

Seja C o conjunto de *clusters* encontrados pela solução incumbente de F2, H_d o conjunto de horários do dia $d \in D$, H_i o conjunto de horários em que o *cluster* $i \in C$ pode ser alocado, C_a o conjunto de *clusters* que contém trabalhos do autor $a \in A$, C_h o conjunto de *clusters* que podem

ser alocados no horário $h \in H$ e C_{ht} o conjunto de *clusters* que podem ser alocados no horário $h \in H$ e possuem o tema $t \in T$. A quantidade de vezes que o tema $t \in T$ aparece no *cluster* $i \in C$ é dada por Q_{ti}. $NumS_h$ indica o número de sessões no horário $h \in H_d$. A quantidade de trabalhos em um *cluster* é dado por $NumP_i$ e a quantidade máxima de trabalhos que podem ser alocados em um horário $h \in H_d$ é dado por $MaxP_h$. As variáveis binárias ξ_{idh} indicam se o *cluster* $i \in C$ está alocado no dia $d \in D$, no horário $h \in H_d$. O modelo que verifica se uma solução incumbente de F2 é viável pode ser escrito da seguinte maneira.

$$\text{Min } 0 \tag{27}$$

Sujeito a:

$$\sum_{d \in D} \sum_{h \in H_i} \xi_{idh} = 1, \qquad \forall i \in C \tag{28}$$

$$\sum_{i \in C_a} \xi_{idh} \leq 1, \qquad \forall a \in A, \forall d \in D, \forall h \in H_d \tag{29}$$

$$\sum_{i \in C_{ht}} Q_{ti} \xi_{idh} \leq M_h, \qquad \forall t \in T, \forall d \in D, \forall h \in H_d \tag{30}$$

$$\sum_{h \in H_d} \sum_{i \in C_{ht}} Q_{ti} \xi_{idh} \leq M_d, \qquad \forall t \in T, \forall d \in D \tag{31}$$

$$\sum_{i \in C_h} \xi_{idh} \leq NumS_h, \qquad \forall d \in D, \forall h \in H_d \tag{32}$$

$$\sum_{i \in C_h} NumP_i \xi_{idh} \leq MaxP_h, \qquad \forall d \in D \in, \forall h \in H_d \tag{33}$$

$$\xi_{idh} \in \{0, 1\}, \qquad \forall i \in C, \forall d \in D, \forall H \in H_d. \tag{34}$$

O verificador de viabilidade apenas confirma se os *clusters* respeitam as restrições do modelo, e portanto, não possui uma função objetivo. As restrições (28) impõem que cada *cluster* deve ser alocado exatamente em um dia e em um horário. As restrições (29) evitam que um mesmo autor tenha trabalhos em sessões paralelas. As restrições (30) e (31) limitam a quantidade de temas por horário e por dia respectivamente. As restrições (32) limitam a quantidade de *clusters* por faixa de horário. As restrições (33) definem um limite superior para quantidade de trabalhos por faixa de horário. As restrições (34) definem a natureza das variáveis de decisão.

4. Algoritmo proposto

O algoritmo híbrido proposto para a resolução do problema de programação de sessões técnicas de eventos acadêmicos, denominado $\text{HILS}_{\text{SBPO}}$, será apresentado nesta seção. O método baseia-se no *framework* da meta-heurística *iterated local search* (ILS) e utiliza em sua busca local o procedimento baseado em *Variable Neighborhood Descedent* (VND) [Mladenović e Hansen, 1997], além de um procedimento inspirado na meta-heurística *simulated annealing* (SA)[Kirkpatrick et al., 1983] como critério de aceitação de uma solução a ser perturbada.

Conforme apresentado no Algoritmo 1, o algoritmo híbrido *mult-start* executa I_{Max} iterações (linhas 4–25). Para cada iteração uma solução inicial é gerada por um método guloso e aleatório (linha 5). Em seguida, o laço interno do algoritmo (linhas 9–22) tenta melhorar a solução inicial gerada aplicando um procedimento de busca local (linha 10) combinado com um mecanismo de pertubação (linha 20). Esse laço continua até atingir I_{ILS} iterações sem melhora. Note que sempre que uma solução de melhor custo é encontrada o contador $iter_{ILS}$ é reiniciado (linhas 13–15).

Na etapa de pertubação, também são consideradas soluções que não levam a melhoria. Assim, para criar uma maior diversificação, ela pode ser feita sobre uma solução de pior custo, de maneira análoga a meta-heurística SA. As soluções que não levam a melhora serão aceitas com

probabilidade em função do parâmetro T. Inicialmente T é inicializado como sendo T_0 e posteriormente reduzido a cada iteração em uma razão $\Delta \in (0, 1)$, diminuindo consequentemente a probabilidade de aceitação das soluções de pior benefício.

No HILS$_{\text{SBPO}}$, cada solução é composta por um conjunto de *clusters*, e cada *cluster* contém um conjunto de artigos. Os *clusters* são uma representação de cada sessão do problema. Assim, para cada sessão existe um *cluster* equivalente com a mesma capacidade. A cada vez que uma busca local é realizada, os *clusters* associados a solução podem ser adicionados à um *pool* de *clusters* (linha 11). O método adiciona apenas *clusters* que não foram gerados por outras soluções, ou seja, *clusters* diferentes dos que já estão no *pool*.

Ao final de cada iteração do laço principal, a solução é atualizada em caso de melhora e se for viável (linhas 23–25). Neste caso, utiliza-se o mesmo procedimento descrito na Seção 3.2 tendo como entrada os *clusters* encontrados pelo algoritmo naquela iteração.

Quando ambos os laços são finalizados, o algoritmo realiza sua última etapa. Trata-se de um procedimento baseado em particionamento de conjuntos para encontrar a melhor combinação dos *clusters* que foram armazenados no *pool* (linha 26). Novamente o verificador é chamado em tempo de execução para checar se as soluções incumbentes encontradas são viáveis.

Algoritmo 1 HILS$_{\text{SBPO}}$

```
1: Procedimento HILS_SBPO(I_Max, I_ILS, T_0, α)
2: f* ← 0
3: ClusterPool ← ∅
4: para i := 1 até I_Max faça
5:     s ← GeraSolucaoInicial()
6:     s' ← s
7:     iter_ILS ← 0
8:     T ← T_0
9:     enquanto iter_ILS ≤ I_ILS faça
10:        s ← BuscaLocal(s)
11:        ClusterPool ← AtualizaClusterPool(s, ClusterPool)
12:        Δ ← f(s) − f(s')
13:        se Δ > 0 então
14:            s' ← s
15:            iter_ILS ← 0
16:        senão
17:            x ∈ [0, 1]
18:            se T > 0 e x < e^{−Δ/T} então
19:                s' ← s
20:        s ← Perturba(s')
21:        iter_ILS ← iter_ILS + 1
22:        T ← T × α
23:    se f(s') > f* e VerificaViabilidade(s') = true então
24:        s* ← s'
25:        f* ← f(s')
26: s* ← SP(s*, ClusterPool)
27: retorne s*
28: fim HILS_SBPO.
```

4.1. Geração da solução inicial

A solução inicial é gerada a partir de um algoritmo guloso e aleatório. Inicialmente, para cada *cluster* é atribuído um trabalho de forma aleatória. Os outros trabalhos são inseridos iterativamente em uma segunda etapa. Nessa etapa cada trabalho é alocado no *cluster* que obter o melhor benefício, ou seja, que tenha a melhor soma *intracluster*. Se ao final existirem *clusters* que não atingiram a capacidade mínima, os trabalhos contidos neles são removidos e realocados novamente. Se mesmo assim ainda existirem *clusters* com trabalhos abaixo do limite, esses trabalhos serão realocados em um *cluster dummy* com peso negativo. O peso negativo faz com que durante a fase de busca local os trabalhos contidos no *cluster dummy* sejam alocados em outros *clusters*.

4.2. Busca local

A fase de busca local é realizada através de um método baseado no procedimento VND com escolha aleatória na ordem das vizinhanças.

Cada vizinhança realiza um tipo de movimento que leva ao vizinho viável de maior custo. Para a resolução do problema utilizou-se duas estruturas de vizinhança que serão descritas a seguir:

- *Move* — $N^{(1)}$: Um *cluster* que contenha ao menos um trabalho é selecionado e o trabalho é realocado em outro *cluster* que ainda esteja abaixo de sua capacidade máxima.
- *Swap* — $N^{(2)}$: Dois *clusters* não vazios trocam um trabalho entre si.

Utilizou-se a estratégia de melhor aprimorante (*best improvement*) durante a busca em cada vizinhança. Além disso, para melhorar a eficiência da busca local, foi empregada a abordagem conhecida como *don't look bits* [Bentley, 1992], que examina apenas movimentos que possuem potencial de melhora, descartando outros em que já se sabe de antemão que não levarão a melhora por já terem sido explorados anteriormente. Para avaliar os movimentos em tempo constante, foram empregadas estruturas de dados auxiliares que armazenam algumas informações como o benefício de um *cluster* ao se remover/inserir um trabalho. As atualizações de tais estruturas são feitas em tempo $O(n^2)$. Como o tamanho de cada vizinhança é da ordem de $O(n^2)$ e cada movimento é avaliado em tempo $O(1)$, então a complexidade total do procedimento de busca local é $O(n^2)$.

4.3. Mecanismos de pertubação

Dois mecanismos de pertubação são utilizados. A cada iteração é selecionada aleatoriamente uma das seguintes pertubações:

- *Multiple moves* — $P^{(1)}$: De modo similar ao procedimento da busca local, no entanto, tanto o *cluster* de origem quanto o de destino são escolhidos de forma aleatória e o procedimento se repete de 2 a 5 vezes.
- *Multiple swaps* — $P^{(2)}$: De modo similar ao procedimento da busca local, no entanto, ambos os *clusters* são escolhidos de forma aleatória e o procedimento se repete de 2 a 5 vezes.

4.4. Abordagem baseada em particionamento de conjuntos

Conforme discutido anteriormente, ao final do algoritmo $\text{HILS}_{\text{SBPO}}$ foi utilizada uma abordagem baseada em particionamento de conjuntos (linha 26). Considere C um conjunto de *clusters*, cada qual com um benefício $b'_j, j \in C$, associado e K um conjunto dos tamanhos de cada *cluster* (equivalente ao tamanho máximo das sessões). Para cada trabalho $i \in P$ existe um subconjunto C_i de *clusters* que contém aquele trabalho, e para cada tamanho de *cluster* existe um conjunto S_k de sessões de tamanho k e um conjunto C_k de *clusters* aptos a serem alocados em uma sessão de tamanho $k \in K$.

O intuito do uso do particionamento de conjuntos é encontrar os *clusters* armazenados no *pool* de forma a maximizar o benefício total. Assuma λ_j com uma variável binária que indica quando um *cluster* $j \in C$ é utilizado.

$$\text{Max} \sum_{j \in C} b'_j \lambda_j \tag{35}$$

Sujeito a:

$$\sum_{j \in C_i} \lambda_j = 1, \qquad \forall i \in P \tag{36}$$

$$\sum_{j \in C_k} \lambda_j \leq S_k, \qquad \forall k \in K \tag{37}$$

$$\sum_{j \in C} \lambda_j \leq |S| \tag{38}$$

$$\lambda_j \in \{0,1\}, \qquad \forall j \in C. \tag{39}$$

A função objetivo (35) é maximizada pela escolha dos *clusters* de melhor benefício. As restrições (36) obrigam que cada trabalho só deve estar em um *cluster*. As restrições (37) limitam a quantidade de *clusters* de tamanho $k \in K$ que podem entrar na solução. A restrição (38) limita a quantidade de *clusters*. As restrições (39) determinam a natureza da variável de decisão.

5. Resultados computacionais

Os experimentos computacionais foram realizados em um computador com processador Intel® Core™ i7 com 1.80 GHz, 8.0 GB de memória RAM e sistema operacional Windows 10. A linguagem de programação utilizada nas implementações dos modelos matemáticos e do método heurístico foi C# e o *solver* de programação linear inteira adotado foi o CPLEX versão 12.7.

Para o Algoritmo 1, o número de iterações I_{Max} recebeu o valor 30. Já para o número de pertubações I_{Ils}, foi atribuído o valor $\max(100, n)$, em que n é a quantidade de trabalhos. Para o parâmetro α, foi utilizado o valor de 0,80 e, para T_0, foi adotado o valor 1000. Esses valores foram calibrados de forma empírica por meio de testes preliminares.

O valor do benefício b_{ij} de se alocar dois trabalhos $i \in P$ e $j \in P$ na mesma sessão é determinado pela quantidade de temas em comum entre eles, a saber: 10, 50 e 100 para um, dois e três temas em comum, respectivamente.

5.1. Resultados com instâncias de pequeno porte

Para testar os modelos e o algoritmo híbrido, criou-se um gerador de instâncias que toma como base o evento do SBPO 2016, visto que os modelos exatos não são capazes de resolver instâncias de tamanho real. O gerador recebe como parâmetro um valor que representa a porcentagem de proporcionalidade que se deseja criar a instância. Portanto, é possível criar instâncias menores mantendo aspectos da instância real. Como critério de parada para os modelos, foi estabelecido o limite de uma hora para o tempo de execução.

Para averiguar o desempenho das formulações e do algoritmo híbrido, foram gerados três conjuntos de instâncias de tamanhos diferentes. Para cada conjunto, foram geradas 5 instâncias. O tamanho de cada instância representa a quantidade de trabalhos existentes nela. O valor percentual representa o tamanho das instâncias em relação ao SBPO 2016, que possui 288 trabalhos.

As Tabelas 1 e 2 apresentam os resultados obtidos com as formulações F1 e F2. A primeira coluna indica o tamanho das instâncias em termo da quantidade de trabalhos. A coluna UB_R indica o *upper bound* da raiz. As colunas UB e LB indicam, respectivamente, o *upper bound* e *lower bound* da solução. A coluna Gap(%) exibe os *gaps* entre o LB e o UB. T(s) indica o tempo de execução em segundos. A Tabela 3 apresenta os resultados obtidos pelo algoritmo HILS_SBPO, onde $\text{Gap}(\%)_\text{UB}$ indica o desvio percentual entre o valor do benefício encontrado pelo algoritmo híbrido (LB) e o melhor UB obtido pelas formulações.

A formulação F1 atinge todos os ótimos para o primeiro conjunto de instâncias de tamanho 14, mas, ao passar para o segundo conjunto, nem sempre consegue convergir dentro do tempo limite. Para o terceiro conjunto de testes, a F1 não consegue resolver nenhuma instância dentro do limite de tempo estabelecido. A formulação F2 consegue resolver, dentro do tempo limite, todas as instâncias dos dois primeiros conjuntos de testes, mas falha em determinar a solução ótima na maioria das instâncias do terceiro conjunto. O algoritmo híbrido foi capaz de obter o ótimo para as instâncias resolvidas por pelo menos uma das formulações. Nota-se também que, apesar do algoritmo HILS_SBPO apresentar um tempo maior de execução para as instâncias menores, a medida que o tamanho da instância aumenta, seu tempo de execução vai se tornando mais escalável.

5.2. Resultados com instâncias reais

Para a comparação com instâncias reais, utilizou-se as instâncias do SBPO dos anos de 2016 e 2017. Os dados foram obtidos através dos *sites* dos eventos e de planinhas com a listagem dos trabalhos cedidas por um dos organizadores. O SBPO de 2016 teve um total de 288 trabalhos, 22 temas e 93 sessões. Já o SBPO 2017 é uma instância de maior porte, tendo contado com 324

Tabela 1: Resultados computacionais para formulação F1

Tamanho	UB_R	UB	LB	Gap(%)	T(s)
14 (5%)	571.00	523.00	523.00	0.00	0.23
	49.00	44.00	44.00	0.00	0.33
	49.00	42.00	42.00	0.00	0.47
	118.00	116.00	116.00	0.00	0.19
	147.00	147.00	147.00	0.00	0.44
29 (10%)	91.00	87.00	87.00	0.00	387.41
	64.00	63.99	62.00	3.23	3600.00
	83.00	81.99	80.00	2.50	3600.00
	271.00	226.00	225.00	0.44	3600.00
	397.00	349.00	349.00	0.00	2172.00
43 (15%)	290.00	245.00	241.00	1.66	3600.00
	227.00	209.00	205.00	1.95	3600.00
	362.00	344.00	341.00	0.88	3600.00
	173.00	164.00	162.00	1.23	3600.00
	236.00	182.00	179.00	1.68	3600.00

Tabela 2: Resultados computacionais para formulação F2

Tamanho	UB_R	UB	LB	Gap(%)	T(s)
14 (5%)	529.94	523.00	523.00	0.00	0.96
	44.78	44.00	44.00	0.00	0.19
	43.70	42.00	42.00	0.00	1.20
	116.00	116.00	116.00	0.00	0.08
	147.00	147.00	147.00	0.00	0.50
29 (10%)	89.08	87.00	87.00	0.00	17.61
	63.00	62.00	62.00	0.00	36.47
	81.91	80.00	80.00	0.00	65.01
	270.00	225.00	225.00	0.00	420.00
	347.00	349.00	349.00	0.00	46.00
43 (15%)	287.63	243.49	242.00	0.74	3600.00
	222.5	208.00	206.00	0.97	3600.00
	359.67	354.90	337.00	5.31	3600.00
	171.67	163.00	163.00	0.00	2527.80
	206.53	194.73	176.00	10.67	3600.00

Tabela 3: Resultados computacionais para o algoritmo $\text{HILS}_{\text{SBPO}}$

Tamanho	**LB**	**Gap$_{\text{UB}}$(%)**	**T(s)**
14 (5%)	523.00	0.00	7.40
	44.00	0.00	8.41
	42.00	0.00	10.43
	116.00	0.00	10.13
	147.00	0.00	8.47
29 (10%)	87.00	0.00	16.30
	62.00	0.00	16.78
	80.00	0.00	14.94
	225.00	0.00	17.89
	349.00	0.00	21.95
43 (15%)	242.00	0.49	24.83
	206.00	0.46	40.10
	341.00	0.87	30.79
	163.00	0.00	35.40
	179.00	1.64	26.50

trabalhos, 22 temas e 95 sessões. Nesses dois eventos, a programação das sessões técnicas foi realizada de forma manual. Para comparação com a solução proposta, a função objetivo foi calculada para as duas edições do evento. As funções objetivo da solução manual das edições de 2016 e 2017 foram 718 e 1926, respectivamente. O algoritmo híbrido, por sua vez, atingiu os valores de 3048 e 5495, respectivamente. O tempo consumindo pelo algoritmo híbrido nas instância de 2016 de 2017 foi cerca de 50 e 60 minutos, respectivamente.

De um modo geral, a solução manual para o problema tenta apenas agrupar os trabalhos por um tema principal. Por outro lado, o algoritmo híbrido leva em consideração todos os temas do artigo, gerando consequentemente sessões com alto benefício, o que possivelmente justifica o maior valor da função objetivo. Vale salientar, também, que a solução obtida está livre de choques de horários para trabalhos de um mesmo autor, o que não é necessariamente verdade na solução manual, além de balancear a distribuição dos temas.

6. Considerações finais

Este trabalho apresentou uma abordagem para o problema de programação de sessões técnicas de eventos acadêmicos, em particular, o caso do SBPO. Para resolvê-lo, foram propostas formulações matemáticas, além de um algoritmo híbrido que combina elementos das metaheurísticas ILS e SA com um procedimento exato baseado em um modelo de particionamento de conjuntos. As formulações não escalam bem na prática e, por isso limitam-se a instâncias de pequeno porte. Já o algoritmo híbrido foi capaz de atingir os ótimos para tais instâncias, além de gerar soluções viáveis, superiores aquelas geradas manualmente, para instâncias reais que foram obtidas a partir dos eventos realizados em 2016 e 2017. Apesar das soluções propostas terem sido voltadas para um caso específico, pode-se com pouca ou nenhuma adaptação torná-las aptas a organizar outros eventos, visto que muitos deles têm características semelhantes ou iguais ao SBPO.

Referências

Bentley, J. J. (1992). Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411.

Bulhões, T., de Sousa Filho, G. F., Subramanian, A., e dos Anjos F. Cabral, L. (2017). Branch-and-cut approaches for p-cluster editing. *Discrete Applied Mathematics*, 219:51 – 64. ISSN 0166-218X.

Eglese, R. W. e Rand, G. K. (1987). Conference seminar timetabling. *The Journal of the Operational Research Society*, 38:591–598.

Grötschel, M. e Wakabayashi, Y. (1990). Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1):367–387.

Ibrahim, H., Ramli, R., e Hassan, M. H. (2008). Combinatorial design for a conference: constructing a balanced three-parallel session schedule. *Journal of Discrete Mathematical Sciences and Cryptography*, 11:305–317.

Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. ISSN 00368075.

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res.*, 24(11): 1097–1100. ISSN 0305-0548.

Müller-Merbach, H. (1975). Modelling techniques and heuristics for combinatorial problems. In *Combinatorial Programming: Methods and Applications. Proceedings of the NATO Advanced Study Institute held at the Palais des Congrès, Versailles, France, 2–13 September, 1974*.

Sampson, S. E. e Weiss, E. N. (1995). Increasing service levels in conference and educational scheduling: A heuristic approach. *Management Science*, 41:1816–1825.

Silva, P. J. et al. (2013). Alocação de sessões de artigos em eventos acadêmicos: Modelo e estudo de caso. *Pesquisa Operacional para o Desenvolvimento*, 6(1):54–66.

Tanaka, M., Mori, Y., e Bargiela, A. (2002). Granulation of keywords into sessions for timetabling conferences. In *Soft Computing and Intelligent Systems SCIS'2002, Tsukuba, Japan, October 2002*.

Thompson, G. M. (2002). Improving conferences through session scheduling preferences. *Cornell Hotel and Restaurant Administration Quarterly*, 43:71–76.

Vangerven, B., Ficker, A. M., Goossens, D. R., Passchyn, W., Spieksma, F. C., e Woeginger, G. J. (2017). Conference scheduling — a personalized approach. *Omega*. ISSN 0305-0483.