

O PROBLEMA DE PROGRAMAÇÃO DE SESSÕES TÉCNICAS DE CONFERÊNCIAS COM AVALIADORES

Felipe Nunes, Carlos Neves, Teobaldo Bulhões, Anand Subramanian

Universidade Federal da Paraíba – Centro de informática

Rua dos Escoteiros, s/n – Mangabeira, João Pessoa – PB, 58055-000

josefsilva@eng.ci.ufpb.br, carlosneves@eng.ci.ufpb.br,

tbulhoes@ci.ufpb.br, anand@ci.ufpb.br

RESUMO

Conferências acadêmicas são realizadas todos os anos a fim de difundir novas descobertas científicas. O Encontro de Iniciação Científica (ENIC) é o principal evento neste escopo sediado na Universidade Federal da Paraíba. A programação do evento é um problema combinatório difícil de se resolver manualmente, pois envolve escolher quais avaliadores devem estar presentes em cada sessão técnica baseando-se em vários fatores. Este trabalho propõe um algoritmo híbrido para a resolução desse problema que combina formulações matemáticas e elementos das meta-heurísticas *Iterated Local Search* e *Simulated Annealing*. O método busca programar o evento de forma que suas sessões possuam trabalhos com temática semelhante e que os avaliadores não fiquem ociosos. Experimentos computacionais com edições anteriores do ENIC demonstram que o algoritmo é eficaz, sendo capaz de encontrar resultados melhores do que as programações manuais para todas as instâncias, principalmente no que diz respeito a reduzir a ociosidade dos avaliadores.

PALAVRAS CHAVE. *Matheuristic*, Programação de conferências acadêmicas, Programação Inteira.

Tópicos: OC - Otimização Combinatória

ABSTRACT

Academic conferences are held every year in order to spread scientific knowledge. The Scientific Initiation Meeting (ENIC) is the main event in this scope hosted at the Federal University of Paraíba. Scheduling the event is a combinatorial problem that is difficult to be solved manually, as it involves choosing which examiners should be present at each technical session based on several factors. This work presents a hybrid algorithm that combines mathematical formulations and elements from the metaheuristics *Iterated Local Search* and *Simulated Annealing*. Our method aims to schedule the event in such a way that its sessions have papers with similar themes, and examiners are not idle. Computational experiments with previous editions of ENIC demonstrate that our algorithm is effective, being able to find better results than the manual scheduling for all instances, especially when it comes to reducing the idleness of the examiners.

KEYWORDS. *Matheuristic*, Academic conference scheduling, Integer Programming.

Paper topics: CO - Combinatorial Optimization

1. Introdução

Todos os anos são realizados diversos eventos com o intuito de difundir o conhecimento e apresentar os avanços realizados por estudos científicos. Na Universidade Federal da Paraíba (UFPB), o principal evento nesse escopo é o Encontro de Iniciação Científica (ENIC), organizado pela Pró-Reitoria de Pesquisa (PROPESQ), no qual os trabalhos de iniciação científica desenvolvidos durante um ano são apresentados e avaliados. Para cada apresentação são designados três avaliadores, o orientador do trabalho e dois avaliadores, que precisam apresentar afinidade com as áreas de conhecimento, ou tópicos, do trabalho apresentado. A maior dificuldade na organização do evento, segundo funcionários da PROPESQ, encontra-se na escolha de avaliadores e da ordem de apresentação dos trabalhos, sendo destinadas cerca de duas semanas para a realização dessa programação. Essa dificuldade advém do caráter combinatório do problema, i.e., seu grande número de soluções possíveis e variáveis envolvidas, como afirma Silva et al. [2013].

No presente contexto, conferências são eventos nos quais trabalhos científicos são apresentados, tal qual o ENIC. A programação dessas apresentações, respeitando as particularidades de cada evento, configura o problema de organização de conferências. Na literatura, há certa diversidade de abordagens de resolução de problemas de organização de conferências, mas Thompson [2002] ressaltava duas principais. Na primeira, a programação é feita sob a perspectiva dos apresentadores, levando-se em conta fatores tais como a disponibilidade destes. Na segunda, por outro lado, o foco é na perspectiva dos participantes, de modo que podem ser considerados, por exemplo, os interesses destes pelos temas do evento. Eglese e Rand [1987] realizaram um dos primeiros trabalhos com essa perspectiva. O problema levantado na organização do ENIC se enquadra na primeira abordagem, por considerar as preferências do apresentador, nesse caso o orientador. Levando em conta, ainda, o fato de que o orientador de cada trabalho pode vir a ser avaliador de outros trabalhos. Dessa forma, há uma semelhança com a abordagem sugerida por Potthoff e Munger [2003], que consideram o fato de apresentadores poderem desempenhar outras atividades no mesmo evento. Há, ainda, similaridades com a abordagem exposta por Vangerven et al. [2018], que visa minimizar o número de saltos (i.e., saída de uma sessão para outra sem que a primeira tenha terminado) realizados pelos participantes da conferência.

Além de considerar a perspectiva dos apresentadores, também deve-se levar em conta, para o ENIC, os tópicos presentes nos trabalhos apresentados, como sugeriram Tanaka et al. [2002], que utilizaram as palavras-chave dos artigos para realizar a programação. É preferível que as sessões do evento sejam formadas por trabalhos que possuem tópicos em comum, aproximando-se do que propõem Edis e Edis [2013], que aloca tópicos às sessões e preenche cada uma destas com trabalhos do tópico designado. Dito isso, o presente trabalho visa resolver o Problema de Programação de Sessões Técnicas de Conferências com Avaliadores (PPSTCA), tendo como caso de estudo o ENIC.

Para a resolução do PPSTCA é proposto um algoritmo dividido em duas partes com objetivos hierárquicos. Tal estratégia assemelha-se com a adotada por Stidsen et al. [2018], que realiza a programação de conferências através da resolução de subproblemas sequencialmente. A primeira parte consiste em um processo de clusterização, no qual os trabalhos são agrupados de forma a maximizar a quantidade desses que possuem tópicos em comum numa mesma sessão. A segunda parte da formulação é responsável por atribuir os avaliadores de cada trabalho, dentro de cada sessão, de forma a minimizar o tempo ocioso de cada avaliador durante o evento. Para resolver a etapa de clusterização, foi utilizada uma adaptação do algoritmo híbrido proposto por Correia et al. [2018], combinando as meta-heurísticas *Iterated Local Search* (ILS) e *Simulated Annealing* (SA), além de uma formulação matemática responsável por checar a viabilidade da solução encontrada. Para as instâncias do ENIC a segunda parte pôde ser resolvida de forma exata.

O restante deste trabalho é organizado da seguinte forma. A Seção 2 apresenta a descrição do problema. A Seção 3 apresenta uma descrição do algoritmo proposto para solucioná-lo. A Seção 4 apresenta uma discussão sobre os resultados encontrados em instâncias reais do ENIC. Por fim, a Seção 5 apresenta as considerações finais.

2. Descrição do problema

Sejam T um conjunto de trabalhos, D um conjunto de dias, H um conjunto de horários não sobrepostos (e.g., 14–16h, 16–18h), $H_d \subset H$ o conjunto de horários do dia $d \in D$, J um conjunto de temas, $A = \{1, \dots, m\}$ um conjunto de avaliadores e K um conjunto de sessões. Cada sessão $k \in K$ — que ocorre em um horário h_k e dia d_k — está associada a um conjunto de *slots* S_k , que consistem em intervalos de tempo homogêneos e contíguos nos quais o horário h_k do dia d_k está dividido. Sessões diferentes podem ocorrer paralelamente no mesmo horário e dia. Cada avaliador $i \in A$ está associado a um conjunto de trabalhos $T_i \subset T$ dos quais é o único orientador. Cada *slot* $s \in \bigcup_{k \in K} S_k$ deve contemplar a apresentação de um único trabalho. Além disso, considere b_{tw} o benefício de dois trabalhos diferentes $t \in T$ e $w \in T$ serem apresentados na mesma sessão (determinado por fatores como temas em comum, possuírem o mesmo orientador, etc.).

Se, entre quaisquer duas apresentações em que um avaliador $i \in A$ está presente, houver *slots* nos quais ele fica ocioso (não avalia nada), diz-se que houve um “salto”, e que o número de *slots* consecutivos em que ele fica ocioso é o tamanho desse salto. A Figura 1 apresenta um exemplo de uma sessão com apresentações sem saltos, enquanto a Figura 2 apresenta uma sessão com um salto de tamanho 1 realizado pelo avaliador a_1 no *slot* 3.

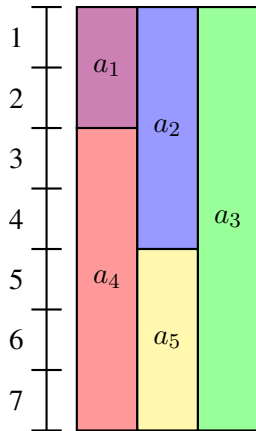


Figura 1: Solução sem saltos.

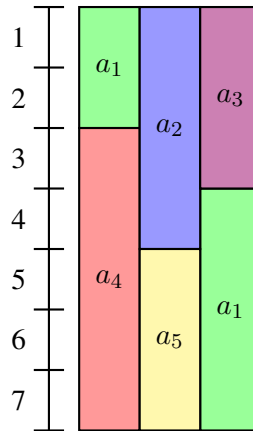


Figura 2: Solução com um único salto.

Assim, o PPSTCA consiste em alocar trabalhos a sessões e avaliadores a *slots*, e é constituído de duas fases: (i) Atribuir trabalhos (e consequentemente seus orientadores) a sessões de forma a maximizar o benefício total (clusterização); e (ii) minimizar, para cada sessão $k \in K$, a ocorrência de “saltos” realizados pelos avaliadores do conjunto A_k , que é o conjunto de avaliadores presentes na sessão $k \in K$ (escalonamento de avaliadores).

A fase de clusterização deve respeitar as seguintes restrições: (i) sessões paralelas não podem ter avaliadores em comum; e (ii) cada sessão $k \in K$ deve conter até $|S_k|$ trabalhos. Durante essa etapa, cada sessão $k \in K$ deve ser associada a um conjunto de trabalhos T_k , e a um conjunto de avaliadores $A_k = O_k \cup E_k$, sendo O_k o conjunto de orientadores de cada trabalho de T_k e E_k um conjunto opcional de avaliadores extra (que não orientam nenhum trabalho do conjunto T_k , mas estão aptos a avaliá-los).

Após a conclusão da fase de clusterização, inicia-se a fase de escalonamento de avaliadores. Para cada sessão $k \in K$, os trabalhos do conjunto T_k e os avaliadores do conjunto A_k devem ser distribuídos nos *slots* do conjunto S_k de forma a minimizar o número de saltos, considerando que: (i) cada trabalho deve ser alocado a um único *slot*; (ii) exatamente três avaliadores devem estar presentes durante a apresentação de cada trabalho, sendo um deles sempre o orientador do trabalho; e (iii) cada avaliador $i \in A_k$ deve avaliar entre c_i^k e d_i^k trabalhos (incluindo aqueles dos quais é orientador). Essa etapa é, ainda, dividida em duas fases: (i) minimizar o tamanho do maior salto realizado h_{max} ; e (ii) minimizar o número de saltos, sabendo-se que o valor h_{max} determinado anteriormente é o maior salto permitido.

O problema de descobrir se $h_{max} = 0$ em uma sessão $k \in K$ (descobrir se a sessão k pode ser organizada de forma a não conter nenhum salto) é NP-Completo, pois o Problema da Partição (PP) pode ser reduzido a ele. Considere um multiconjunto de inteiros $U = \{v_1, \dots, v_n\}$. O PP consiste em encontrar, se possível, uma partição de U em dois conjuntos U_1 e U_2 tais que $\sum_{i \in U_1} v_i = \sum_{i \in U_2} v_i = \sum_{i \in U} v_i / 2$. Para cada inteiro $v_i \in U$, cria-se um avaliador $i \in A_k$ e faz-se com que $c_i^k = d_i^k = v_i$ e $T_i \cap T_k = \emptyset$. Faz-se com que $|S_k| = \sum_{i \in U} v_i / 2$. Cria-se, então, um avaliador artificial $j \in A_k$ e faz-se $c_j^k = d_j^k = |T_j|$ e $T_j = T_k$. Uma solução sem saltos para essa instância só pode ser encontrada se o multiconjunto U puder ser particionado em dois multiconjuntos de soma igual.

3. Algoritmo proposto

Um algoritmo híbrido foi utilizado para resolver o PPSTCA. Na fase de clusterização, uma adaptação do algoritmo proposto por Correia et al. [2018] — elaborada considerando-se as particularidades do ENIC — é utilizada para atribuir trabalhos (e seus orientadores) a sessões de forma a maximizar o benefício (conforme a Seção 3.1). Na fase do escalonamento de avaliadores, que ocorre após a fase de clusterização, precisa-se, para cada sessão $k \in K$, distribuir os avaliadores do conjunto A_k nos *slots* do conjunto S_k de forma a evitar a ocorrência de saltos. Caso o número total de avaliações que cada orientador do conjunto O_k pode realizar na sessão $k \in K$ já seja o suficiente para preencher todos os slots S_k , tem-se que $E_k = \emptyset$. Para distribuir os avaliadores, uma formulação matemática é utilizada em cada sessão (conforme a Seção 3.2).

3.1. Fase de clusterização

Para a resolução da etapa de clusterização do PPSTCA, foram utilizadas heurísticas conhecidas unidas a um método exato para verificação de viabilidade das soluções. O procedimento é baseado no *framework* da meta-heurística *Iterated Local Search* (ILS) [Lourenço et al., 2003]. A etapa de buscas locais é baseada em *Variable Neighborhood Descent* (VND) [Mladenović e Hansen, 1997]. Além disso, uma abordagem inspirada em *Simulated Annealing* (SA) [Kirkpatrick et al., 1983] foi utilizada como critério de aceitação de perturbações nas soluções. O procedimento, descrito no Algoritmo 1, foi denominado *HILS*.

Inicialmente, uma solução é gerada de forma gulosa (linha 4). Em seguida, uma busca local é realizada nessa solução (linha 10). Visando uma maior diversificação, soluções de pior custo também podem ser aceitas com base em uma probabilidade que é função de T — que é iniciado com o valor T_0 — (linhas 12–21), de forma análoga à meta-heurística SA. Após isso, a solução é submetida a uma perturbação (linha 22) e T é diminuído com base em um parâmetro $\alpha \in (0, 1)$ (linha 24), tornando cada vez menores as chances de que uma solução de pior custo seja aceita. O processo recomeça da linha 9 e continua até que falhe I_{ILS} vezes sem escapar de um ótimo local.

A melhor solução da iteração tem sua viabilidade verificada de maneira exata (linhas 25–27). Se ela for viável e melhor que a melhor solução, a melhor solução é atualizada. O algoritmo recomeça da linha 3 e o processo se repete por I_{Max} iterações. Retorna-se a melhor solução viável

Algoritmo 1 *HILS*

```
1: procedimento HILS( $I_{Max}, I_{ILS}, T_0, \alpha$ )
2:    $f^* \leftarrow 0$ 
3:   para  $i := 1$  até  $I_{Max}$  faça
4:      $s \leftarrow \text{GeraSoluçãoInicial}()$ 
5:      $s' \leftarrow s$ 
6:      $s'' \leftarrow s$ 
7:      $iter_{ILS} \leftarrow 0$ 
8:      $T \leftarrow T_0$ 
9:     enquanto  $iter_{ILS} \leq I_{ILS}$  faça
10:       $s \leftarrow \text{BuscaLocal}(s)$ 
11:       $\Delta \leftarrow f(s) - f(s')$ 
12:      se  $\Delta > 0$  então
13:         $s' \leftarrow s$ 
14:         $s'' \leftarrow s$ 
15:         $iter_{ILS} \leftarrow 0$ 
16:      senão
17:         $x \in [0, 1]$ 
18:        se  $T > 0$  e  $x < e^{-\Delta/T}$  então
19:           $s' \leftarrow s$ 
20:        senão
21:           $s' \leftarrow s''$ 
22:         $s \leftarrow \text{Perturba}(s')$ 
23:         $iter_{ILS} \leftarrow iter_{ILS} + 1$ 
24:         $T \leftarrow T \times \alpha$ 
25:      se  $f(s') > f^*$  e  $\text{VerificaViabilidade}(s') = \text{true}$  então
26:         $s^* \leftarrow s'$ 
27:         $f^* \leftarrow f(s')$ 
28:   retorne  $s^*$ 
29: fim HILS
```

encontrada na linha 28. Por fim, cada *cluster* de trabalhos é tratado como uma das sessões do evento. Então, para cada sessão, realiza-se a fase de escalonamento de avaliadores, conforme a sessão 3.2.

3.1.1. Geração de solução inicial

O processo de geração de soluções iniciais é realizado de forma gulosa e aleatória. Primeiro, é atribuído a cada *cluster* um trabalho escolhido aleatoriamente. Em seguida, de forma iterativa, cada trabalho é adicionado ao *cluster* que maximiza seu benefício (i.e., obtém a maior soma *intracluster*). O procedimento é realizado até que todos os trabalhos sejam alocados a um *cluster* de forma que respeitem os limites mínimo e máximo de trabalhos.

3.1.2. Busca local

A busca local é baseada no procedimento VND. Durante a exploração de cada uma das estruturas de vizinhança descritas a seguir (escolhida aleatoriamente), o melhor vizinho (aquele que leva ao maior custo) é escolhido.

- *Move* — $N^{(1)}$: Um *cluster* formado por um ou mais trabalhos é escolhido e o trabalho é realocado em outro *cluster* que ainda não ultrapassou sua capacidade máxima.

- *Swap* — $N^{(2)}$: Dois trabalhos de *clusters* diferentes (e não vazios) são trocados.

3.1.3. Mecanismos de perturbação

A cada iteração, um dos seguintes mecanismos de perturbação é aleatoriamente selecionado:

- *Multiple moves* — $P^{(1)}$: Similar ao movimento *Move*, mas *clusters* de origem e destino são selecionados aleatoriamente. Além disso, o procedimento se repete de 2 a 5 vezes.
- *Multiple swaps* — $P^{(2)}$: Similar ao movimento *Swap*, mas os dois *clusters* são escolhidos de forma aleatória. O procedimento também se repete de 2 a 5 vezes.

3.1.4. Checagem de viabilidade

A checagem de viabilidade da melhor solução de cada uma das I_{Max} iterações foi feita a partir da formulação matemática F1, descrita como segue.

Sejam C o conjunto de *clusters* que compõem a solução encontrada, K_c o conjunto de sessões nas quais o *cluster* $c \in C$ pode ser alocado (i.e., sessões que possam comportar o número de trabalhos do *cluster* $c \in C$), C_i o conjunto de *clusters* que contém trabalhos orientados por um avaliador $i \in A$. Finalmente, a variável binária ξ_k^c assume o valor 1 se o *cluster* $c \in C$ foi alocado na sessão $k \in K$.

$$\text{Min } 0 \quad (1)$$

sujeito a

$$\sum_{k \in K_c} \xi_k^c = 1 \quad c \in C \quad (2)$$

$$\sum_{c \in C_i} \sum_{\substack{k \in K_c \\ d_k = d \wedge h_k = h}} \xi_k^c \leq 1 \quad i \in A, d \in D, h \in H_d \quad (3)$$

$$\sum_{c \in C: k \in K_c} \xi_k^c \leq 1 \quad k \in K \quad (4)$$

$$\xi_k^c \in \{0, 1\} \quad c \in C, k \in K_c. \quad (5)$$

A finalidade do modelo é apenas verificar se a solução encontrada respeita as restrições apresentadas, não sendo necessária uma função objetivo. As restrições (2) impõem que cada *cluster* deve ser alocado a exatamente uma sessão. As restrições (3) garantem que não haja trabalhos de um mesmo avaliador em sessões paralelas. As restrições (4) implicam que no máximo um *cluster* possa ser alocado a uma sessão. As restrições (5) indicam a natureza das variáveis de decisão.

3.2. Escalonamento de avaliadores

Para cada uma das sessões $k \in K$, o escalonamento de avaliadores é feito utilizando-se a formulação matemática F2, descrita a seguir, e que faz uso das seguintes variáveis binárias: (i) x_{is} assume o valor 1 se o avaliador $i \in A_k$ avalia o trabalho no *slot* $s \in S_k$; (ii) y_{is} assume o valor 1 se o avaliador $i \in A_k$ é orientador do trabalho apresentado no *slot* $s \in S_k$; (iii) h_{is} é o tamanho do salto realizado pelo avaliador $i \in A_k$ imediatamente antes de chegar ao *slot* $s \in S_k$; (iv) l_{is} assume o valor 1 se o *slot* $s \in S_k$ não é o último no qual o avaliador $i \in A_k$ se encontra. O *slot*

auxiliar $s = 0$, que não pertence ao conjunto S_k , tem o propósito de ajudar a contabilizar o número de saltos.

$$\text{Min } h_{max} \quad (6)$$

sujeito a

$$\sum_{i \in A_k} x_{is} = 3 \quad s \in S_k \quad (7)$$

$$\max\{c_i^k, |T_k \cap T_i|\} \leq \sum_{s \in S_k} x_{is} \leq d_i^k \quad i \in A_k \quad (8)$$

$$h_{is} \geq h_{is-1} + (1/|S_k|) \sum_{s'=0}^{s-1} x_{is'} - |S_k| x_{is} + l_{is} - 1 \quad i \in A_k, s \in S_k \quad (9)$$

$$h_{max} \geq h_{is} \quad i \in A_k, s \in S_k \quad (10)$$

$$h_{i0} = 0 \quad i \in A_k \quad (11)$$

$$l_{is} \geq x_{is'} \quad i \in A_k, s \in S_k, s' \geq s+1 \quad (12)$$

$$y_{is} \leq x_{is} \quad i \in A_k, s \in S_k \quad (13)$$

$$\sum_{s \in S_K} y_{is} \geq |T_i \cap T_k| \quad i \in A_k \quad (14)$$

$$\sum_{i \in A_k} y_{is} = 1 \quad s \in S_k \quad (15)$$

$$x_{is}, y_{is}, l_{is} \in \{0, 1\} \quad i \in A_k, s \in S_k \quad (16)$$

$$h_{is} \in \mathbb{Z} \quad i \in A_k, s \in S_k \cup \{0\} \quad (17)$$

$$h_{max} \geq 0 \quad (18)$$

A função objetivo (1) busca minimizar o tamanho do maior salto na solução. As restrições (7) garantem que cada *slot* da sessão $k \in K$ possui exatamente três avaliadores. As restrições (8) limitam o número de trabalhos que cada avaliador $i \in A_k$ pode avaliar. As restrições (9)–(12) são responsáveis por contabilizar os saltos através das variáveis h_{is} , além de garantir que h_{max} é o maior salto. As restrições (13)–(15) garantem que cada *slot* da sessão $k \in K$ possui um único orientador, e que ele deve avaliar o próprio trabalho. As restrições (16)–(18) dizem respeito ao domínio das variáveis.

Após a aplicação da formulação F2, verifica-se o valor de h_{max} . Se $h_{max} = 0$, a solução encontrada não possui saltos. Se $h_{max} > 0$, sabe-se que a solução inevitavelmente possui saltos. Então, uma segunda formulação F3 é utilizada, desta vez com o objetivo de minimizar

$$\sum_{i \in A_k} \sum_{s \in S_k} h_{is},$$

para que se obtenha o menor número de saltos possível. Por fim, além de possuir as mesmas restrições que F2, a formulação F3 também está sujeita às restrições

$$h_{is} \leq h_{max} \quad i \in A_k, s \in S_k,$$

que impõem que todos os saltos na sessão $k \in K$ não podem ser maiores que h_{max} .

4. Resultados computacionais

O algoritmo proposto para a resolução do PPSTCA foi implementado com a linguagem de programação C++ e com o auxílio do *framework* de otimização Google OR Tools e do *solver* *Coin-or branch and cut* (CBC) para a implementação dos modelos matemáticos. Os experimentos foram realizados em um computador com processador Intel Core i7 de 3.9 GHz, 16 GB de memória RAM e sistema operacional Ubuntu 16.04 LTS.

Assim como em [Correia et al., 2018], os valores utilizados como parâmetros do Algoritmo 1 foram $I_{Max} = 30$, $I_{ILS} = \max\{100, |T|\}$, $\alpha = 0,8$ e $T_0 = 1000$. Nos casos em que dois trabalhos $t \in T$ e $w \in T$ possuem algum tema em comum, utilizou-se $b_{tw} = 10^{\beta-1} \times \gamma$ para os valores de benefício, sendo β o número de temas que os trabalhos $t \in T$ e $w \in T$ têm em comum. Já o valor γ é igual a 1000 se os trabalhos $t \in T$ e $w \in T$ possuem o mesmo orientador; do contrário γ assume o valor 1. Se os trabalhos $t \in T$ e $w \in T$ não possuem temas ou orientadores em comum, é utilizado $b_{tw} = -10$. Por fim, se $t \in T$ e $w \in T$ não possuem temas em comum mas possuem o mesmo orientador, utiliza-se $b_{tw} = 10000$.

Durante a fase de escalonamento de avaliadores, no caso particular do ENIC, utilizou-se $c_i^k = 0$ e $d_i^k = |S_k|$. Além disso, caso alguma das sessões não possua quórum suficiente (i.e., no mínimo três avaliadores), ela é preenchida importando-se um ou mais avaliadores que possuam afinidade com os tópicos dos trabalhos presentes na sessão. Tais avaliadores são providos de uma base externa, disponibilizada pelo evento.

4.1. Resultados da clusterização

Uma comparação da eficácia da meta-heurística descrita no Algoritmo 1 em relação à solução manual da programação das edições de 2014 a 2018 do ENIC — cedida por organizadores do evento — foi feita. A comparação leva em consideração a distribuição dos trabalhos nas sessões do evento, e faz uso da mesma métrica descrita anteriormente para o cálculo da função objetivo. As instâncias para testes foram geradas a partir da programação de cada edição do evento, obtendo-se uma instância para cada local onde o evento foi realizado e o dia designado para a realização naquele local, sabendo-se o número de salas e os horários disponíveis. Cada sessão pode comportar até 13 apresentações de trabalhos.

A Tabela 1 apresenta os resultados obtidos nos experimentos. Cada linha da tabela representa um dia do evento. As colunas *Ano* e *Tamanho* contêm, respectivamente, o ano em que o dia do evento ocorreu e o número total de trabalhos apresentados no dia. A coluna *Manual* contêm os custos (i.e., o benefício total de todas as sessões) das soluções manuais de cada dia. As colunas *Melhor* e *Média* contêm o melhor custo e o custo médio de cada sessão encontrados pelo Algoritmo 1. Por fim, a coluna *Ganho(%)* contêm o ganho percentual obtido pelo Algoritmo 1 quando comparado às soluções manuais. É possível notar, com os resultados presentes na coluna *Ganho(%)*, que há melhoras nas soluções encontradas pelo Algoritmo 1, em relação às obtidas de forma manual, para todas as instâncias fornecidas.

4.2. Resultados do escalonamento de avaliadores

O escalonamento de avaliadores foi realizado em cada uma das sessões geradas como solução do Algoritmo 1. Novamente, uma comparação é feita com as soluções manuais, desta vez levando em conta o número total de saltos de avaliadores (i.e., quanto tempo passam ociosos entre suas avaliações). Os resultados para esta etapa são apresentados na Tabela 2. Assim como na Tabela 1, os resultados para cada instância foram organizados por ano e local.

A Tabela 2 contém a quantidade total de saltos de avaliadores para cada instância, tanto para as soluções manuais (coluna *Manual*) quanto para as soluções obtidas pela formulações matemáticas descritas na Seção 3.2 (coluna *Formulação*). As colunas *Saltos* contêm o número de

Tabela 1: Comparação entre os benefícios totais obtidos manualmente e pelo Algoritmo 1.

Ano	Tamanho	Manual	Algoritmo 1		
			Melhor	Média	Ganho(%)
2014	96	3224052	3332066	3332066,0	3,35
	188	6007422	6043370	6042899,5	0,59
	251	6705749	6755305	6755251,0	0,74
	221	7012928	7056232	7056166,5	0,62
	136	4840881	4889463	4889418,0	1,00
2015	119	3439980	3467861	3467761,0	0,81
	249	8244142	8297033	8296850,0	0,64
	259	7649096	7714449	7714372,0	0,85
	169	5278741	5412585	5412287,0	2,53
	127	4162610	4300522	4300479,5	3,31
2016	24	800917	800932	800932,0	<0,01
	94	2546256	2577251	2576925,0	1,20
	167	5391373	5429557	5429478,5	0,71
	230	8447639	8531965	8531953,0	1,00
	182	5124404	5161135	5161077,5	0,72
	131	5902189	6020476	6020328,0	2,00
	62	2615560	2625286	2625286,0	0,37
	60	1822026	1834556	1834556,0	0,69
2017	204	17213671	17304148	17304010,5	0,52
	110	8343761	8365412	8364900,5	0,25
	348	18079154	18207366	18207147,5	0,71
	209	11830197	11853230	11853192,0	0,19
	201	11845530	11882775	11881930,0	0,31
	46	3024320	3026096	3026096,0	0,06
	54	2774178	2813524	2813524,0	1,42
	200	15099595	15235382	15235085,0	0,90
2018	125	913035	994925	994778,6	8,95
	74	591527	662933	662911,6	12,07
	91	514667	527920	527920,0	2,58
	60	301136	302300	302295,3	0,38
	414	4005073	4143730	4143540,5	3,46
	57	273958	274098	274098,0	0,05
	211	1105132	1183805	1183756,5	7,11
	218	1554839	1661226	1654923,1	6,44
	197	1394753	1438098	1437971,8	3,10

saltos total de cada instância para os dois tipos de soluções. As colunas *Extra* contêm o número de avaliadores extras que precisaram ser providos nos casos em que o número de avaliadores da sessão é insuficiente. Por fim, as colunas *Sessões* contêm o número total de sessões necessárias para comportar o evento nas soluções manual e automática.

É possível observar uma redução expressiva tanto no número de saltos quanto no número de casos de importação de avaliadores extras nas soluções obtidas pela formulação matemática. Isso demonstra que, embora haja uma preocupação com a criação de sessões com temas semelhantes por parte das soluções manuais, o mesmo não pode ser dito quanto aos períodos de ociosidade dos avaliadores.

4.3. Tempos computacionais

O tempo de cada uma das etapas do algoritmo de resolução do PPSTCA (em segundos) para cada instância (i.e., cada linha da tabela) é mostrado na Tabela 3, em que as linhas estão dispostas na mesma ordem da Tabela 1. Pode-se perceber que as etapas de busca local e escalonamento de avaliadores são as que mais demandam tempo computacional. Embora a resolução do problema não seja instantânea, os tempos computacionais tornam-se razoáveis quando vistos como uma alternativa à criação de soluções de forma manual.

Tabela 2: Comparação entre os saltos obtidos manualmente e pelo Algoritmo 1.

Manual			Abordagem proposta		
Saltos	Extra	Sessões	Saltos	Extra	Sessões
11	1	10	2,0	0,0	9,0
12	11	20	0,0	1,0	18,5
13	31	26	1,0	2,0	25,5
8	18	23	0,0	3,5	23,0
9	14	15	1,0	1,5	15,0
7	18	14	0,0	1,0	12,0
65	34	28	0,0	1,5	24,5
59	30	32	0,0	3,5	27,0
36	12	18	0,0	0,5	16,0
36	18	15	1,0	0,0	15,0
8	6	5	0,0	2,0	5,0
17	13	10	0,0	0,0	8,0
25	14	18	0,0	0,0	14,5
54	12	24	0,0	2,0	20,5
39	19	18	0,0	0,0	15,0
39	20	13	0,0	2,0	12,5
25	17	8	0,0	0,0	5,0
19	6	9	0,0	0,0	7,0
164	13	21	0,0	8,0	21,5
49	12	11	1,0	0,5	10,5
166	44	36	0,0	3,0	31,5
112	18	22	0,0	5,5	20,5
178	18	21	0,0	2,0	20,0
37	4	5	0,0	0,0	5,0
38	1	6	0,0	2,0	6,0
92	7	21	0,0	1,5	17,5
34	15	12	0,0	2,2	12,0
19	11	8	0,9	1,3	8,0
4	11	9	1,0	2,0	9,0
14	2	6	0,2	0,2	6,0
39	13	39	0,0	6,7	36,2
32	4	6	0,0	0,0	6,0
72	7	20	0,0	2,1	20,0
59	16	21	0,0	1,9	18,9
52	15	19	0,1	2,2	19,3

5. Considerações finais

Este trabalho apresentou o Problema de Programação de Sessões Técnicas com Avaliadores. Para resolvê-lo, foi proposto um algoritmo híbrido, que combina meta-heurísticas conhecidas, como ILS e SA, e modelos matemáticos para a checagem da clusterização e o escalonamento de avaliadores. Foram apresentados, ainda, resultados comparando as soluções manuais às soluções do algoritmo, tendo como caso de estudo o evento ENIC organizado pela PROPESQ e sediado na UFPB. As comparações demonstram que o algoritmo apresentou ganhos expressivos com relação à solução manual, sendo necessário um menor número de avaliadores extras e, principalmente, reduzindo o tempo de ociosidade entre as participações de avaliadores no evento.

O sucesso dos experimentos com o ENIC sugere que o algoritmo pode apresentar bons resultados em eventos técnicos de grande porte, podendo até mesmo substituir o processo de elaboração de soluções de forma manual, que pode levar muito tempo.

Tabela 3: Tempo de cada etapa dos experimentos.

Solução Inicial	Busca Local	Perturbação	Checador	Formulação
< 0,01 (0,01%)	27,49 (72,19%)	0,10 (0,26%)	0,07 (0,19%)	10,41 (27,35%)
0,01 (< 0,01%)	321,72 (95,19%)	0,62 (0,18%)	0,12 (0,04%)	15,48 (4,58%)
0,03 (< 0,01%)	952,45 (94,69%)	1,58 (0,16%)	0,14 (0,01%)	51,63 (5,13%)
0,02 (< 0,01%)	537,21 (97,19%)	0,83 (0,15%)	0,14 (0,02%)	14,53 (2,63%)
0,01 (0,01%)	102,66 (92,59%)	0,22 (0,20%)	0,08 (0,07%)	7,90 (7,13%)
0,01 (0,01%)	50,14 (76,40%)	0,17 (0,26%)	0,07 (0,11%)	15,23 (23,21%)
0,03 (< 0,01%)	773,47 (98,20%)	1,51 (0,19%)	0,14 (0,02%)	12,53 (1,59%)
0,03 (< 0,01%)	1086,77 (97,64%)	1,55 (0,14%)	0,15 (0,01%)	24,58 (2,21%)
0,01 (0,01%)	199,21 (90,40%)	0,41 (0,19%)	0,08 (0,04%)	20,65 (9,37%)
0,01 (0,01%)	103,37 (94,93%)	0,22 (0,20%)	0,06 (0,06%)	5,24 (4,81%)
< 0,01 (0,08%)	0,88 (82,33%)	0,02 (1,58%)	0,05 (5,08%)	0,12 (10,92%)
0,01 (0,02%)	28,46 (98,95%)	0,10 (0,35%)	0,07 (0,25%)	0,12 (0,43%)
0,01 (0,01%)	161,86 (88,49%)	0,27 (0,15%)	0,12 (0,06%)	20,64 (11,29%)
0,02 (< 0,01%)	564,44 (94,36%)	0,74 (0,12%)	0,14 (0,02%)	32,82 (5,49%)
0,02 (< 0,01%)	316,63 (92,78%)	0,45 (0,13%)	0,13 (0,04%)	24,07 (7,05%)
0,01 (0,01%)	71,88 (79,60%)	0,18 (0,20%)	0,08 (0,09%)	18,16 (20,11%)
< 0,01 (0,01%)	11,51 (46,05%)	0,07 (0,29%)	0,07 (0,27%)	13,35 (53,39%)
< 0,01 (0,02%)	10,66 (69,03%)	0,06 (0,40%)	0,07 (0,42%)	4,65 (30,13%)
0,02 (< 0,01%)	410,42 (98,21%)	0,52 (0,12%)	0,13 (0,03%)	6,81 (1,63%)
0,01 (0,01%)	37,75 (86,67%)	0,11 (0,26%)	0,07 (0,16%)	5,62 (12,89%)
0,04 (< 0,01%)	2826,95 (99,26%)	2,19 (0,08%)	0,17 (0,01%)	18,65 (0,65%)
0,02 (< 0,01%)	391,06 (98,20%)	0,52 (0,13%)	0,13 (0,03%)	6,49 (1,63%)
0,02 (< 0,01%)	476,15 (70,62%)	0,62 (0,09%)	0,13 (0,02%)	197,32 (29,27%)
< 0,01 (0,05%)	3,19 (89,88%)	0,03 (0,85%)	0,06 (1,66%)	0,27 (7,56%)
< 0,01 (0,03%)	5,98 (82,63%)	0,04 (0,60%)	0,09 (1,21%)	1,12 (15,52%)
0,02 (0,01%)	351,68 (97,39%)	0,56 (0,16%)	0,13 (0,04%)	8,71 (2,41%)
0,01 (0,02%)	28,42 (84,47%)	0,08 (0,24%)	0,06 (0,18%)	5,08 (15,10%)
< 0,01 (0,02%)	10,11 (82,18%)	0,05 (0,38%)	0,05 (0,43%)	2,09 (16,99%)
< 0,01 (0,01%)	11,74 (17,98%)	0,05 (0,07%)	0,05 (0,08%)	53,44 (81,86%)
< 0,01 (0,03%)	4,95 (54,92%)	0,03 (0,35%)	0,05 (0,57%)	3,98 (44,13%)
0,04 (< 0,01%)	1707,18 (98,50%)	1,07 (0,06%)	0,14 (0,01%)	24,8 (1,43%)
< 0,01 (0,06%)	4,46 (92,23%)	0,04 (0,75%)	0,06 (1,20%)	0,28 (5,77%)
0,01 (0,01%)	205,78 (91,99%)	0,32 (0,14%)	0,10 (0,04%)	17,48 (7,82%)
0,01 (0,01%)	148,86 (78,37%)	0,21 (0,11%)	0,10 (0,05%)	40,75 (21,45%)
0,01 (0,01%)	90,18 (88,71%)	0,16 (0,16%)	0,10 (0,10%)	11,21 (11,03%)

Referências

- Correia, R. G., Subramanian, A., Bulhões, T., e Vaz Penna, P. H. (2018). O problema de programação de sessões técnicas de conferências: o caso do sbpo. In *L Simpósio Brasileiro de Pesquisa Operacional. Rio de Janeiro, Brasil. Campinas : Galoá, 2018.*
- Edis, E. e Edis, R. S. (2013). An integer programming model for the conference timetabling problem-konferans çizelgeleme problemi için bir tamsayili programlama modeli. *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, 9(2):55–62.
- Egglese, R. W. e Rand, G. K. (1987). Conference seminar timetabling. *Journal of the Operational Research Society*, 38(7):591–598.
- Kirkpatrick, S., Gelatt, C. D., e Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.
- Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, p. 320–353. Springer.
- Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100. ISSN 0305-0548.
- Potthoff, R. F. e Munger, M. C. (2003). Use of integer programming to optimize the scheduling of panels at annual meetings of the public choice society. *Public Choice*, 117(1):163–175. ISSN 1573-7101.
- Silva, P., Dias, I., Alvarenga, L., Campos, M., e Silva, A. (2013). Alocação de sessões de artigos em eventos acadêmicos: Modelo e estudo de caso. *Pesquisa Operacional para o Desenvolvimento*, 6(1):54–66.
- Stidsen, T., Pisinger, D., e Vigo, D. (2018). Scheduling euro-k conferences. *European Journal of Operational Research*, 270(3):1138–1147. ISSN 0377-2217.
- Tanaka, M., Mori, Y., e Bargiela, A. (2002). Granulation of keywords into sessions for timetabling conferences. *Proceedings of soft computing and intelligent systems (SCIS 2002)*, p. 1–5.
- Thompson, G. M. (2002). Improving conferences through session scheduling. *Cornell Hotel and Restaurant Administration Quarterly*, 43(3):71–76.
- Vangerven, B., Ficker, A. M., Goossens, D. R., Passchyn, W., Spijksma, F. C., e Woeginger, G. J. (2018). Conference scheduling—a personalized approach. *Omega*, 81:38–47.