Universidade Federal da Paraíba
Centro de Informática
Graduação em Engenharia da Computação

O Problema de Programação de Sessões Técnicas de Conferências com Avaliadores

José Felipe Nunes da Silva

João Pessoa - PB
2023

José Felipe Nunes da Silva

O Problema de Programação de Sessões Técnicas de Conferências com Avaliadores

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal da Paraíba (UFPB), como requisito para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: Anand Subramanian

João Pessoa - PB
2023

Catalogação na publicação
Seção de Catalogação e Classificação

S595p Silva, Jose Felipe Nunes da.
O problema de programação de sessões técnicas de conferências com avaliadores / Jose Felipe Nunes da Silva. - João Pessoa, 2023.
41 f. : il.

Orientação: Anand Subramanian.
TCC (Graduação) - UFPB/CI.

1. Conference scheduling. 2. Otimização. 3. Meta-heurística. I. Subramanian, Anand. II. Título.

UFPB/CI CDU 004

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

José Felipe Nunes da Silva

O Problema de Programação de Sessões Técnicas de Conferências com Avaliadores

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal da Paraíba (UFPB), como requisito para obtenção do grau de Bacharel em Engenharia da Computação.

Trabalho aprovado. João Pessoa - PB, 16 de novembro de 2023:

gov.br Documento assinado digitalmente
ANAND SUBRAMANIAN
Data: 27/11/2023 15:35:42-0300
Verifique em https://validar.iti.gov.br

Anand Subramanian
Orientador

gov.br Documento assinado digitalmente
TEOBALDO LEITE BULHOES JUNIOR
Data: 28/11/2023 15:50:09-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Teobaldo Leite Bulhões Júnior
Examinador

gov.br Documento assinado digitalmente
LUCIANO CARLOS AZEVEDO DA COSTA
Data: 27/11/2023 15:38:57-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luciano Carlos Azevedo Da Costa
Examinador

João Pessoa - PB
2023

Agradecimentos

- A Deus pela vida e por tornar tudo possível;
- Aos meus pais, por todo amor, exemplo, inspiração, apoio e dedicação;
- À minha esposa que me acompanha, apoia e inspira todos os dias;
- Aos meus irmãos que me têm como exemplo, que eu possa inspirá-los;
- A toda minha família pela torcida, apoio e afeto;
- Aos meus amigos, colegas e professores do laboratório pela orientação e contribuições na realização deste e outros trabalhos, além dos momentos de descontração e conversas inspiradoras;
- A todos os professores que fizeram parte da minha formação até aqui.

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 apresentações e avaliadores devem ser alocados a cada sessão técnica baseando-se em vários fatores, como a similaridade entre os trabalhos e a afinidade avaliador com a temática dos trabalhos. 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: Pesquisa Operacional, Otimização, Meta-heurística.

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 presentations and examiners should be allocated to each technical session based on several factors, such as the similarity between the papers and the examiners' affinity with the themes of the papers. 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: Operational Research, Optimization, Metaheuristics.

Lista de tabelas

Tabela 1 – Trabalhos relacionados ao PPSTCA. 22
Tabela 2 – Sessão com trabalhos de temática semelhante. 23
Tabela 3 – Comparação entre os benefícios totais obtidos manualmente e pelo Algoritmo 1. 36
Tabela 4 – Comparação entre o número de avaliadores extra alocados e os saltos obtidos manualmente e pelo Algoritmo 1. 37
Tabela 5 – Tempo de cada etapa do algoritmo. 38

Lista de ilustrações

Figura 1 – Sessão com participação contígua dos avaliadores. 24
Figura 2 – Sessão com participação com um avaliador ocioso entre suas participações. 25
Figura 3 – Solução sem saltos. 26
Figura 4 – Solução com um único salto. 26
Figura 5 – *Relocate* aplicado entre os *clusters* c_1 e c_2. 29
Figura 6 – *Swap* aplicado entre os *clusters* c_1 e c_2. 30

Sumário

1 INTRODUÇÃO 12
1.1 Definição do Tema 12
1.2 Justificativa 13
1.3 Objetivos 14
1.3.1 Objetivo Geral 14
1.3.2 Objetivos específicos 14
1.4 Estrutura do Trabalho 15

2 REVISÃO DA LITERATURA 16
2.1 Perspectiva dos Apresentadores 16
2.2 Perspectiva dos Participantes 18
2.3 Perspectiva Híbrida 21

3 MÉTODOS METODOLÓGICOS 23
3.1 Descrição do problema 23
3.1.1 Salto de um avaliador 24
3.2 Definição formal 25
3.3 Prova de $\mathcal{NP}$-Completude 27
3.4 Algoritmo proposto 27
3.4.1 Fase de clusterização 27
3.4.1.1 Geração de solução inicial 29
3.4.1.2 Busca local 29
3.4.1.3 Mecanismos de perturbação 30
3.4.1.4 Checagem de viabilidade 30
3.4.1.5 Alocação de avaliadores a sessões 31
3.4.2 Escalonamento de avaliadores 32

4 RESULTADOS COMPUTACIONAIS 34
4.1 Calibração dos parâmetros do algoritmo 34
4.2 Resultados da clusterização 34
4.3 Resultados do escalonamento de avaliadores 35
4.4 Tempos computacionais 36

5 CONSIDERAÇÕES FINAIS 39

REFERÊNCIAS 40

1 Introdução

1.1 Definição do Tema

Todos os anos, diversos eventos são realizados com o intuito de difundir o conhecimento e apresentar os avanços realizados por estudos científicos. Na Universidade Federal da Paraíba (UFPB), o principal evento desse escopo é o Encontro de Iniciação Científica (ENIC), organizado pela Pró-Reitoria de Pesquisa (PROPESQ). Nesse evento, os trabalhos de iniciação científica desenvolvidos durante um ano são apresentados e avaliados. Para cada apresentação, são designados três avaliadores, sendo um deles o orientador do trabalho. Os avaliadores precisam apresentar afinidade com as áreas de conhecimento ou tópicos do trabalho apresentado. Segundo funcionários da PROPESQ, a maior dificuldade na organização do evento está na escolha de avaliadores e na ordem de apresentação dos trabalhos.

No presente contexto, conferências são eventos nos quais trabalhos científicos são apresentados, tal qual o ENIC. A organização destas apresentações, respeitando as particularidades de cada evento, configura o problema de programação de sessões técnicas de conferências (PPSTC), presente na literatura há mais de 30 anos (EGLESE; RAND, 1987). No entanto, para a programação do ENIC, é necessário atribuir dois avaliadores para cada trabalho, além do orientador, introduzindo assim uma nova variante deste problema denominada de problema de programação de sessões técnicas de conferências com avaliadores (PPSTCA). Sendo assim, é correto afirmar que o PPSTCA consiste em alocar apresentações a sessões técnicas e avaliadores às apresentações, de forma que os trabalhos em uma sessão apresentem temática semelhante, que seja minimizado o tempo ocioso de cada avaliador e que não hajam sessões paralelas com um mesmo apresentador, avaliador ou orientador.

Na literatura, as abordagens para resolver o PPSTC são diversas. Há autores que apresentam métodos de resolução baseados em algoritmos exatos (EGLESE; RAND, 1987; VANGERVEN et al., 2018; STIDSEN; PISINGER; VIGO, 2018; BULHÕES; CORREIA; SUBRAMANIAN, 2022), garantindo encontrar a melhor solução possível porém limitando o tamanho de suas instâncias, uma vez que esse tipo de estratégia requer recursos computacionais elevados em certos casos. Outra estratégia comumente empregada é a utilização de heurísticas (SAMPSON; WEISS, 1995; THOMPSON, 2002; SAMPSON, 2004; VANGERVEN et al., 2018; CASTAÑO; VELASCO; CARVAJAL, 2019; CORREIA et al., 2022; RIQUELME et al., 2022; MARENCO, 2023), i.e. algoritmos que não garantem encontrar a melhor solução mas se aproximam dela o suficiente com tempo de execução razoável, tornando possível a obtenção de soluções para instâncias de maior tamanho.

Dito isso, o presente trabalho apresenta a definição do Problema de Programação de Sessões Técnicas de Conferências com Avaliadores (PPSTCA), além da proposta de um algoritmo híbrido (combinando métodos heurísticos e exatos) para resolvê-lo, tendo como caso de estudo o ENIC.

1.2 Justificativa

A programação de conferências técnicas é um problema difícil de ser resolvido devido ao seu caráter combinatório. O número de programações possíveis é tão grande que torna inviável a análise manual, resultando em um resultado insatisfatório (SILVA et al., 2013). De acordo com funcionários da PROPESQ-UFPB, a programação manual do ENIC leva pelo menos duas semanas.

É possível encontrar na literatura diversos trabalhos que abordam o PPSTC. Contudo, os autores apresentam o problema de diversas maneiras, porém sempre há semelhanças, o que permitiu que Thompson (2002) destacasse as duas principais perspectivas pelas quais os trabalhos buscavam apresentar o problema: a perspectiva dos apresentadores e a perspectiva dos participantes. Os trabalhos que se enquadram na primeira abordagem buscam atender as preferências dos apresentadores dos trabalhos, como evitar que hajam sessões simultâneas com a presença de um mesmo autor (TANAKA; MORI; BARGIELA, 2002) ou levar em conta a sua disponibilidade para participar do evento (POTTHOFF; MUNGER, 2003). Os autores de trabalhos que se enquadram na segunda abordagem tendem a atender as preferências dos participantes dos eventos em assistir certas apresentações (EGLESE; RAND, 1987). Vangerven et al. (2018) argumentam contra esta última estratégia ao afirmar que pode ser ruim depender de informações oriundas dos participantes e que participantes que optarem por um número maior de apresentações vão influenciar mais a solução, além da possibilidade de serem formadas sessões com temática incoerente. Já Bulhões, Correia e Subramanian (2022) apresentam em seu trabalho uma perspectiva que chamam de híbrida, por levar em conta tanto as preferências dos apresentadores como dos participantes, além de citarem os ganhos atribuídos aos organizadores dos eventos.

A abordagem adotada para apresentar o PPSTCA se assemelha a observada em Potthoff e Munger (2003) para apresentar o PPSTC, onde os autores consideram que os apresentadores podem assumir outros papéis na conferência, uma vez que no PPSTCA os apresentadores (orientadores) podem vir a ser avaliadores de outros trabalhos. O problema é apresentado de tal forma que pode se encaixar na perspectiva dos apresentadores (orientadores), já que um de seus objetivos é minimizar o seu tempo ocioso, i.e. o tempo entre suas participações, no decorrer do evento. Porém, há ainda o objetivo de formar sessões com temáticas coerentes, i.e. sessões nas quais os trabalhos apresentados possuem tópicos em comum, assim como fazem Tanaka, Mori e Bargiela (2002), tal objetivo é comumente apresentado por trabalhos que focam na perspectiva dos participantes. Desta

maneira, pode-se afirmar que neste trabalho é apresentada uma abordagem híbrida para o tema, diferenciando de Bulhões, Correia e Subramanian (2022) por levar em conta o papel de avaliador que os orientadores podem assumir.

Para resolver o PPSTCA é apresentada uma abordagem com objetivos hierárquicos dividida em duas partes, primeiro é proposto um algoritmo heurístico híbrido semelhante ao apresentado por Correia et al. (2018), baseado no *framework Iterated Local Search* (ILS) e *Simulated Annealing* (SA), assim como uma formulação matemática para checar a validade das soluções obtidas pela heurística, uma versão desta mesma formulação é aplicada para atribuir avaliadores a sessões, de acordo com a demanda. Nesta etapa, os trabalhos são agrupados em *clusters* (que no fim são associados às sessões do evento), baseado na semelhança entre eles. Numa segunda fase, após formadas as sessões, um modelo de programação inteira mista é utilizado para escalonar os avaliadores.

Os resultados dos experimentos envolvendo a programação do ENIC podem indicar a eficácia do método para eventos similares. Tal contribuição pode trazer benefícios para instituições públicas e privadas organizadoras de eventos deste tipo.

Sendo o método validado, os seguintes benefícios são observados:

- As sessões do evento contarão com uma temática homogênea;
- Os avaliadores terão afinidade com os trabalhos que irão examinar;
- Os avaliadores dos trabalhos passarão menos tempo ociosos durante o evento;
- A organização do evento levará menos tempo e terá qualidade assegurada.

1.3 Objetivos

1.3.1 Objetivo Geral

Apresentar a variante do Problema de Programação de Sessões Técnicas de Conferências com Avaliadores e um método híbrido capaz de resolvê-lo.

1.3.2 Objetivos específicos

- Propor uma algoritmo híbrido para resolver o PPSTCA.
- Utilizar o algoritmo desenvolvido para resolver instâncias reais do problema, originadas pela programação manual do ENIC de anos anteriores.
- Comparar os resultados do algoritmo com os resultados manuais do ENIC.

1.4 Estrutura do Trabalho

O restante deste trabalho é organizado da seguinte forma. O Capítulo 2 apresenta a revisão da literatura, o Capítulo 3 aborda a metodologia aplicada, na qual é discutida a descrição do problema e uma descrição do algoritmo proposto para solucioná-lo. No Capítulo 4 é apresentada uma discussão sobre os resultados encontrados em instâncias do ENIC. O Capítulo 5 apresenta as considerações finais.

2 Revisão da Literatura

Os trabalhos apresentados neste capítulo possuem alguma contribuição quanto ao estudo do Problema de Programação de Sessões Técnicas de Conferências. Os autores não levam em conta a alocação de avaliadores, mas trazem características importantes para a contextualização e o desenvolvimento das formulações e algoritmos apresentados neste trabalho.

Destaca-se nesta relação de trabalhos as abordagens adotadas pelos autores para apresentar o problema. Isso permite uma segregação dos trabalhos em três categorias, os que priorizam a perspectiva dos apresentadores, aqueles que levam em conta a perspectiva dos participantes e por fim os que apresentam uma abordagem híbrida, que buscam considerar as duas perspectivas citadas. Desse modos, este capítulo será dividido em três partes, de acordo com a abordagem elencada pelos autores.

Um outro ponto a ser notado na variedade de trabalhos destacados neste capítulo é a metodologia de resolução que estes adotam. Nota-se que os trabalhos que datam de épocas em que os recursos computacionais eram escaços preferem utilizar métodos heurísticos, isto é, algoritmos que constroem soluções viáveis para o problema mas que não garantem que estas sejam ótimas. Trabalhos mais recentes tendem a utilizar modelos de programação inteira mista (MPI), que são resolvidos por softwares resolvedores difundidos no mercado (e.g., CPLEX, Gurobi, CBC Coin). Algumas técnicas heurísticas avançadas também são apresentadas por este último grupo, como o emprego de heurísticas baseadas no algoritmo *Simulated Annealing* (SA) (KIRKPATRICK; GELATT; VECCHI, 1983), *Designer* Combinatorial (DC) (ANDERSON, 1997), *Iterated Local Search* (ILS) (LOURENÇO; MARTIN; STÜTZLE, 2003), *Greedy randomized adaptive search procedure* (GRASP) (FEO; RESENDE, 1995), Geração de Colunas (GC) (GILMORE; GOMORY, 1961), além dos métodos *Branch-and-Cut* (BC) (PADBERG; RINALDI, 1987) e *Branch-and-Cut-and-Price* (BCP) (FUKASAWA et al., 2006). Também são empregados métodos geralmente utilizados em estudos que envolvem o aprendizado de máquinas, como *Self-organizing map* (SOM) (TERMINI, 1995) e Hiper-Heurísticas (BURKE et al., 2010).

2.1 Perspectiva dos Apresentadores

Tanaka, Mori e Bargiela (2002) propõem um método que evita a ocorrência de sessões distintas num mesmo horário com a presença de um mesmo apresentador, e por isso se enquadra nesta categoria. Neste trabalho, o método proposto também visa propor soluções de modo que as sessões tenham temáticas homogêneas, preferindo alocar a uma mesma sessão trabalhos que possuam palavras-chave em comum. O método utilizado

para resolução do problema foi o *Self Organizing Map*, um paradigma de rede neural de aprendizado não supervisionado, utilizando dados de uma conferência com 313 trrabalhos e 86 palavras-chave, que foram manualmente categorizadas.

O método proposto por Potthoff e Munger (2003) consideram que os apresentadores podem receber outras atribuições durante a conferência, como *chair* de alguma sessão. Dessa forma, deve-se garantir que os apresentadores não sejam alocados em sessões paralelas, seja como autores, co-autores, *chair*, entre outros papéis que possa assumir no evento. Além disso, têm por objetivo realizar a distribuição dos trabalhos em sessões, e das sessões em faixas de horários, visando balancear os temas ao decorrer da realização da conferência. O problema foi resolvido utilizando um modelo de programação inteira. Seus resultados foram comparados à programação do evento já realizado, a edição de 2001 do *Annual Meetings of the Public Choice Society*. Em Potthoff e Brams (2007), os autores utilizaram o mesmo método para resolver as edições de 2005 e 2006 do evento citado, dessa vez levando em conta a disponibilidade dos apresentadores.

O objetivo de Nicholls (2007) é alocar trabalhos a sessões de modo que atenda a preferência dos apresentadores, ao permitir que estes selecionem dias e horários que desejam participar da conferência, contudo, leva em conta também algumas preferências dos participantes, que são atendidas ao tentar evitar que hajam sessões paralelas de um mesmo tema, permitindo que aqueles que desejarem possam assistir às apresentações de seu interesse ao longo do evento. Para a resolução do problema, foi proposto um método heurístico que, respeitando as restrições, atende o máximo possível das preferências de dias e horários dos apresentadores, ao alocar sessões a faixas de horários. Não há função objetivo explícita, a heurística deve apenas respeitar as restrições definidas. O método foi utilizado na programação de uma edição da conferência anual do *Western Decision Sciences Institute*, contando com 80 sessões de 200.

Em Ibrahim, Ramli e Hassan (2008) é proposto um *designer* combinatorial para garantir que a programação do evento seja realizada balanceando os temas das sessões ao longo do tempo. Esse método consiste em selecionar subconjuntos de um conjunto finito de modo que as restrições sejam respeitadas. É apresentada uma versão do problema em que cada sessão é formada por trabalhos de 3 áreas, são consideradas 3 sessões paralelas e a programação deve ser distribuída num período de k dias.

A abordagem adotada por Edis e Edis (2013) visa resolver o problema utilizando um modelo de programação inteira mista genérico com dois objetivos. Primeiro cada sessão recebe um tópico e é preenchida por trabalhos deste tópico, minimizando o número de sessões paralelas de um mesmo tópico. Depois é minimizada a diferença de tamanho entre sessões paralelas. A programação é feita de forma que não hajam sessões paralelas com um mesmo apresentador. Os experimentos foram realizados utilizando uma instância hipotética.

Em Quesnelle e Steffy (2015) é proposto um modelo de programação inteira que garante que não haja sessões paralelas com apresentações de um mesmo autor. Além disso, o modelo busca minimizar o conflito de preferência dos participantes do evento e garantir que as sessões sejam alocadas a salas que atendam a requisitos predefinidos. Os temas dos trabalhos não são levados em consideração. Foi notada uma melhora na performance do método ao dualizar algumas restrições.

Eltayeb e Ahmed (2021) apresenta o problema de forma que sua resolução está sujeita a restrições fortes e fracas, restrições fortes não podem ser violadas, e o número de violações das restrições fracas deve ser minimizado. As restrições fracas incluem as preferências doas apresentadores, como, por exemplo, apresentações de um mesmo apresentador devem ser alocadas a uma mesma sala. O método consiste em gerar uma solução inicial factível e, partindo dessa, aplicar uma seleção de hiper-heurísticas. Foram utilizadas duas instâncias baseadas em eventos reais para testar a eficácia do método, uma considerada pequena e outra considerada grande. Os resultados do emprego das hiper-heurísticas foram comparados e organizados em *rankings* pelos autores.

No trabalho de Riquelme et al. (2022) é proposta uma nova variante do problema, onde é introduzido o conceito de *tracks*, que os autores definem como sendo sub-eventos de uma conferência e agrupam apresentações de uma temática cada. A programação do evento é realizada evitando a aparição de um mesmo autor em sessões paralelas. Para resolver o problema eles propõem um modelo de programação linear inteira, bem como uma meta-heurística baseada no método *Simulated Annealing*. Propõem ainda um gerador de instâncias para fins de *benchmark*.

2.2 Perspectiva dos Participantes

Eglese e Rand (1987) apresentam uma abordagem que tem como objetivo maximizar o número de apresentações assistidas pelos participantes do evento. Fazem isso levando em conta o interesse dos participantes, que fornecem uma lista de apresentações indicando a ordem de prioridade. Dessa forma, a função objetivo minimiza o custo das violações de preferências dos participantes. As apresentações são alocadas a faixas de horários e a salas, os tópicos abordados não são levados em conta. Para solucionar o problema, foi proposto um modelo de programação linear inteira mista, porém não foi possível resolvê-lo de maneira exata devido às limitações computacionais da época, sendo assim os autores decidiram desenvolver uma heurística capaz de gerar boas soluções para o problema. O método consiste em encontrar uma primeira solução factível e melhorá-la utilizando um algoritmo de *annealing*. O método proposto fora utilizado para programar a conferência anual da organização *Tear Fund*, contendo 15 apresentações para serem distribuídas, podendo haver repetição de apresentações.

Sampson e Weiss (1995) apresentam uma versão estendida da apresentada por Eglese

e Rand (1987) e distribuem não só os trabalhos a sessões, mas também os participantes do evento, levando em consideração a capacidade das salas associadas às sessões, de forma que estes estejam devidamente acomodados durante as apresentações. Os participantes demonstram seu interesse em certas apresentações e isso é levado em conta na resolução do problema. O método para resolver o problema consiste de duas partes principais: considerando a capacidade das salas, distribuir as apresentações de forma viável; e alocar os participantes às sessões. Uma heurística é proposta para resolver as duas partes do problema de forma simultânea. As instâncias utilizadas para testar a eficácia do método foram geradas de maneira aleatória.

Thompson (2002) compara as abordagens de resolução do problema de programação de conferências, e assume como objetivo mostrar que a satisfação dos participantes está diretamente ligada com a realização da programação que leva em conta as preferências destes. Foram colhidas as preferências dos participantes, que listaram até 8 apresentações que queriam assistir, sem ordem de prioridade. Também é levada em conta a capacidade das salas e sua disponibilidade. O método para resolução combinou um algoritmo construtivo e a meta-heurística *Simulated Annealing*. Em seus experimentos, notou-se que ainda que para pequenas conferências, com 100 participantes, a programação manual não apresenta bons resultados se comparados com a programação obtida por seu método heurístico.

Um embasamento teórico é apresentado por Sampson (2004) para explicar a abordagem da programação de conferências baseada em preferências. São ressaltadas as semelhanças do problema com outros da literatura. É proposta uma formulação matemática que tem como função objetivo maximizar a satisfação dos participantes das conferências. A abordagem é similar a vista em Eglese e Rand (1987), porém não permite a repetição de apresentações e levam em consideração a capacidade das salas de das sessões. O método de resolução foi uma abordagem baseada na heurística *Simulated Annealing* e os experimentos foram realizados em uma instância de 213 sessões e 10 períodos de tempo.

O conceito de *session hopping* é introduzido por Vangerven et al. (2018) que define como o ato de um participante ter que "saltar" de uma sessão para outra, isto é, trocar de sala enquanto uma sessão está em andamento para que possa assistir a apresentações de seu interesse. O objetivo da abordagem proposta pelos autores é minimizar os saltos entre sessões. Esse tipo de abordagem é possibilitada pelo fornecimento prévio das preferências de apresentações por parte dos participantes, o resultado pode conter sessões com temática mista, mas, segundo os autores, evita incômodos ao diminuir o fluxo de pessoas saindo e entrando nas salas durante as apresentações. Para resolver o problema um método hierárquico dividido em três partes é proposto. Primeiro é resolvido um modelo de programação inteira com o objetivo de maximizar as preferências dos participantes. Na segunda fase, a depender da natureza da instância, é utilizado um modelo de programação inteira ou um método heurístico para minimizar os saltos dos participantes. Já na terceira fase o número de violações de disponibilidade dos participantes é minimizado, com o

emprego de um modelo de programação inteira. O método é utilizado para programar diversas conferências reais.

Em Stidsen, Pisinger e Vigo (2018) é apresentado um modelo de programação linear inteira mista multi-objetivo para programar conferência EURO-k, a maior conferência de Pesquisa Operacional da Europa e uma das maiores do mundo. Os objetivos são hierárquicos e a sua resolução apresenta soluções que são flexíveis o suficiente para que os organizadores possam realizar alterações caso necessário. Como as apresentações são realizadas em prédios diferentes, o primeiro objetivo é minimizar o número de áreas que estão presentes em mais de um prédio. O segundo objetivo é maximizar o úmero de áreas correlacionadas em um mesmo prédio. O terceiro objetivo minimiza o número de salas alocadas para cada trilha (i.e., grupo de sessões consecutivas com temática coerente). O quarto objetivo minimiza o tempo ocioso em cada trilha, isso é, o tempo entre uma apresentação e outra. Por fim, o quinto objetivo maximiza a probabilidade de ter capacidade suficiente nas salas para os participantes. A execução dos métodos se deu em algumas horas e os organizadores do evento puderam realizar alterações e gerar novos resultados lhes poupando tempo e aumentando a satisfação dos participantes.

Castaño, Velasco e Carvajal (2019) apresentam uma abordagem que busca atender aos interesses dos participantes ao evitar que trilhas de um mesmo tema ocorram em paralelo. Seu método é divido em duas fases, a primeira consiste agrupar os trabalhos em sessões maximizando a similaridade entre os trabalhos de cada sessão, e a segunda fase consiste em minimizar a similaridade entre sessões paralelas. Nesta abordagem o problema foi resolvido utilizando tanto modelagem de programação inteira mista como também abordagens de algoritmos avançados como uma heurística de geração de colunas e o emprego de um algoritmo GRASP caso não sejam encontradas soluções satisfatórias. O método foi capaz de encontrar soluções próximas do ótimo em tempos de execução relativamente baixos.

Marenco (2023) traz uma abordagem semelhante às citadas quando evita que sessões com temáticas que possam interessar a uma mesma audiência sejam programadas simultaneamente. Foi proposta uma formulação para o problema baseado na edição de 2022 da *Latin-Iberoamerican Conference on Operations Research*. Contando com duas funções objetivas, uma para maximizar a coerência temática das sessões e outra para minimizar a quantidade de sessões paralelas coma mesma temática. Para resolver o problema, foi implementada um algoritmo híbrido dividido em duas fases: uma heurística *GRASP*, para agrupar os trabalhos em sessões; e na segunda parte um modelo de programação linear inteira foi resolvido para alocar as sessões em faixas de horários. A programação gerada pelo método foi utilizada na conferência, que contou com um público de cerca de 1000 participantes e 341 apresentações, distribuídas em 8 períodos de tempo compostos por 12 sessões paralelas e cada sessão podendo comportar até 4 apresentações.

2.3 Perspectiva Híbrida

Bulhões, Correia e Subramanian (2022) apresentam uma modelagem genérica para o problema. Os autores consideram as principais semelhanças entre os trabalhos da literatura para a construção de seu modelo. A abordagem adotada pelos autores é considerada híbrida por considerar sessões com temáticas coerentes e restringindo a quantidade de trabalhos com mesma temática em sessões simultâneas, aumentando assim a satisfação dos participantes, e por considerar que trabalhos que possuam um mesmo apresentador não podem estar em sessões paralelas, existe ainda um benefício para os organizadores da conferência. Foram apresentadas 3 versões do modelo matemático proposto. Primeiro, um modelo compacto, resolvido por um resolvedor de programação linear inteira mista, que foi capaz de resolver instâncias consideradas pequenas pelos autores. Segundo, uma versão do modelo resolvida com uma implementação do algoritmo *Branch-and-cut* proposta pelos autores, capaz de resolver instâncias de tamanho moderado do problema. Por fim, uma terceira versão contando adaptações para ser resolvida pelo algoritmo *Branch-and-cut-and-price*, que obteve soluções suficientemente boas para instâncias maiores do problema de até 163 trabalhos. O método apresentado pelos autores foi empregado em instâncias de dois eventos reais, como também em instâncias artificiais.

Correia et al. (2022) apresentam um algoritmo da classe *matheuristic* para resolver o problema seguindo a modelagem proposta por Bulhões, Correia e Subramanian (2022). Os autores apresentam um algoritmo (denominado HILS) que combina os princípios dos métodos *iterated local search* e *simulated annealing*, além do emprego de dois procedimentos baseados em modelos matemáticos, o primeiro utiliza os conjuntos de restrições para validar a solução obtida na execução do HILS, e o segundo trata-se de um modelo do problema de partição de conjuntos, que busca encontrar a combinação ótima de *clusters*, que no fim são associados às sessões do evento. O método é aplicado e bem sucedido em instâncias reais baseadas no evento *Simposio Brasileiro de Pesquisa Operacional*, contendo até 324 trabalhos.

A Tabela 1 destaca os métodos utilizados por cada trabalho, os classificando de acordo com a perspectiva adotada por seus autores.

Tabela 1 – Trabalhos relacionados ao PPSTCA.

Perspectiva dos Apresentadores		
Autor(es)	**Ano**	**Método de resolução**
Tanaka, Mori e Bargiela	2002	SOM
Potthoff e Munger	2003	MPI
Nicholls	2007	Heurística
Ibrahim, Ramli e Hassan	2008	DC
Edis e Edis	2013	MPI
Quesnelle e Steffy	2015	MPI
Eltayeb e Ahmed	2021	Hiper-Heurísticas
Riquelme et al.	2022	MPI + SA
Perspectiva dos Participantes		
Autor(es)	**Ano**	**Método de resolução**
Eglese e Rand	1987	MPI, Heurística
Sampson e Weiss	1995	Heurística
Thompson	2002	SA
Sampson	2004	SA
Vangerven et al.	2018	MPI, Heurística
Stidsen, Pisinger e Vigo	2018	MPI
Castaño, Velasco e Carvajal	2019	GRASP + GC
Marenco	2023	GRASP + MPI
Perspectiva Híbrida		
Autor(es)	**Ano**	**Método de resolução**
Bulhões, Correia e Subramanian	2022	MPI, BC, BCP
Correia et al.	2022	ILS + SA + MPI

3 Métodos Metodológicos

3.1 Descrição do problema

Ao programar uma conferência científica é comum levar em conta a temática dos trabalhos para construir sessões com apresentações que possam interessar um certo público. Isso pode ser feito contabilizando o número de palavras-chave que os trabalhos compartilham (TANAKA; MORI; BARGIELA, 2002), por exemplo. No ENIC, isso é feito através dos temas que os autores atribuem a seus trabalhos. A Tabela 2 representa uma sessão na qual os trabalhos possuem ao menos um tema em comum, além de haverem trabalhos que compartilham o mesmo orientador. É desejável que sessões como esta façam parte da programação do ENIC.

Tabela 2 – Sessão com trabalhos de temática semelhante.

Autor	**Orientador**	**Tema 1**	**Tema 2**	**Tema 3**
Autor 1	Orient. 1	*t1*	*t2*	*t4*
Autor 2	Orient. 2	*t1*	*t3*	*t4*
Autor 3	Orient. 2	*t1*	*t3*	*t5*
Autor 4	Orient. 3	*t1*	*t2*	*t5*

Ainda no contexto do ENIC deve se levar em conta que os trabalhos possuem orientadores, que precisam estar presentes nas suas apresentações. Estes orientadores podem vir a ser avaliadores de outros trabalhos, seja na mesma sessão que estão os trabalhos sob sua orientação ou não. Além do fato de que cada apresentação deve ter obrigatoriamente 3 avaliadores, sendo um deles o orientador do trabalho apresentado.

Em eventos deste tipo é comum que hajam sessões ocorrendo numa mesma faixa de horário, isso é, ocorrem paralelamente. Os apresentadores dos trabalhos não podem ser alocados a sessões paralelas. No ENIC não é diferente, na sua programação são evitados choques de horários de apresentadores, avaliadores e orientadores.

Dito isso, programar uma conferência como o ENIC é uma tarefa que configura o PPSTCA. Este problema consiste em, dados os dias e horários das sessões do evento e os trabalhos a serem apresentados, alocar trabalhos a sessões considerando a temática e designar avaliadores disponíveis para atuar em sessões que não possuam o número suficiente de orientadores para participar das avaliações. Por fim, é necessário escalonar os avaliadores das apresentações de cada sessão, de forma a minimizar o tempo ocioso entre as suas participações. Este último objetivo é atingido minimizando o número de saltos realizados pelos avaliadores ao longo de cada sessão. A Seção 3.1.1 apresenta em detalhes o conceito de salto utilizado neste trabalho.

Uma definição formal para o PPSTCA é abordada na Seção 3.2 e o método proposto para resolvê-lo é apresentado na Seção 3.4.

3.1.1 Salto de um avaliador

A Figura 1 apresenta um exemplo de sessão em que 11 apresentações de 20 minutos são avaliadas pelos avaliadores a1, a2, a3, a4 e a5. Os retângulos verdes indicam que o avaliador é orientador do trabalho apresentado. Nota-se que nesta sessão a participação dos avaliadores é contígua, isto é, não há ociosidade entre as avaliações de cada um deles.

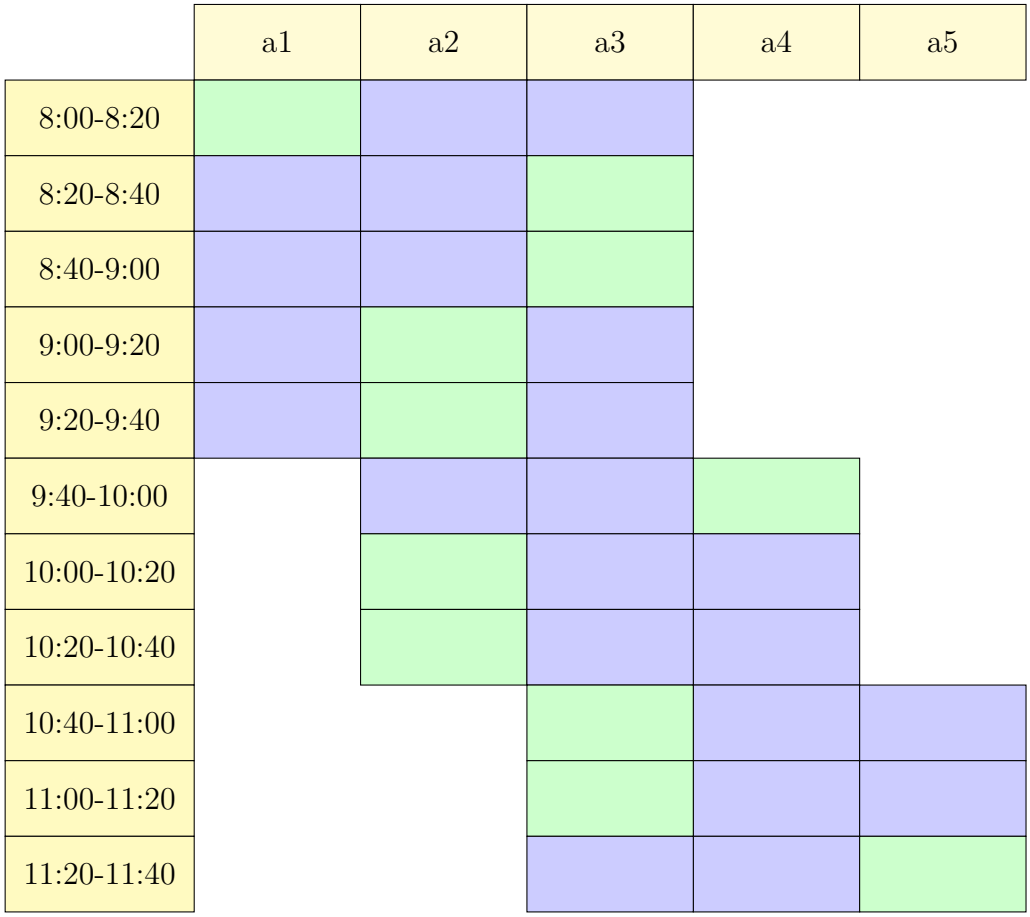

Figura 1 – Sessão com participação contígua dos avaliadores.

Em contraponto, a Figura 2 apresenta uma organização em que o avaliador a3 participa das 8:00 as 10:40, não avalia no horário das 10:40 e volta a avaliar às 11:00. Uma vez que cada apresentação tem duração de 20 minutos, o tempo que o avaliador passou ocioso entre as suas participações, diz-se que o avaliador a3 realizou um salto de tamanho 1.

O tempo designado para ocorrer uma apresentação é denominado *slot*. O tamanho dos saltos realizados pelos avaliadores é equivalente à quantidade de *slots* consecutivos em que ele ficou ocioso entre as suas participações.

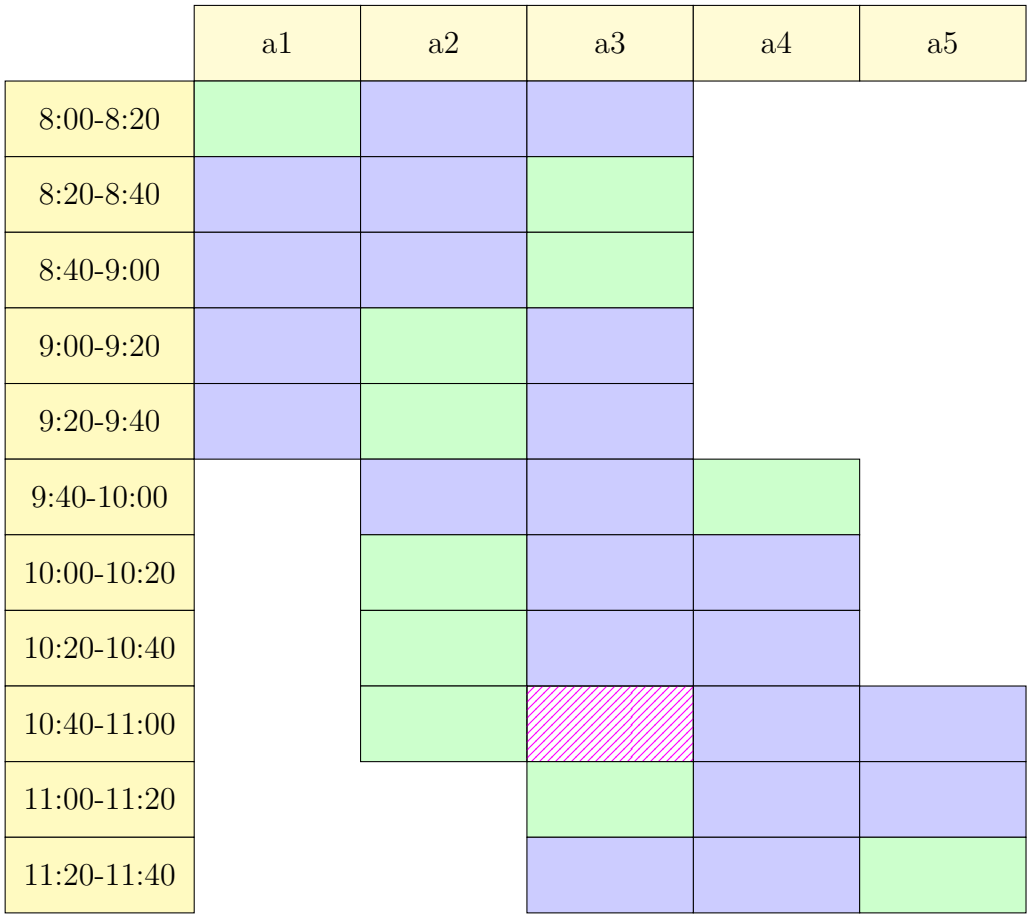

Figura 2 – Sessão com participação com um avaliador ocioso entre suas participações.

3.2 Definição formal

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 $d \in D$, $A = \{1, \ldots, m\}$ um conjunto de avaliadores e K um conjunto de sessões. Cada sessão $k \in K$ — que ocorre em um horario 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 n o número de avaliadores que deve estar presente em cada *slot*, 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 3 apresenta um exemplo de uma sessão com apresentações sem saltos, pois as participações

dos avaliadores a_i estão contíguas, enquanto a Figura 4 apresenta uma sessão com um salto de tamanho 1 realizado pelo avaliador a_1 no *slot* 3, já que ele avalia nos *slots* 1 e 2, e só volta a fazê-lo no *slot* 4.

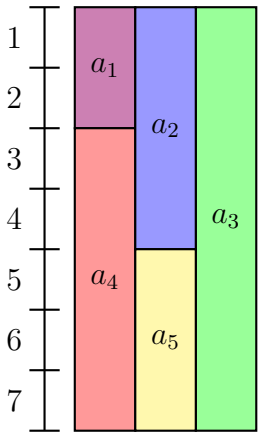

Figura 3 – Solução sem saltos.

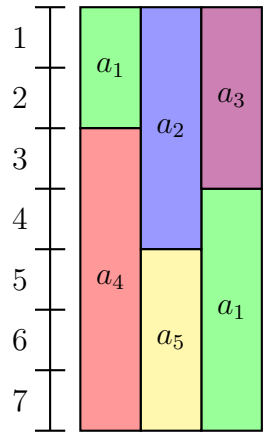

Figura 4 – Solução com um único salto.

Assim, o PPSTCA consiste em alocar trabalhos a sessões e avaliadores a *slots*, e é constituído de duas fases:

- Atribuir trabalhos (e consequentemente seus orientadores) e possíveis avaliadores (quando necessário) a sessões de forma a maximizar o benefício total;
- Minimizar, para cada sessão $k \in K$, a ocorrência de "saltos" realizados pelos avaliadores presentes na sessão $k \in K$.

A primeira fase está sujeita às seguintes restrições.

- Sessões paralelas não podem ter avaliadores em comum;
- Cada sessão $k \in K$ deve conter até $|S_k|$ trabalhos;
- Cada avaliador i só pode ser alocado a no máximo L_i sessões das quais não existem trabalhos sob a sua orientação.

As fases possuem objetivos hierárquicos e ao obter o resultado da primeira fase cada sessão $k \in K$ deve ser associada a um conjunto de trabalhos T_k, e a um conjunto de avaliadores A_k, cujo os membros podem ou não possuir trabalhos que são orientadores na sessão $k \in K$. Dessa forma, a segunda fase está sujeita às seguintes restrições.

- Cada trabalho deve ser alocado a um único *slot*;
- Exatamente n avaliadores devem estar presentes durante a apresentação de cada trabalho, sendo um deles sempre o orientador do trabalho;
- Cada avaliador $i \in A_k$ deve avaliar entre c_i^k e d_i^k trabalhos (incluindo aqueles dos quais é orientador).

3.3 Prova de $\mathcal{NP}$-Completude

O problema de determinar 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 $\mathcal{NP}$-Complete, pois o Problema da Partição (PP) pode ser reduzido a ele. Considere um multiconjunto de inteiros $U = \{v_1, \ldots, v_n\}$. 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.

3.4 Algoritmo proposto

Um algoritmo híbrido foi utilizado para resolver o PPSTCA. Na primeira fase (denominada 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) e possíveis avaliadores a *clusters* de forma a maximizar o benefício, por fim os *clusters* são associados a sessões (conforme a Seção 3.4.1). 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. Para distribuir os avaliadores, uma formulação matemática é utilizada em cada sessão (conforme a Seção 3.4.2).

3.4.1 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; MARTIN; STÜTZLE, 2003). A etapa de buscas locais é baseada em *Variable Neighborhood Descedent* (VND) (MLADENOVIĆ; HANSEN, 1997). Além disso, uma abordagem inspirada em *Simulated Annealing* (SA) (KIRKPATRICK; GELATT; VECCHI, 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 α, sendo $0 > \alpha > 1$ (linha 24), tornando cada vez menores as

Algoritmo 1 *HILS*

```
1: procedimento HILS(I_Max, I_ILS, T_0, α)
2:     f* ← 0
3:     para i := 1 até I_Max faça
4:         s ← GeraSoluçãoInicial()
5:         s' ← s
6:         s'' ← s
7:         iter_ILS ← 0
8:         T ← T_0
9:         enquanto iter_ILS ≤ I_ILS faça
10:            s ← BuscaLocal(s)
11:            Δ ← f(s) − f(s')
12:            se Δ > 0 então
13:                s' ← s
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:                senão
21:                    s' ← s''
22:            s ← Perturba(s')
23:            iter_ILS ← iter_ILS + 1
24:            T ← T × α
25:        se f(s') > f* e VerificaViabilidade(s') = true então
26:            s* ← s'
27:            f* ← f(s')
28:    s* ← AlocaAvaliadoresAClusters(s')
29:    retorne s*
30: fim HILS
```

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 (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. Ao fim das I_{Max} iterações, avaliadores são alocados aos *clusters* da melhor solução, conforme necessidade (linha 28). Retorna-se a melhor solução viável encontrada na linha 29. 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 Seção 3.4.2.

3.4.1.1 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.

3.4.1.2 Busca local

A busca local aplicada é o procedimento RVND (SUBRAMANIAN et al., 2010), no qual a ordem das vizinhanças é aleatória. Durante a exploração de cada uma das estruturas de vizinhança descritas a seguir, o melhor vizinho (aquele que leva ao maior custo) é escolhido. Quando uma vizinhança não leva a uma melhora na solução, a próxima é escolhida, até que não reste mais vizinhanças a serem exploradas.

- *Relocate* — $N^{(1)}$: Um *cluster* formado por um ou mais trabalhos é escolhido e um 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.

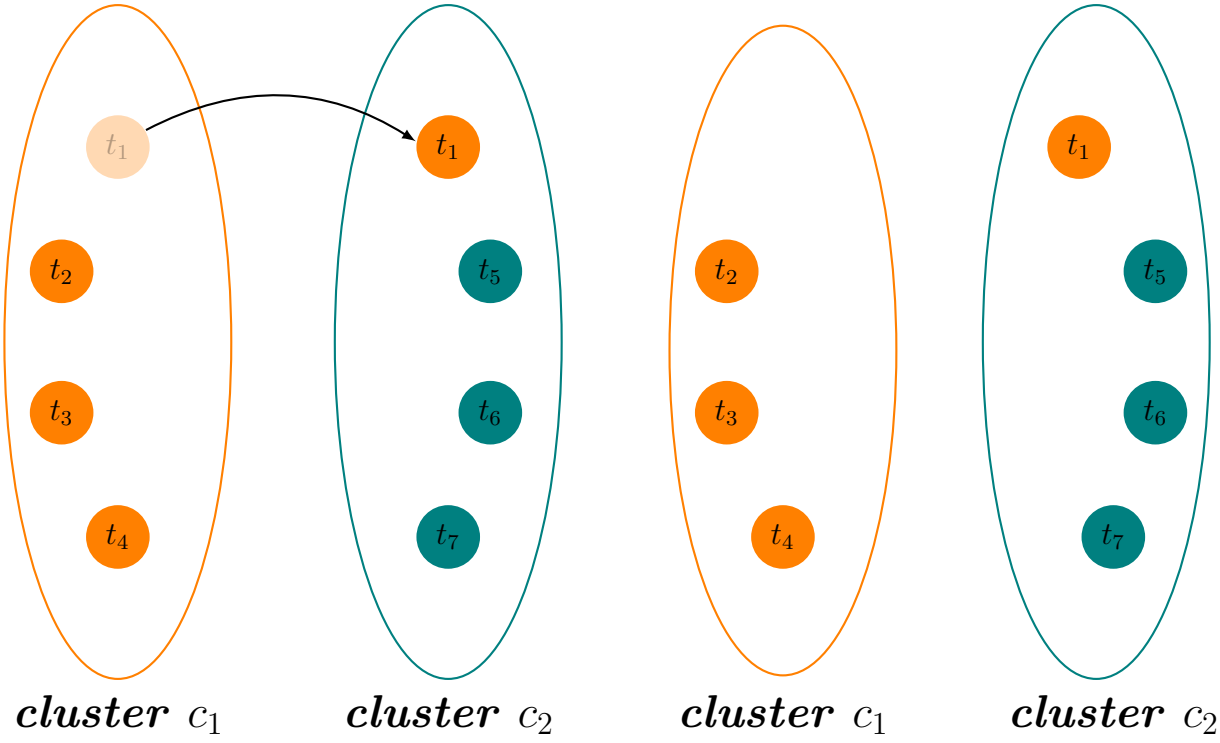

Figura 5 – *Relocate* aplicado entre os *clusters* c_1 e c_2.

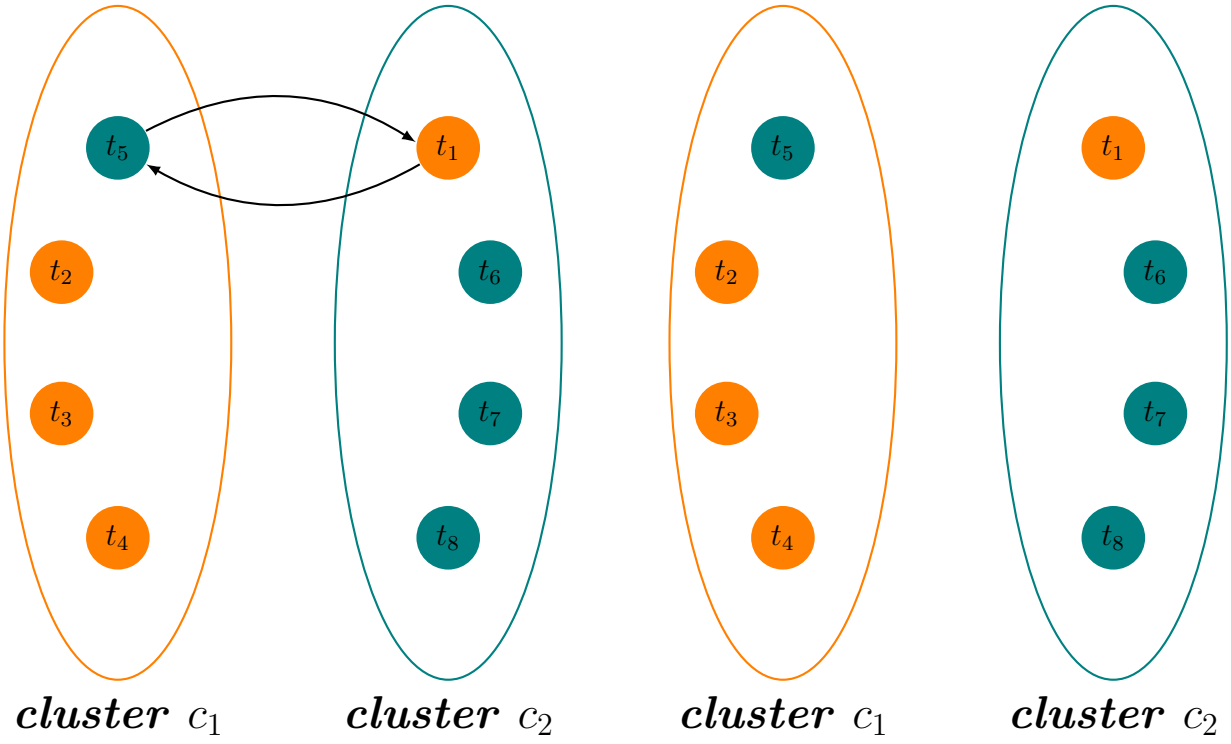

Figura 6 – *Swap* aplicado entre os *clusters* c_1 e c_2.

Na Figura 5 o trabalho t_1 é realocado do *cluster* c_1 para o *cluster* c_2, que antes possuía apenas os trabalhos t_5, t_6 e t_7. Já na Figura 6, há uma troca de trabalhos entre os *clusters* c_1 e c_2, na qual os trabalhos t_1 e t_5 são realocados de seus *clusters* de origem para um outro.

3.4.1.3 Mecanismos de perturbação

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

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

3.4.1.4 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 $i \in A$ e C_i' o conjunto de *clusters* cujo o avaliador i não é orientador, m_c é a demanda do *cluster* $c \in C$ por avaliadores, e M_i o parâmetro que determina quantas avaliações o avaliador $i \in A$ pode realizar ao ser alocado a um cluster. Finalmente, a variável binária ξ_k^c assume o valor 1 se o *cluster* $c \in C$ foi alocado na sessão $k \in K$ e a variável binária $\lambda_{i,k}^c$ que assume valor 1 quando um avaliador i é alocado para avaliar trabalhos numa sessão k associada ao *cluster* c e 0 caso contrário. O modelo de programação linear inteira para checar a validade da solução pode ser escrito da seguinte forma:

$$\text{Min } 0 \tag{3.1}$$

sujeito a

$$\sum_{k \in K_c} \xi_k^c = 1 \qquad c \in C \tag{3.2}$$

$$\sum_{c \in C_i} \sum_{k \in K_c} \xi_k^c + \sum_{c \in C_i'} \sum_{k \in K_c} \lambda_{i,k}^c \leq 1 \qquad i \in A, d \in D, h \in H_d \tag{3.3}$$

$$\sum_{i \in A_k} M_i \lambda_{i,k}^c = m_c \xi_k^c \qquad k \in K_c \tag{3.4}$$

$$\sum_{c \in C_i'} \sum_{k \in K_c} \lambda_{i,k}^c \leq L_i \qquad i \in A \tag{3.5}$$

$$\sum_{c \in C : k \in K_c} \xi_k^c \leq 1 \qquad k \in K \tag{3.6}$$

$$\xi_k^c, \lambda_{i,k}^c \in \{0, 1\} \qquad c \in C, k \in K_c, i \in A. \tag{3.7}$$

A finalidade do modelo é apenas verificar se a solução encontrada respeita as restrições apresentadas, não sendo necessária uma função objetivo. As restrições (3.2) impõem que cada *cluster* deve ser alocado a exatamente uma sessão. As restrições (3.3) garantem que não haja trabalhos de um mesmo avaliador em sessões paralelas e que um orientador não é alocado a sessões paralelas. As restrições (3.4) garantem que o número de avaliadores associados a um *custer* atenda a demanda daquele *cluster*. As restrições (3.5) garantem que o limite de sessões que um orientador pode ser alocado como avaliador não seja excedido. As restrições (3.6) implicam que no máximo um *cluster* possa ser alocado a uma sessão. As restrições (3.7) indicam a natureza das variáveis de decisão.

3.4.1.5 Alocação de avaliadores a sessões

A alocação de avaliadores para suprir a demanda das *clusters* foi realizada utilizando uma variação da formulação matemática `F1`, denominada `F2`, que possui todas as restrições

da primeira porém com a aplicação de uma função objetivo como segue.

$$\text{Min} \sum_{c \in C} \sum_{k \in K} \sum_{i \in A} \lambda_{i,k}^{c} \tag{3.8}$$

A função objetivo (3.8) miniza o número de alocações de avaliadores a *clusters*.

3.4.2 Escalonamento de avaliadores

Para cada uma das sessões $k \in K$, o escalonamento de avaliadores é feito utilizando-se a formulação matemática `F3`, 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. O modelo de programação linear inteira para escalonar os avaliadores pode ser escrito da seguinte forma:

$$\text{Min } h_{max} \tag{3.9}$$

sujeito a

$$\sum_{i \in A_k} x_{is} = n \qquad s \in S_k \tag{3.10}$$

$$\max\{c_i^k, |T_k \cap T_i|\} \leq \sum_{s \in S_k} x_{is} \leq d_i^k \qquad i \in A_k \tag{3.11}$$

$$h_{is} \geq h_{is-1} + (1/|S_k|) \sum_{s'=0}^{s-1} x_{is'} - |S_k| x_{is} + l_{is} - 1 \qquad i \in A_k, s \in S_k \tag{3.12}$$

$$h_{max} \geq h_{is} \qquad i \in A_k, s \in S_k \tag{3.13}$$

$$h_{i0} = 0 \qquad i \in A_k \tag{3.14}$$

$$l_{is} \geq x_{is'} \qquad i \in A_k, s \in S_k, s' \geq s+1 \tag{3.15}$$

$$y_{is} \leq x_{is} \qquad i \in A_k, s \in S_k \tag{3.16}$$

$$\sum_{s \in S_K} y_{is} \geq |T_i \cap T_k| \qquad i \in A_k \tag{3.17}$$

$$\sum_{i \in A_k} y_{is} = 1 \qquad s \in S_k \tag{3.18}$$

$$x_{is}, y_{is}, l_{is} \in \{0, 1\} \qquad i \in A_k, s \in S_k \tag{3.19}$$

$$h_{is} \in \mathbb{Z} \qquad i \in A_k, s \in S_k \cup \{0\} \tag{3.20}$$

$$h_{max} \geq 0 \tag{3.21}$$

A função objetivo (1) busca minimizar o tamanho do maior salto na solução. As restrições (3.10) garantem que cada *slot* da sessão $k \in K$ possui exatamente n avaliadores.

As restrições (3.11) limitam o número de trabalhos que cada avaliador $i \in A_k$ pode avaliar. As restrições (3.12)–(3.15) são responsáveis por contabilizar os saltos através das variáveis h_{is}, além de garantir que h_{max} é o maior salto. As restrições (3.16)–(3.18) garantem que cada *slot* da sessão $k \in K$ possui um único orientador, e que ele deve avaliar o próprio trabalho. As restrições (3.19)–(3.21) dizem respeito ao domínio das variáveis.

Após a aplicação da formulação `F4`, 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 a seguir.

$$\text{Min} \sum_{i \in A_k} \sum_{s \in S_k} h_{is} \tag{3.22}$$

A função objetivo (3.22) busca obter o menor número de saltos possível. Por fim, além de possuir as mesmas restrições que `F3`, a formulação `F4` também está sujeita às restrições a seguir.

$$h_{is} \leq h_{max} \qquad i \in A_k, s \in S_k \tag{3.23}$$

As restrições (3.23) 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 utilizando a linguagem de programação C++, compilador g++ (versão 11.4.0) 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 AMD Ryzen 5 5600G, 3.9 Ghz, 16 GB de memória RAM, 3200 Mhz, e sistema operacional Ubuntu 22.04 LTS.

A Seção 4.1 destaca os valores assumidos pelos parâmetros do algoritmo para obter as soluções discutidas neste trabalho, as Seções 4.2 e 4.3 apresentam em detalhes uma comparação entre as soluções manuais e aquelas obtidas utilizando o algoritmo proposto. Já a Seção 4.4 destaca o tempo que cada parte da resolução do problema levou para ser executada. A discussão abrange soluções contendo dados dos anos de 2014 a 2018, porque a partir de 2019 o ENIC passou a ser programado por uma versão preliminar do método proposto neste trabalho, que apresentado em Silva et al. (2020).

4.1 Calibração dos parâmetros do algoritmo

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 outro lado, se $t \in T$ e $w \in T$ não possuem temas em comum mas possuem o mesmo orientador, utiliza-se $b_{tw} = 10000$. Além disso, caso uma sessão apresente quórum insuficiente, é assumida demanda (m_c) equivalente ao número de trabalhos da sessão multiplicado pelo número de avaliadores faltantes. A capacidade de avaliação (M_i) por sessão de cada avaliador é sempre o tamanho da sessão.

Durante a fase de escalonamento de avaliadores, no caso particular do ENIC, utilizou-se $n = 3$, $c_i^k = 0$ e $d_i^k = |S_k|$.

4.2 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 3 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* contém 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(%)* contém 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.

Durante a clusterização também foram alocados avaliadores extra às sessões, quando necessário. Dessa forma, a Tabela 4 apresenta nas colunas *Extra* a quantidade de avaliadores desse tipo utilizados na solução manual e nas soluções encontradas pela abordagem proposta. Os resultados que possuem o marcador $*$ são aqueles em que os avaliadores extra utilizados foram oriundos de uma fonte externa, isto é, uma relação de professores que não atuam como orientadores no evento. A Seção 4.3 aborda uma análise desta tabela destacando a melhoria em relação ao número de saltos dados pelos avaliadores durante o evento.

4.3 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 4. Assim como na Tabela 3, os resultados para cada instância foram organizados por ano e local.

A Tabela 4 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 3.4.2 (coluna *Abordagem proposta*). As colunas *Saltos* contêm o número de saltos total de cada instância para os dois tipos de soluções. Por fim, as colunas *Sessões* contêm o número total de sessões necessárias para comportar o evento nas soluções manual e automática.

Tabela 3 – Comparação entre os benefícios totais obtidos manualmente e pelo Algoritmo 1.

			Algoritmo 1		
Ano	**Tamanho**	**Manual**	**Melhor**	**Média**	**Ganho(%)**
2014	96	3224052	3332066	3332066,0	3,35
	188	6007422	6044295	6044018,3	0,61
	251	6705749	6755630	6755289,2	0,74
	221	7012928	7057300	7055009,6	0,60
	136	4840881	4889191	4888043,4	0,97
2015	119	3439980	3467553	3466912,8	0,78
	249	8244142	8296936	8296662,0	0,63
	259	7649096	7714829	7714209,2	0,85
	169	5278741	5412770	5412117,8	2,53
	127	4162610	4299735	4298415,4	3,26
2016	24	800917	800932	800932,0	<0,01
	94	2546256	2576575	2576300,6	1,17
	167	5391373	5429405	5429321,4	0,70
	230	8447639	8532255	8532145,2	1,00
	182	5124404	5161383	5161058,2	0,71
	131	5902189	6019879	6019227,8	1,98
	62	2615560	2625286	2625286,0	0,37
	60	1822026	1834556	1834556,0	0,69
2017	204	17213671	17300681	17289175,0	0,43
	110	8343761	8364389	8360882,0	0,20
	348	18079154	18208461	18204474,4	0,69
	209	11830197	11853440	11851924,6	0,18
	201	11845530	11882176	11881769,6	0,30
	46	3024320	3026096	3026096,0	0,06
	54	2774178	2813434	2813351,6	1,41
	200	15099595	15238356	15235389,8	0,90
2018	125	913035	994925	994778,6	8,95
	74	591527	662933	662911,6	12,07
	91	514667	527838	527563,2	2,50
	60	301136	302300	302292,2	0,38
	414	4005073	4143960	4143791,6	3,46
	57	273958	274098	274098,0	0,05
	211	1105132	1183712	1183607,4	7,10
	218	1554839	1661380	1661136,0	6,83
	197	1394753	1437481	1437419,6	3,05

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.

4.4 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 5, em que as linhas estão dispostas na mesma ordem da Tabela 3. 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 4 – Comparação entre o número de avaliadores extra alocados e os saltos obtidos manualmente e pelo Algoritmo 1.

		Manual			**Abordagem proposta**		
Ano	**Tamanho**	**Saltos**	**Extra**	**Sessões**	**Saltos**	**Extra**	**Sessões**
2014	96	11	1	10	2,0	0,0	9,0
	188	12	11	20	0,2	0,8	18,4
	251	13	31	26	1,0	3,0	26,0
	221	8	18	23	0,0	1,0	22,0
	136	9	14	15	1,0	1,5	15,0
2015	119	7	18	14	0,0	0,0	12,0
	249	65	34	28	0,8	2,6	26,6
	259	59	30	32	0,0	1,8	25,2
	169	36	12	18	0,0	0,5	16,8
	127	36	18	15	1,2	0,0	15,0
2016	24	8	6	5	0,0	2,0*	5,0
	94	17	13	10	0,2	0,0	8,2
	167	25	14	18	0,2	0,0	14,6
	230	54	12	24	0,0	1,0	20,6
	182	39	19	18	0,0	0,0	15,4
	131	39	20	13	0,0	0,0	12,0
	62	25	17	8	0,0	0,0	5,0
	60	19	6	9	0,0	0,0	7
2017	204	164	13	21	0,0	4,4	16,8
	110	49	12	11	1,0	0,0	10,0
	348	166	44	36	0,0	4,2	31,6
	209	112	18	22	0,0	2,8	20,4
	201	178	18	21	0,2	1,6	19,2
	46	37	4	5	0,0	0,0	5,0
	54	38	1	6	0,0	1,0	6,0
	200	92	7	21	0,0	1,8	18,0
2018	125	34	15	12	0,0	2,2*	12,0
	74	19	11	8	0,9	1,3*	8,0
	91	4	11	9	0,0	0,0	8,0
	60	14	2	6	0,0	0,0	6,0
	414	39	13	39	0,0	8,6	37,4
	57	32	4	6	0,0	0,0	6,0
	211	72	7	20	0,0	1,4	19,6
	218	59	16	21	0,0	2,6	19,8
	197	52	15	19	0,0	2,6	19,2

Tabela 5 – Tempo de cada etapa do algoritmo.

Solução Inicial	**Busca Local**	**Perturbação**	**Checador**	**Formulação**
< 0,01 (0,01%)	14,29 (69,19%)	0,04 (0,21%)	0,22 (1,07%)	6,09 (29,52%)
0,01 (< 0,01%)	130,91 (94,08%)	0,25 (0,18%)	0,86 (0,62%)	7,10 (5,11%)
0,01 (< 0,01%)	339,46 (94,93%)	1,62 (0,46%)	0,94 (0,26%)	15,55 (4,35%)
0,01 (< 0,01%)	214,03 (96,47%)	0,23 (0,10%)	1,19 (0,54%)	6,39 (2,88%)
< 0,01 (0,01%)	46,69 (89,99%)	0,21 (0,42%)	0,09 (0,18%)	4,87 (9,39%)
< 0,01 (0,01%)	29,45 (86,95%)	0,11 (0,33%)	0,26 (0,78%)	4,40 (11,92%)
0,01 (< 0,01%)	418,22 (96,10%)	1,26 (0,29%)	1,11 (0,26%)	14,56 (3,35%)
0,01 (< 0,01%)	652,93 (97,08%)	0,75 (0,11%)	1,39 (0,21%)	17,45 (2,59%)
0,01 (0,01%)	250,33 (94,34%)	3,20 (1,21%)	0,62 (0,24%)	11,17 (4,21%)
< 0,01 (0,01%)	83,69 (94,68%)	0,26 (0,30%)	0,16 (0,19%)	4,26 (4,83%)
< 0,01 (0,08%)	0,88 (82,33%)	0,02 (1,58%)	0,05 (5,08%)	0,12 (10,92%)
< 0,01 (0,01%)	12,90 (40,84%)	0,03 (0,11%)	0,20 (0,63%)	18,46 (58,41%)
< 0,01 (0,01%)	74,51 (74,85%)	0,13 (0,14%)	0,78 (0786%)	24,11 (24,22%)
0,01 (< 0,01%)	302,95 (96,03%)	0,34 (0,11%)	0,86 (0,27%)	11,31 (3,59%)
0,01 (< 0,01%)	166,92 (60,04%)	0,23 (0,08%)	0,76 (0,28%)	110,10 (39,60%)
< 0,01 (0,01%)	39,67 (61,78%)	0,13 (0,21%)	0,20 (0,32%)	24,19 (37,68%)
< 0,01 (0,01%)	5,27 (17,97%)	0,02 (0,19%)	0,09 (0,31%)	23,93 (81,61%)
< 0,01 (< 0,01 %)	4,48 (6,86%)	0,02 (0,04%)	0,07 (0,12%)	60,86 (92,99%)
0,01 (< 0,01%)	157,24 (98,19%)	0,18 (0,11%)	0,20 (0,13%)	2,51 (1,57%)
< 0,01 (< 0,01%)	15,44 (27,53%)	0,05 (0,10%)	0,14 (0,26%)	40,45 (72,10%)
0,01 (< 0,01%)	720,17 (98,63%)	0,46 (0,06%)	2,12 (0,29%)	7,36 (1,01%)
0,01 (0,01%)	119,64 (92,46%)	0,15 (0,12%)	0,37 (0,29%)	9,21 (7,13%)
0,01 (0,01%)	147,10 (95,76%)	0,17 (0,12%)	0,89 (0,58%)	5,43 (3,54%)
< 0,01 (0,05%)	1,02 (62,19%)	0,01 (0,52%)	0,10 (5,93%)	0,51 (31,93%)
< 0,01 (0,04%)	2,04 (81,98%)	0,01 (0,55%)	0,12 (5,12%)	0,30 (12,31%)
0,01 (0,01%)	126,14 (96,19%)	0,16 (0,13%)	0,52 (0,40%)	4,29 (3,28%)
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%)	6,62 (24,74%)	0,02 (0,07%)	0,11 (0,41%)	20,00 (74,76%)
< 0,01 (0,02%)	2,28 (37,88%)	0,01 (0,21%)	0,10 (1,75%)	3,63 (60,14%)
0,02 (< 0,01%)	1267,43 (98,84%)	0,59 (0,05%)	1,13 (0,09%)	13,16 (1,03%)
< 0,01 (0,06%)	2,24 (89,29%)	0,01 (0,5%)	0,10 (4,34%)	0,14 (5,74%)
0,01 (0,01%)	145,82 (80,27%)	0,17 (0,10%)	0,29 (0,16%)	35,36 (19,47%)
0,01 (0,01%)	81,95 (85,69%)	0,10 (0,10%)	0,57 (0,60%)	13,01 (13,60%)
< 0,01 (0,01%)	48,09 (86,84%)	0,06 (0,11%)	0,28 (0,51%)	6,93 (12,52%)

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, alocação de avaliadores a sessões 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.

Algumas sugestões para trabalhos futuros são: adaptar o método proposto para que se torne genérico o suficiente para que possa ser utilizado por diversas conferências de outras instituições; propor instâncias artificiais para a realização de experimentos com fins de *benchmark*; e desenvolver uma ferramenta de apoio a tomada de decisão que permita não só realizar a programação de conferências de forma automática, mas também alterar a solução respeitando as restrições do problema.

Referências

ANDERSON, I. *Combinatorial designs and tournaments*. [S.l.]: Oxford University Press, 1997. v. 6.

BULHÕES, T.; CORREIA, R.; SUBRAMANIAN, A. Conference scheduling: A clustering-based approach. *European Journal of Operational Research*, v. 297, n. 1, p. 15–26, 2022. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0377221721003775>.

BURKE, E. K. et al. A classification of hyper-heuristic approaches. *Handbook of metaheuristics*, Springer, p. 449–468, 2010.

CASTAÑO, F.; VELASCO, N.; CARVAJAL, J. Content-based conference scheduling optimization. *IEEE Latin America Transactions*, v. 17, n. 04, p. 597–606, 2019.

CORREIA, R. et al. Scheduling the brazilian or conference. *Journal of the Operational Research Society*, Taylor & Francis, v. 73, n. 7, p. 1487–1498, 2022. Disponível em: <https://doi.org/10.1080/01605682.2021.1915194>.

CORREIA, R. G. et al. 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*. [S.l.: s.n.], 2018.

EDIS, E.; EDIS, R. S. An integer programming model for the conference timetabling problem-konferans çizelgeleme problemi için bir tamsayili programlama modeli. *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, Citeseer, v. 9, n. 2, p. 55–62, 2013.

EGLESE, R. W.; RAND, G. K. Conference seminar timetabling. *Journal of the Operational Research Society*, Taylor & Francis, v. 38, n. 7, p. 591–598, 1987.

ELTAYEB, I. S.; AHMED, A. S. A comparison of selection hyper-heuristic approaches on the conference scheduling optimization problem. In: IEEE. *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*. [S.l.], 2021. p. 1–6.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, p. 109–133, 1995.

FUKASAWA, R. et al. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, v. 106, n. 3, p. 491–511, May 2006. ISSN 1436-4646. Disponível em: <https://doi.org/10.1007/s10107-005-0644-x>.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. *Operations research*, INFORMS, v. 9, n. 6, p. 849–859, 1961.

IBRAHIM, H.; RAMLI, R.; HASSAN, M. H. Combinatorial design for a conference: constructing a balanced three-parallel session schedule. *Journal of Discrete Mathematical Sciences and Cryptography*, Taylor & Francis, v. 11, n. 3, p. 305–317, 2008.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *science*, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2003. p. 320–353.

MARENCO, J. Designing the claio 2022 conference program with combinatorial optimization techniques. *Event Management*, Cognizant Communication Corporation, 2023.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097 – 1100, 1997. ISSN 0305-0548.

NICHOLLS, M. G. A small-to-medium-sized conference scheduling heuristic incorporating presenter and limited attendee preferences. *Journal of the Operational Research Society*, Taylor & Francis, v. 58, n. 3, p. 301–308, 2007.

PADBERG, M.; RINALDI, G. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations research letters*, Elsevier, v. 6, n. 1, p. 1–7, 1987.

POTTHOFF, R. F.; BRAMS, S. J. Scheduling of panels by integer programming: Results for the 2005 and 2006 new orleans meetings. *Public Choice*, v. 131, n. 3, p. 465–468, Jun 2007. ISSN 1573-7101. Disponível em: <https://doi.org/10.1007/s11127-006-9126-9>.

POTTHOFF, R. F.; MUNGER, M. C. Use of integer programming to optimize the scheduling of panels at annual meetings of the public choice society. *Public Choice*, v. 117, n. 1, p. 163–175, Oct 2003. ISSN 1573-7101.

QUESNELLE, J.; STEFFY, D. Scheduling a conference to minimize attendee preference conflicts. In: *Proceedings of the 7th multidisciplinary international conference on scheduling: theory and applications (MISTA)*. [S.l.: s.n.], 2015. p. 379–392.

RIQUELME, F. et al. A track-based conference scheduling problem. *Mathematics*, v. 10, n. 21, 2022. ISSN 2227-7390. Disponível em: <https://www.mdpi.com/2227-7390/10/21/3976>.

SAMPSON, S. E. Practical implications of preference-based conference scheduling. *Production and Operations Management*, Wiley Online Library, v. 13, n. 3, p. 205–215, 2004.

SAMPSON, S. E.; WEISS, E. N. Increasing service levels in conference and educational scheduling: A heuristic approach. *Management Science*, INFORMS, v. 41, n. 11, p. 1816–1825, 1995.

SILVA, J. F. N. da et al. O problema de programação de sessões técnicas de conferências com avaliadores. Galoá, 2020.

SILVA, P. et al. Alocação de sessões de artigos em eventos acadêmicos: Modelo e estudo de caso. *Pesquisa Operacional para o Desenvolvimento*, v. 6, n. 1, p. 54–66, nov. 2013.

STIDSEN, T.; PISINGER, D.; VIGO, D. Scheduling euro-k conferences. *European Journal of Operational Research*, Elsevier, v. 270, n. 3, p. 1138–1147, 2018. ISSN 0377-2217.

SUBRAMANIAN, A. et al. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, n. 11, p. 1899–1911, 2010. ISSN 0305-0548. Metaheuristics for Logistics and Vehicle Routing. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0305054809002779>.

TANAKA, M.; MORI, Y.; BARGIELA, A. Granulation of keywords into sessions for timetabling conferences. *Proceedings of soft computing and intelligent systems (SCIS 2002)*, v. 2002, p. 1–5, 2002.

TERMINI, S. *T. Kohonen, self-organizing maps: Springer-Verlag, Berlin, Heidelberg, New York, 1995, XVI+ 362pp.* [S.l.]: Springer, 1995.

THOMPSON, G. M. Improving conferences through session scheduling. *Cornell Hotel and Restaurant Administration Quarterly*, v. 43, n. 3, p. 71–76, 2002.

VANGERVEN, B. et al. Conference scheduling—a personalized approach. *Omega*, Elsevier, v. 81, p. 38–47, 2018.