Conference Program Scheduling using Genetic Algorithms

Rucha Deshpande
Purdue University, USA
deshpa37@purdue.edu

Aishwarya Devi Akila Pandian
Purdue University, USA
aakilapa@purdue.edu

Vigneshwaran Dharmalingam
Purdue University, USA
vdharmal@purdue.edu

ABSTRACT

Program creation is the process of allocating presentation slots for each paper accepted to a conference with parallel sessions. This process, generally done manually, involves intricate decision-making to align with multiple constraints and optimize goals. The total time for all presentations in a session cannot be longer than the length of the session, the total number of sessions has to be equal to the number of sessions provided by the Program Committee chairs, and if there are parallel tracks, no two papers with common authors can be scheduled in parallel at the same time. We propose a Conference Program Scheduler using Genetic Algorithms to automate the conference schedule creation process while addressing these constraints and maximizing session theme coherence. Our evaluation focuses on the algorithm's ability to meet the outlined constraints and its success in creating thematically coherent and time-efficient sessions.

CCS CONCEPTS

• **Software and its engineering** → *Designing software*; • **Computing methodologies** → **Artificial intelligence**; • **Theory of computation** → **Bio-inspired optimization**.

KEYWORDS

Automated Scheduling, Genetic Algorithms, Optimization Algorithms

ACM Reference Format:
Rucha Deshpande, Aishwarya Devi Akila Pandian, and Vigneshwaran Dharmalingam. 2024. Conference Program Scheduling using Genetic Algorithms. In *Proceedings of the 1st ACM International Conference on AI-Powered Software (AIware '24), July 15–16, 2024, Porto de Galinhas, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3664646.3665086

1 APPROACH

We use a Genetic Algorithm (GA) [1, 3] for our AIware as it can efficiently explore and optimize complex solution spaces. The input data comprises the number of conference sessions, the length of each session, the number of parallel tracks, and the accepted papers in a particular conference along with all the corresponding data: title, topics, abstract, paper, authors and allocated time for the presentation of each paper. Applying the Genetic Algorithm to the conference scheduling problem involves several key components:

AIware '24, July 15–16, 2024, Porto de Galinhas, Brazil

ACM ISBN 979-8-4007-0685-1/24/07
https://doi.org/10.1145/3664646.3665086

1.1 Representation

Each schedule is conceptualized as a "solution" or individual in the GA's population. We encode a schedule as a collection of "Session" objects, each comprising the given number of parallel tracks. A track is a list of "Paper" objects, representing papers assigned to that track. The "Paper" class includes attributes such as "id", "authors", "duration", and "topic", which are necessary for satisfying the scheduling constraints.

1.2 Fitness Function

The fitness function quantifies how well a schedule meets the predefined constraints and objectives, returning a fitness score that guides the selection process. In our implementation, the fitness function considers five types of penalties:

Time Penalty (t): Penalizes solutions where the total duration of papers in any track exceeds the maximum allowed session length.

Author Penalty (a): Applied when the same author is scheduled to present in parallel tracks within the same session.

Inclusion Penalty (i): Penalizes solution that fails to include any papers in the conference program.

Dissimilarity Penalty (d): This penalty is calculated using the entropy measure, thus measuring the randomness of topics in a given track of a session.

Utilization Penalty (u): A penalty for significantly under utilizing the available session time, given by the amount of unutilized time for every session.

All the individual penalties obtained above are normalized. The total penalty is a weighted sum of the individual penalties:

$$\text{Total Penalty} = w_1 \cdot t + w_2 \cdot a + w_3 \cdot i + w_4 \cdot d + w_5 \cdot u \quad (1)$$

$$\text{Fitness Score} = \frac{1}{1 + \text{Total Penalty}} \quad (2)$$

We fine-tune the weights during experiments to satisfy hard and soft constraints. This fitness function allows us to balance between strict adherence to constraints and optimization goals, guiding the GA towards feasible and coherent schedules.

1.3 Genetic Operations

1.3.1 Selection. We implement a selection mechanism to choose parent solutions for producing the next generation. We implement Tournament selection [5] to favor individuals with high fitness scores as well as strike the balance between the exploitation of the best solutions and exploration of the solution space.

1.3.2 Crossover. Crossover encourages the exchange of beneficial traits between solutions. In our system, the crossover operation involves swapping session slots between two schedules. We implement single-point crossover.

1.3.3 Mutation. Mutation introduces random modifications to a solution, preventing premature convergence to local optima and

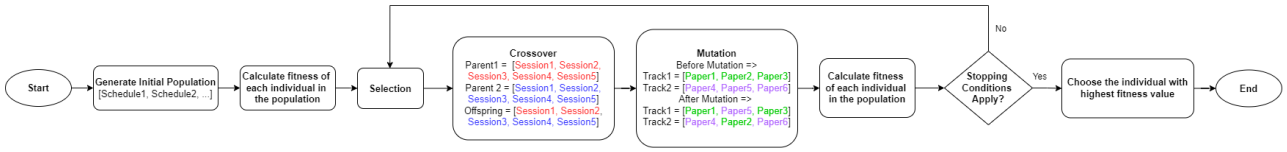

Figure 1: A flowchart showing the working Genetic Algorithm for Conference Scheduling

maintaining genetic diversity within the population. Our approach includes two mutation strategies:

1.3.4 Swap Mutation. This method involves selecting two papers within a session and swapping their parallel tracks. The swap mutation is designed to test the impact of different track assignments on the thematic coherence and the overall fitness of the schedule, without altering the set of papers within a session.

1.3.5 Move Mutation. This approach allows the genetic algorithm to explore adjustments to the schedule by relocating a single paper from its current session to a different session.

2 EXPERIMENTAL SETUP AND EVALUATION

We initialize the population by filling the tracks with the papers in the input data. During initialization, we ensure that a paper is only scheduled within a session if its duration does not exceed the available remaining time. Additionally, we keep track of the authors scheduled in previous tracks of the same session to prevent their papers from being scheduled in parallel. While these constraints may be broken during genetic operations as the algorithm evolves, starting with these settings rather than a random initialization helps the algorithm to converge faster.

We conduct an evaluation using real-world conference data from the MSR 2018 to MSR 2023 programs. For constraint satisfaction, the calculation involves the average percentages of exceeded time per session, authors being scheduled in parallel tracks, and the papers remaining unscheduled. We calculate the average number of topics per session to evaluate thematic coherence. Schedule efficiency evaluates how well the conference schedule utilizes session capacities by calculating the average percentage of unutilized time per session.

We test our approach with population sizes in the range of 50-200, iterating the algorithm through 1,000 to 10,000 generations. We use a small portion of the MSR 2018 data set for adjusting the weights associated with various penalties. Table 1 shows the average values derived from the aggregate data across all test cases.

3 RESULTS AND DISCUSSION

To benchmark our solution, we compare its performance against a baseline Integer Linear Programming (ILP) model [2, 4]. The results show that the GA-based algorithm successfully satisfies the time constraints indicating efficient time management within the sessions. The algorithm manages to avoid any instances where authors are scheduled to present in parallel sessions. There are no instances of papers that are not scheduled and the average unused time per session of 4.6 minutes indicates that the algorithm minimizes idle time while avoiding over-packed sessions. However, the solution

Table 1: Average Results on MSR 2018-2023 data

Metric	GA	ILP
Exceeded Session time (min)	0	0
Authors Scheduled in Parallel	0	0
Papers not scheduled	0	0
Number of topics per track	2.2	5.95
Unused Time Per Session (min)	4.6	6.8

averages two topics per track which does not demonstrate a strong thematic focus. Observations show that in over 80% of the sessions, a single topic is predominant, validating the method's capability to group primary themes, but further fine-tuning is required to generate more thematically coherent sessions. ILP shows a much higher average number of topics per track and longer unused time per session, showing less efficiency than GA.

We observe several key findings related to the performance of our GA approach. When more emphasis is placed on topic coherence by increasing the relative weight of topic coherence penalty, a very slight deviation is observed in the results of other constraints, such as parallel author scheduling. A significant challenge and limitation of this approach is its high sensitivity to the weights, which may not always be straightforward to determine without trial and error. Additionally, GAs require many generations to converge to a good solution and are complex in nature. We implement elitism, which retains the top 10% of solutions from each generation. This method ensures stable and efficient convergence by preserving high-quality genetic material for the next generations. We test our algorithm with a variable number of papers, showing consistent performance across different datasets and ensuring the scalability of the approach. Furthermore, in the event of changes to the conference lineup or schedule, the algorithm can be rerun efficiently with minimal time investment, easily integrating any updates.

The GA-based solution satisfies all hard constraints, similar to human-generated schedules, which are manually designed to meet these requirements. However, human-curated schedules achieve better thematic coherence. The AI approach significantly reduces the total time and effort involved, making scaling up or implementing last-minute changes more manageable compared to the more laborious and error-prone manual method.

Overall, our AIware solution substantially simplifies the conference program creation process. The implementation of our approach is available on GitHub and Zenodo.

ACKNOWLEDGMENTS

We would like to thank Tianyi Zhang (Purdue University) for his constant guidance and support.

REFERENCES

[1] Edmund K Burke, David Elliman, and Rupert Weare. 1994. A genetic algorithm based university timetabling system. In *Proceedings of the 2nd east-west international conference on computer technologies in education*, Vol. 1. Citeseer, 35–40.
[2] Jack E Graver. 1975. On the foundations of linear and integer linear programming I. *Mathematical Programming* 9, 1 (1975), 207–226.
[3] John H Holland. 1992. Genetic algorithms. *Scientific american* 267, 1 (1992), 66–73.
[4] Emilia Kondili, Constantinos C Pantelides, and Roger WH Sargent. 1993. A general algorithm for short-term scheduling of batch operations—I. MILP formulation. *Computers & Chemical Engineering* 17, 2 (1993), 211–227.
[5] Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu. 2005. Comparison of performance between different selection strategies on simple genetic algorithms. In *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, Vol. 2. IEEE, 1115–1121.

Received 2024-04-24; accepted 2024-05-10