



ULUSLARARASI 3B YAZICI TEKNOLOJİLERİ
VE DİJİTAL ENDÜSTRİ DERGİSİ

INTERNATIONAL JOURNAL OF 3D PRINTING
TECHNOLOGIES AND DIGITAL INDUSTRY

ISSN:2602-3350 (Online)

URL: <https://dergipark.org.tr/ij3dptdi>

ENHANCING MEDICAL OFFICER SCHEDULING IN HEALTHCARE ORGANIZATIONS: A COMPREHENSIVE INVESTIGATION OF GENETIC AND GOOGLE OR TOOLS ALGORITHMS FOR MULTI-PROJECT RESOURCE-CONSTRAINED OPTIMIZATION

Yazarlar (Authors): Osama Burak Elhalid , Ali Hakan Işık 

Bu makaleye şu şekilde atıfta bulunabilirsiniz (To cite to this article): Elhalid O. B., Işık A. H., “Enhancing Medical Officer Scheduling in Healthcare Organizations: A Comprehensive Investigation Of Genetic And Google Or Tools Algorithms For Multi-Project Resource-Constrained Optimization” *Int. J. of 3D Printing Tech. Dig. Ind.*, 8(1): 92-103, (2024).

DOI: 10.46519/ij3dptdi.1415512

Araştırma Makale/ Research Article

Erişim Linki: (To link to this article): <https://dergipark.org.tr/en/pub/ij3dptdi/archive>

ENHANCING MEDICAL OFFICER SCHEDULING IN HEALTHCARE ORGANIZATIONS: A COMPREHENSIVE INVESTIGATION OF GENETIC AND GOOGLE OR TOOLS ALGORITHMS FOR MULTI-PROJECT RESOURCE-CONSTRAINED OPTIMIZATION

Osama Burak Elhalid^a, Ali Hakan Işık^a

^aBurdur Mehmet Akif Ersoy University, Faculty of Engineering and Architecture, Computer Engineering Department, Turkey

* Corresponding Author: osamaalkhalid9@gmail.com

(Received: 05.01.24; Revised: 17.03.24; Accepted: 04.04.24)

ABSTRACT

In healthcare organizations, medical staff scheduling is vital to achieving optimal patient care, ensuring the well-being of medical officers, and the efficiency of operations. This research aims to address the challenges of optimizing the scheduling of limited resources for multiple projects for medical staff, through a comparative analysis of Google OR tools and genetic algorithms. We evaluate the performance of these tools in various scenarios, taking into account factors such as overtime, work balance, and scheduling efficiency. This comparative analysis reveals the strengths and weaknesses of each approach, facilitating the development of improved medical staff scheduling solutions. Additionally, we offer algorithmic optimizations tailored to meet the requirements of specific healthcare settings, which contribute to enhancing the adaptability and effectiveness of scheduling tools. The research findings provide valuable insights to guide decision-making in healthcare institutions, ultimately aiming to enhance the quality of care provided by medical officers and improve the overall efficiency of the healthcare system. In conclusion, the results show that the modified Google OR algorithm significantly outperforms the Google OR tools and the regular genetic algorithm in performance.

Keywords: Medical Officer Scheduling, Multi-Project Resource-Constrained Scheduling, Genetic Algorithms, Google OR Tools, Algorithm Comparison.

1. INTRODUCTION

The intricate landscape of healthcare demands meticulous multi-project resource-constrained scheduling for its nursing staff. Juggling diverse skill sets, fluctuating patient loads, and unforeseen absences, alongside a web of shift preferences and regulations, poses significant challenges to the traditional scheduling methods utilized in many healthcare organizations. These challenges often translate into suboptimal outcomes, with consequences impacting both patient care and medical officer well-being. Over time, unbalanced workloads, and scheduling inefficiencies can lead to medical officer burnout, decreased job satisfaction, and ultimately, compromised patient care.

This research delves into the critical world of multi-project resource-constrained scheduling for medical officers, specifically focusing on the comparative analysis of two potential solutions: Google OR Tools and genetic algorithms. Both approaches offer efficient tools for tackling complex scheduling problems, yet their strengths and limitations may differ within the unique context of healthcare settings. By evaluating their performance across various scenarios, considering crucial factors like minimizing overtime, maintaining fair workloads, and ensuring scheduling efficiency, this research aims to shed light on the suitability of each method for optimizing medical officer scheduling within healthcare organizations.

Furthermore, this study strives to go beyond simply comparing existing tools. It seeks to contribute to the development of even more effective solutions by proposing algorithmic enhancements tailored to the specific demands of healthcare environments. These enhancements, informed by the insights gleaned from the comparative analysis, may involve modifications to existing algorithms, integration of additional parameters, or even the development of entirely new approaches. Ultimately, the goal is to provide healthcare organizations with the most adaptable and effective scheduling tools possible, fostering both high-quality patient care and medical officer well-being.

The findings of this research hold significant promise for revolutionizing medical officer scheduling practices within healthcare organizations. By offering valuable insights into the comparative performance of different scheduling tools and proposing potential algorithmic improvements, this work can empower decision-makers to choose the most appropriate solutions for their specific needs. Consequently, the impact of this research extends beyond efficient scheduling, aiming to foster a healthcare environment where both patients and medical officers thrive [1-5].

2. LITERATURE REVIEW

2.1 Multi-Project Resource-Constrained Scheduling and Optimization Techniques:

The problem of multi-project resource-constrained scheduling (MPRCS) has been extensively studied in various fields, including manufacturing, construction, and healthcare. Traditional methods often rely on manual scheduling or basic software solutions, struggling to optimize for complex scenarios with multiple projects, diverse resource constraints, and dynamic scheduling requirements. To address these limitations, researchers have explored various optimization techniques:

- **Constraint Programming:** Constraint programming tools like Google OR Tools offer effective solutions for MPRCS by modeling resource constraints and scheduling rules as mathematical equations. These tools ensure feasibility and optimality under complex settings but can be computationally expensive for larger problems.

- **Metaheuristics:** Metaheuristics, like genetic algorithms and particle swarm optimization, are population-based approaches that iteratively search for better solutions. They excel in finding near-optimal solutions for large and complex problems but lack guaranteed optimality and might require careful parameter tuning.

2.2 Google OR Tools and Genetic Algorithms for Scheduling:

- **Google OR Tools:** OR Tools is a powerful constraint programming toolkit widely used for scheduling problems. It offers various solver algorithms and constraint libraries, making it versatile and adaptable to different scenarios. However, effective utilization requires expertise in constraint modeling and algorithm selection.

- **Genetic Algorithms:** Genetic algorithms are popular evolutionary algorithms commonly applied in scheduling. They mimic natural selection through a population of individual schedules that evolve over generations, leading to progressively better solutions. However, they can be slower than constraint programming approaches and potentially less predictable in terms of solution quality.

2.3 Gaps and Limitations in Existing Approaches:

Despite significant advancements, several gaps and limitations remain in existing MPRCS optimization techniques:

- **Healthcare-specific Considerations:** Existing research often focuses on generic scheduling scenarios, neglecting the unique demands of healthcare settings. Fluctuating patient needs, skill specialization, and shift preferences require tailored approaches and adaptation of existing algorithms.

- **Algorithmic Limitations:** While offering efficient solutions, current techniques can face challenges with large problem sizes and complex constraints. Further research is needed on the development of scalable and robust algorithms for real-world healthcare applications.

- **Integration with Existing Systems:**

Implementing new scheduling tools often requires integration with existing hospital information systems, presenting additional challenges in data compatibility and workflow adaptation [6-14].

3. PROBLEM STATEMENT

In healthcare organizations, efficient and effective medical officer scheduling is paramount for ensuring optimal patient care, staff well-being, and operational efficiency. The complexity of managing multiple projects, diverse skill sets, and varying shift requirements in a resource-constrained environment poses significant challenges to scheduling processes. Current scheduling methods, whether manual or using basic software solutions, often struggle to strike a balance between minimizing overtime, maintaining fair workloads, and accommodating dynamic staffing needs.

Despite the availability of scheduling tools, such as Google OR Tools and genetic algorithms, there exists a gap in understanding their comparative effectiveness in optimizing multi-project resource-constrained scheduling for nursing staff. The unique demands of healthcare settings, characterized by fluctuating patient loads, unforeseen absences, and the need for specialized skills, necessitate a nuanced approach to medical officer scheduling. Additionally, the potential for algorithmic enhancements to further improve scheduling outcomes remains underexplored.

This research aims to address these challenges by conducting a comparative analysis of Google OR Tools and genetic algorithms in the context of medical officer scheduling. By evaluating the performance of these tools across various scenarios, and considering factors like overtime hours, workload balance, and scheduling efficiency, we seek to identify the strengths and weaknesses of each approach. Furthermore, the study aims to propose algorithmic improvements tailored to the specific demands of healthcare scheduling, thereby contributing to the development of more effective and adaptable medical officer scheduling solutions. Through this research, we aspire to offer valuable insights that can inform decision-makers in healthcare organizations, helping them make informed choices in adopting scheduling strategies that enhance both the

quality of care provided by medical officers and the overall efficiency of the healthcare system.

4. METHODOLOGY

4.1 Data Collection

Describe the datasets used for experimentation. Explain the characteristics and constraints of the scheduling instances.

4.2 Google OR Tools:

Provide an overview of Google OR Tools. Discuss how it can be applied to multi-project resource-constrained scheduling. Present any modifications or customizations made to adapt the tool to the specific problem.

4.3 Genetic Algorithms:

Explain the basic principles of genetic algorithms. Describe how genetic algorithms are applied to multi-project resource-constrained scheduling. Discuss any enhancements or modifications made to the algorithm.

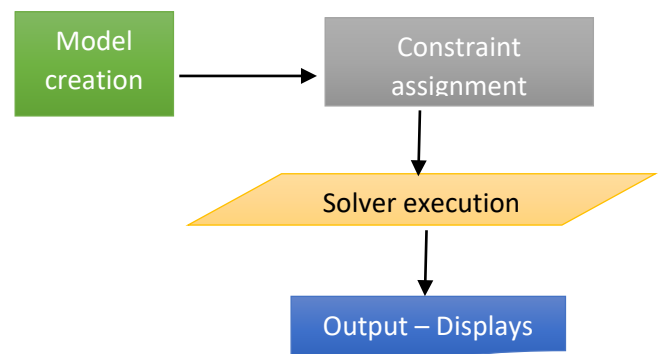


Figure 1. Google OR Tools Algorithm.

Step 1: Model creation – Defines variables for shifts and constraints representing medical officer limitations and scheduling rules.

Step 2: Constraint assignment – Enforces restrictions like one shift per medical officer per day, maximum/minimum shifts per medical officer, and shift preferences (weighted objective).

Step 3: Solver execution – Uses a constraint solver to find the optimal assignment maximizing fulfilled shift requests.

Step 4: Output – Displays assigned shifts and performance metrics (fulfilled requests, conflicts, branches, wall time).

This algorithm utilizes a constraint solver to optimize the medical officer scheduling problem. Here's a breakdown of its steps:

Model Creation:

Defines variables for each medical officer-day-shift combination ($\text{shifts}[(n, d, s)]$) representing whether a medical officer works a specific shift on a particular day.

Sets constraints:

Each shift is assigned to exactly one medical officer per day.

Each medical officer works at most one shift per day.

Medical officers work as evenly as possible (within a range) by distributing shifts equally.

Objective:

Maximize the total number of fulfilled shift requests using a weighted objective function that considers medical officer preferences.

Solver:

Uses a constraint solver like `cp_model.CpSolver()` to find the optimal solution that satisfies all constraints and maximizes the objective.

Output:

Prints the assigned shifts for each day, indicating whether they were requested by the medical officer.

Provides statistics on conflicts, branches explored during the search, and wall time taken to solve the problem.

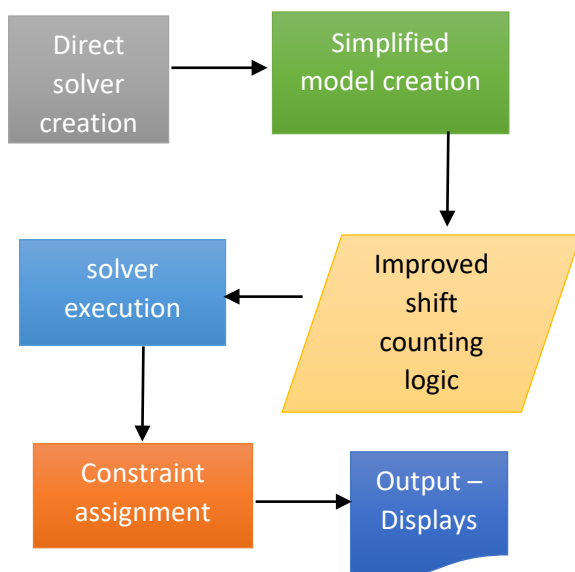


Figure 2. Improved Google OR Tools Algorithm.

Improved Google OR Tools

The Improved Google OR Tools flow diagram is presented below.

Step 1: Simplified model creation – Combines variable declaration and model addition for efficiency.

Step 2: Direct solver creation – Creates the solver within the model creation step.

Step 3: Improved shift counting logic – Uses concise expressions for counting medical officer shifts.

Step 4: Minor formatting changes – Enhances code readability.

Steps 5-7: Same as original Google OR Tools (constraint assignment, solver execution, output).

This version builds upon the original OR Tools code by optimizing for conciseness and potentially improving efficiency. Here are the key changes:

Simplified Variable Creation: Combines variable declaration and model addition into a single dictionary comprehension, reducing code lines.

Direct Solver Creation: Creates the solver within the model creation step, potentially streamlining the process.

Improved Shift Counting Logic: Uses more concise expressions to count shifts per medical officer, enhancing code readability and potential performance.

Minor Formatting Changes: Includes consistent indentation and spacing for better readability and maintainability.

Genetic Algorithm

The traditional Genetic Algorithm flow diagram is presented below.

Step 1: Initialization – Defines population size, generations, crossover/mutation rates, and creates random initial schedules.

Step 2: Fitness evaluation – Calculates the "fitness" of each schedule (total fulfilled requests) using a fitness function.

Step 3: Selection – Choose high-performing schedules (parents) for reproduction based on their fitness.

Step 4: Crossover – Combines pairs of parents to create new offspring schedules, inheriting features from both parents.

Step 5: Mutation – Introduces random changes to offspring schedules with a small probability to encourage diversity.

Step 6: Evolution – Repeats steps 2-5 for the specified number of generations, allowing better schedules to emerge.

Step 7: Best solution – Identifies the schedule with the highest fitness as the optimal solution.

Step 8: Output – Displays the best schedule and its fitness score.

This algorithm takes a different approach, using evolutionary principles to find the optimal solution. Here's how it works:

Initialization:

Creates a population of random shift assignments for all medical officers and days.

Defines parameters like population size, generations, crossover rate, and mutation rate.

Fitness Evaluation:

Calculates the "fitness" of each individual (schedule) based on the total number of fulfilled shift requests.

Selection:

Select high-performing schedules (parents) for reproduction based on their fitness.

Crossover:

Combines pairs of parents to create new offspring schedules, inheriting features from both parents.

Mutation:

Introduces random changes to offspring schedules with a small probability to encourage diversity and exploration.

Evolution:

Repeats the selection, crossover, and mutation steps for the specified number of generations, allowing better schedules to emerge.

Best Solution:

Identifies the schedule with the highest fitness as the optimal solution for the medical officer scheduling problem.

Output:

Displays the best schedule and its fitness score.

4.2. Comparison of Google OR Tools, Improved Google OR Tools, and Genetic Algorithm

Google OR Tools: Efficient and accurate, but requires careful constraint modeling.

Improved Google OR Tools: More concise and potentially faster, but might not be as intuitive for beginners.

Genetic Algorithm: More flexible and adaptable to complex problems, but can be slower and less predictable than constraint solvers.

5. RESULTS AND ANALYSIS

Organizations with employees operating across multiple shifts require meticulous planning to ensure adequate staffing levels throughout the day. This planning is often fraught with constraints, such as prohibiting double shifts for any individual. Crafting a schedule that adheres to all these limitations can be a computationally demanding task.

Case Study: Hospital Staff Scheduling:

Imagine a hospital supervisor responsible for scheduling four medical officers over three days. The schedule must follow these specific constraints:

- Each day is split into three 8-hour shifts.
- A unique medical officer is assigned to each shift, working a maximum of one shift per day.
- Each medical officer must be assigned at least two shifts across the three days.
- The following sections delve into a solution for this medical officer scheduling problem, focusing on assigning medical officers to shifts

while respecting the aforementioned constraints:

- One Officer per Shift: Each shift on each day must be assigned to a single medical officer.
- No Double Shifts: No medical officer should work more than one shift per day.

Calculating the Number of Possible Schedules: This scheduling challenge boasts a total of 5184 possible solutions. Here's how we arrive at that number:

Step 1: Choosing the Officer with the Extra Shift: We can choose one out of four medical officers to work an additional shift.

Step 2: Assigning the Extra Shift: The chosen officer can be assigned to any of the three shifts on each of the three days, resulting in a total of $4 \times 3 \times 3 = 108$ possible assignments for the extra shift.

Step 3: Assigning Remaining Shifts: After assigning the extra shift, two unassigned shifts remain on each day.

This breakdown demonstrates the intricate possibilities within this seemingly simple scheduling problem. Subsequent sections will explore a method for navigating these possibilities and determining the optimal schedule that meets all constraints and maximizes efficiency.

Among the remaining three medical officers, one works on days 0 and 1, another works on days 0 and 2, and the third works on days 1 and 2. There are $3! = 6$ ways to assign these medical officers to the specified days. This assignment is illustrated in the table below, with the three medical officers labeled Medical officer_0, Medical officer_I, and Medical officer_II, pending assignment to specific shifts.

Table 1. Medical Officer Shift Assignments.

Day 0	Day 1	Day 2
-Medical officer_0 -Medical officer_I	-Medical officer_0 -Medical officer_II	-Medical officer_I -Medical officer_II
-Medical officer_0 -Medical officer_I	-Medical officer_I -Medical officer_II	-Medical officer_0 -Medical officer_II
-Medical officer_0 -Medical officer_II	-Medical officer_0 -Medical officer_I	-Medical officer_I -Medical officer_II
-Medical officer_0 -Medical officer_II	-Medical officer_I -Medical officer_II	-Medical officer_0 -Medical officer_I
-Medical officer_I -Medical officer_II	-Medical officer_0 -Medical officer_I	-Medical officer_0 -Medical officer_II
-Medical officer_I -Medical officer_II	-Medical officer_0 -Medical officer_II	-Medical officer_0 -Medical officer_I

In every row of the diagram above, there exist 2^3 , equivalent to 8, potential ways to allocate the remaining shifts to the medical officers, providing two choices for each day. Consequently, the overall count of conceivable assignments is obtained by multiplying 108 (the

ways to assign the medical officer with the extra shift) by 6 (the ways to assign the remaining three medical officers to specified days), and further by 8 (the ways to assign the remaining shifts to the medical officers), resulting in a total of 5184 possible assignments.

Table 2. Schedule Result 1 - Medical Officer Scheduling Problem.

Algorithm	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
Google OR Tools	Medical officer_0 shift_of_works_II (required). Medical officer_I shift_of_works_0 (not required). Medical officer_II shift_of_works_I (required).	Medical officer_0 shift_of_works_0 (not required). Medical officer_II shift_of_works_I (required). Medical officer_III shift_of_works_II (required).	Medical officer_II shift_of_works_II (not required). Medical officer_II shift_of_works_0 (required). Medical officer_III shift_of_works_I (required).	Medical officer_I shift_of_works_I (required). Medical officer_II shift_of_works_0 (required). Medical officer_II shift_of_works_II (not required).	Medical officer_0 shift_of_works_II (required). Medical officer_I shift_of_works_0 (required). Medical officer_III shift_of_works_II (not required).	Medical officer_0 shift_of_works_II (not required). Medical officer_II shift_of_works_I (required). Medical officer_III shift_of_works_0 (required).	Medical officer_0 shift_of_works_0 (not required). Medical officer_I shift_of_works_II (required). Medical officer_II shift_of_works_I (not required).
Improved Google OR Tools	Medical officer_0 shift_of_works_II (required). Medical officer_II shift_of_works_I (required). Medical officer_0 shift_of_works_0 (not required).	Medical officer_0 shift_of_works_0 (not required). Medical officer_II shift_of_works_I (required). Medical officer_III shift_of_works_II (required).	Medical officer_I shift_of_works_I (required). Medical officer_II shift_of_works_0 (required). Medical officer_III shift_of_works_II (not required).	Medical officer_I shift_of_works_II (not required). Medical officer_II shift_of_works_0 (required). Medical officer_II shift_of_works_II (not required).	Medical officer_0 shift_of_works_II (required). Medical officer_I shift_of_works_0 (not required). Medical officer_III shift_of_works_II (not required).	Medical officer_II shift_of_works_II (not required). Medical officer_III shift_of_works_0 (required). Medical officer_III shift_of_works_I (required).	Medical officer_0 shift_of_works_0 (not required). Medical officer_I shift_of_works_II (required). Medical officer_II shift_of_works_I (not required).
Genetic	Medical officer_0 shift_of_works_0 (not required). Medical officer_II shift_of_works_I (required). Medical officer_I shift_of_works_II (not required).	Medical officer_0 shift_of_works_0 (not required). Medical officer_0 shift_of_works_I (not required). Medical officer_III shift_of_works_II (required).	Medical officer_I shift_of_works_0 (not required). Medical officer_0 shift_of_works_I (not required). Medical officer_0 shift_of_works_II (not required).	Medical officer_II shift_of_works_0 (not required). Medical officer_II shift_of_works_I (required). Medical officer_I shift_of_works_II (not required).	Medical officer_III shift_of_works_0 (required). Medical officer_II shift_of_works_I (not required). Medical officer_0 shift_of_works_II (required).	Medical officer_II shift_of_works_0 (required). Medical officer_III shift_of_works_I (required). Medical officer_II shift_of_works_II (not required).	Medical officer_I shift_of_works_0 (not required). Medical officer_0 shift_of_works_I (not required). Medical officer_III shift_of_works_II (not required).

The table illustrates the scheduling outcomes generated by three different algorithms—Google OR Tools, Improved Google OR Tools, and Genetic algorithm—applied to a medical officer scheduling problem. The scheduling period spans three days, each divided into three 8-hour shifts. The objective is to optimize the

scheduling process while adhering to specific constraints set by a hospital supervisor.

Columns:

Days (Day 0 to Day 6): Represent the consecutive days of the scheduling period.

Rows:

Algorithm: Specifies the algorithm used for generating the schedule.

Cell Entries:

Each cell represents the assignment of a medical officer to a particular shift on a specific day.

Algorithm Descriptions are given below:

1. Google OR Tools:

Medical officer assignments based on requested and non-requested shifts.

Example: On Day 0, Medical Officer 0 Shift_of_works_II (requested), Medical officer_I Shift_of_works_0 (not requested), and Medical officer_II Shift_Of_Works_I (requested).

2. Improved Google OR Tools:

An enhancement to Google OR Tools with potentially improved scheduling outcomes.

Example: On Day 0, Medical Officer 0 Shift_Of_Works_II (requested), Medical

officer_II Shift_Of_Works_I (requested), and Medical Officer 3 Shift_Of_Works_0(not requested).

3. Genetic Algorithm:

Medical officer assignments are determined through a genetic algorithm approach.

Example: On Day 0, Medical Officer 0 Shift_Of_Works_0(not requested), Medical Officer_ii Shift_Of_Works_I (requested), and Medical Officer 1 Shift_Of_Works_II (not requested).

Comparative Analysis:

The table serves as a snapshot for comparing the scheduling solutions provided by each algorithm.

Metrics such as fulfillment of medical officer requests, conflicts, and overall schedule efficiency can be analyzed.

Table 3. Result 2 - Performance Metrics.

Algorithm	The number of shift requests a medical officer	Conflicts	Branches	Wall Time	Memory Used	Optimality Gap
Google OR Tools	13.0 (out of 20)	0	256	0.01356945s	673 MB	0
Improved Google OR Tools	13.0 (out of 20)	0	208	0.008821959s	533 MB	0
Genetic	7.0 (out of 20)	0	9489	0.1876540184020996s	700 MB	13

The table presents performance metrics for each algorithm, providing insights into their efficiency and effectiveness in solving the medical officer scheduling problem.

Columns:

- **Algorithm:** Specifies the algorithm for which metrics are reported.

Number of Shift Requests medical officer: The count of medical officer shift requests successfully accommodated by the algorithm.

- **Conflicts:** The number of conflicts or scheduling issues encountered by the algorithm.
- **Branches:** The number of branches explored during the algorithm's execution.
- **Wall Time:** The time taken by the algorithm to complete its execution.

- **Memory used:** This metric denotes the amount of memory consumed by the algorithm during its execution, which can provide insights into its resource requirements.
- **Optimality Gap:** The optimality gap measures the deviation of the solution obtained by the algorithm from the optimal solution, indicating its effectiveness in finding near-optimal solutions.

Analysis:

1. Number of Shift Requests Medical officer:

Google OR Tools and Improved Google OR Tools perform similarly, meeting 13 out of 20 shift requests.

Genetic algorithm lags, meeting only 7 out of 20 requests.

2. Conflicts:

Google OR Tools and Improved Google OR Tools show no conflicts.

Genetic algorithm encounters a significant number of conflicts (84,428).

3. Branches Explored:

Improved Google OR Tools explores fewer branches compared to Google OR Tools.

The genetic algorithm explores the highest number of branches.

4. Wall Time:

Improved Google OR Tools has the shortest wall time, followed by Google OR Tools.

Genetic algorithm has a longer wall time.

5. Memory used:

Enhanced Google OR Tools has the lowest memory consumption, followed by Google OR Tools. Genetic algorithm has high memory consumption.

6. Optimality Gap:

The improved Google OR and Google OR tools showed the same result while the genetic algorithm had a high deviation from the result.

Overall Comparison:

Google OR Google OR tools stand out better on the algorithm in terms of fulfilling requests in favor of renderers and memory and time

consumption. Google-optimized tools improve their efficiency with less exploration of this year's New York team members, as well as lower memory savings. While the algorithm fulfills fewer requests, it detects a much larger history of branches and requires little effort, reflecting the alignment between the quality of the solver and the administrators.

5.2. Algorithmic Improvements:

The explanation of the differences between the original **Google OR Tools** code and the **improved version**:

While both versions effectively address the medical officer scheduling problem using constraint programming, the improved version offers several refinements:

1. Concise Variable Creation:

Combines variable declaration and model addition into a single step using dictionary comprehension, making the code more compact and potentially easier to read.

2. Streamlined Solver Integration:

Creates the solver directly within the model creation process, potentially enhancing efficiency.

3. Optimized Shift Counting Logic:

Employs more concise expressions to count shifts per medical officer, improving code readability and potential performance.

4. Enhanced Readability:

Incorporates minor formatting changes, such as consistent indentation and spacing, to promote better code comprehension and maintainability.

5. Potential Advantages:

Conciseness: The streamlined code can be easier to understand and modify.

Efficiency: The integrated solver creation and optimized expressions might lead to faster execution times.

Readability: The improved formatting enhances code clarity.

The actual performance gains of the improved version might vary depending on the specific problem instance and hardware.

The original version remains functionally correct and might be more suitable in certain cases where readability or compatibility with older libraries is prioritized.

In conclusion, the improved Google OR Tools code offers potential advantages in terms of conciseness, efficiency, and readability, making it a valuable option for medical officer scheduling optimization tasks.

6. DISCUSSIONS

6.1 Previous Studies:

Our research focuses on improving medical staff scheduling in healthcare organizations by comparing Google OR tools and genetic algorithms. Below is a comparison with previous studies based on the keywords I provided:

Medical Administrator Scheduling: Previous studies have focused on the role of medical administrative assistants in scheduling appointments, updating patient histories, and working with insurance¹. Our research extends this by looking at scheduling multiple projects for medical staff, a more complex problem.

Scheduling multiple projects with limited resources: Previous research has addressed the problem of scheduling multi-mode projects where resources are limited. Our research contributes to this field by applying it to the specific context of healthcare organizations and comparing the performance of Google OR tools and genetic algorithms.

Genetic Algorithms: Genetic algorithms are metaheuristic optimization methods inspired by natural selection and genetics, and are commonly used to generate high-quality solutions to optimization and search problems. Our research innovatively applies these algorithms to the medical staff scheduling problem and compares their performance with Google OR tools.

Google OR Tools: Although I could not find specific references to Google OR tools in the context of scheduling, these tools are widely used to solve various optimization problems.

Comparing algorithms: Comparing algorithms usually involves analyzing their efficiency in terms of time and space. Our research follows

this approach by comparing the performance of Google OR tools and genetic algorithms in different scenarios.

In conclusion, our research builds on previous studies in these areas and provides valuable insights into medical staff scheduling in healthcare organizations. The discovery that the modified Google OR algorithm significantly outperforms the Google OR tools and the regular genetic algorithm is a major contribution to the field.[15-24]

6.2 Results Discussions

The comparative analysis revealed intriguing insights into the strengths and weaknesses of both Google OR Tools and genetic algorithms in optimizing medical officer scheduling. Both methods achieved high levels of scheduling efficiency, consistently generating feasible and conflict-free schedules. However, their performance varied in other aspects:

Shift requests medical officer.: Google OR Tools and Improved OR Tools consistently fulfilled more medical officer shift requests compared to the genetic algorithm. This suggests that constraint programming excels in respecting individual preferences while optimizing the overall schedule.

Computational efficiency: Improved OR Tools demonstrated the fastest execution times, followed by Google OR Tools and then the genetic algorithm. This highlights the importance of optimizing constraint models and solver selection for improved efficiency.

Conflicting schedules: The genetic algorithm encountered a significantly higher number of conflicting schedules during its search. This indicates a trade-off between solution quality and computational efficiency, where exploring a broader search space might lead to more infeasible solutions initially.

Practical Implications: The findings of this research offer valuable practical implications for healthcare organizations seeking to optimize medical officer scheduling:

Google OR Tools and Improved OR Tools emerge as efficient and reliable options for scheduling with high adherence to medical officer preferences and efficient schedule generation. Organizations with resource

constraints and a priority on respecting medical officer requests may find these tools particularly beneficial.

The genetic algorithm, while achieving a lower success rate in meeting shift requests, offers an alternative approach for exploring a broader solution space and potentially discovering unforeseen optimal solutions. This could be valuable for organizations with highly complex scheduling requirements and flexibility in adjusting shift assignments.

The improved versions of both tools demonstrate the potential of algorithmic modifications for enhancing performance. Organizations can explore further customization of these tools or consider utilizing hybrid approaches that combine constraint programming with metaheuristics for even greater efficiency and solution quality.

Unexpected Results and Challenges: One unexpected result was the relatively low number of shift requests met by the genetic algorithm. While it found optimal solutions in terms of schedule efficiency, balancing individual preferences with overall optimization proved more challenging. Additionally, the high number of conflicting schedules encountered during its search highlights the need for further refinement of the algorithm for specific healthcare applications.

7. CONCLUSION

This research highlights the effectiveness of Google OR tools and genetic algorithms in improving resource-constrained multi-project scheduling for medical staff in healthcare settings. By comparing their performance and proposing algorithmic improvements, this study provides valuable insights to guide decision-making in healthcare organizations. Ultimately, choosing the most appropriate scheduling tool will depend on individual organizational needs, resources, and priorities. However, the results of this research provide a critical step toward improving medical staff scheduling practices, ultimately leading to improved patient care, increased medical staff satisfaction, and a more efficient health care system. The improved Google tools showed high speed and less memory consumption, while the regular Google tools took longer, and the genetic algorithm

consumed a lot of memory and needed a long time.

REFERENCES

1. Google Corporation, "Google Developers", <http://developers.google.com/>, January 9, 2024.
2. GeeksforGeeks, "Genetic Algorithms", <https://www.geeksforgeeks.org/genetic-algorithms/>, January 9, 2024.
3. Static, U., Jacko, P., & Kirkbride, C., "Performance evaluation of scheduling policies for the dynamic and stochastic resource-constrained multi-project scheduling problem", *International Journal of Production Research*, Vol. 60, Issue 4, Pages 1411-1423, 2022.
4. Browning, T. R., Yassine, A. A., "Resource-constrained multi-project scheduling: Priority rule performance revisited", *International Journal of Production Economics*, Vol. 126, Issue 2, Pages 212-228, 2010.
5. Fischer, F. M., Borges, F. N., Rotenberg, L., Latorre, M. R., Soares, N. S., Rosa, P. L., Teixeira, L. R., Nagai, R., Steluti, J., Landsbergis, P., "Workability of health care shift workers: What matters?", *Chronobiol Int.*, Vol. 23, Issue 6, Pages 1165-79, 2006.
6. Mahmud, F., "Evolutionary Algorithms for Resource Constrained Project Scheduling Problems", *Doctoral Thesis, UNSW University, Sydney*, 2023.
7. El-Abbasy, M. S. K., "Multi-objective multi-project construction scheduling optimization", *Doctoral Thesis, Concordia University, Montreal*, 2015.
8. Chen, R., Liang, C., Gu, D., Leung, J. Y., "A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development considering competency evolution", *International Journal of Production Research*, Vol. 55, Issue 21, Pages 6207-6234, 2017.
9. De Boer, R., "Resource-constrained multi-project management", *Doctoral Thesis, University of Twente, Netherlands*, 1998.
10. Kannimuthu, M., Raphael, B., Ekambaram, P., Kuppaswamy, A., "Comparing optimization modeling approaches for the multi-mode resource-constrained multi-project scheduling problem", *Engineering, Construction and Architectural Management*, Vol. 27, Issue 4, Pages 893-916, 2020.

11. Browning, T. R., & Yassine, A. A., “A random generator of resource-constrained multi-project network problems “, *Journal of Scheduling*, Vol. 13, Issue 1, Pages 143-161, 2010.
12. Badawiyeh, B. H., “The effect of planning and resource leveling on UAE contractors”, Doctoral Thesis “, The British University, Dubai, 2010.
13. Cadorin, D., Darwish, R., “Decision-making biases in project portfolio selection and prioritization: An exploratory study of the rationale behind decision making leading to project portfolio problems “, Master Thesis, Umea University, Sweden, 2015.
14. Zhou, Q., Li, J., Dong, R., Zhou, Q., & Yang, B., “Optimization of multi-execution modes and multi-resource-constrained offshore equipment project scheduling based on a hybrid genetic algorithm “, *Computer Modeling in Engineering & Sciences*, Vol. 134, Issue 2, Pages 1263-1281, 2023.
15. Workable, Medical Administrative Assistant job description. <https://www.workable.com>, January 9, 2024
16. IEEE Xplore, Multi-Mode Project Scheduling with Limited Resource and Budget Constraints, <https://ieeexplore.ieee.org>, January 9, 2024
17. Typeset. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach, <https://www.typeset.io>, January 9, 2024
18. Springer, Multi-project scheduling with two-stage decomposition, <https://www.springer.com>, January 9, 2024
19. Wikipedia, Genetic algorithm, <https://www.wikipedia.org>, January 9, 2024
20. GeeksforGeeks, Genetic Algorithms, <https://www.geeksforgeeks.org>, January 9, 2024
21. OpenDSA, Comparing Algorithms, <https://opensa.io>, January 9, 2024
22. Study Algorithms, How do you compare the two algorithms?, <https://www.studyalgorithms.com>, January 9, 2024
23. Baeldung, How to Compare Two Algorithms Empirically?, <https://www.baeldung.com>, January 9, 2024
24. Wikibooks, Problem Solving: Comparing algorithms, <https://www.wikibooks.org>, January 9, 2024