Iranian Journal of Numerical Analysis and Optimization
Vol. 16, No. 2, 2026, pp 590–606
https://doi.org/10.22067/ijnao.2026.95911.1749
https://ijnao.um.ac.ir/

Research Article

Design and implementation of a graph-coloring algorithm for optimizing flight-level allocation in air traffic management in Iran

S.A.A. Mosavi, A. Erfanian and N. Sabeghi*

Abstract

The rapid growth of air traffic demand highlights the necessity of efficient and reliable methods for air traffic flow management (ATFM). In Iran, current flight level allocation is predominantly performed manually by human operators, which is prone to errors, lacks scalability, and does not guarantee optimal use of available airspace resources. To address this limitation, this study proposes a novel optimization framework based on graph coloring techniques for the allocation of flight levels.

The airspace is modeled as a graph, where each flow corresponds to a node and potential conflicts are represented as edges. The problem is then formulated as an optimization model

*Corresponding author

Received 11 October 2025; revised 27 December 2025; accepted 5 January 2026

Seyed Ali Asghar Mosavi
Department of Pure Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran. e-mail: saam9473@gmail.com

Ahmad Erfania
Department of Pure Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran. e-mail: erfanian@um.ac.ir

Narjes Sabeghi
Department of Mathematics, Faculty of Basic Sciences, Velayat University, Iranshahr, Iran. e-mail: n.sabeghi@velayat.ac.ir

How to cite this article
Mosavi, S.A.A., Erfanian, A. and Sabeghi, N., Design and implementation of a graph-coloring algorithm for optimizing flight-level allocation in air traffic management in Iran. *Iran. J. Numer. Anal. Optim.*, 2026; 16(2): 590-606. https://doi.org/10.22067/ijnao.2026.95911.1749

with the goal of minimizing the number of distinct flight levels while ensuring safety constraints. A hybrid algorithm is developed that combines the DSatur heuristic for generating an initial solution with a constraint programming model enhanced by maximal clique detection for refinement and optimization.

The approach is applied to real operational data from Tehran's Mehrabad and Mashhad Hasheminejhad Airports during peak hours. Experimental results demonstrate that the proposed method reduces the number of required flight levels compared to DSatur alone, ensuring both improved airspace efficiency and conflict-free operations.

AMS subject classifications (2020): Primary 05C90, 90C59; Secondary 05C15.

Keywords: Air traffic flow management (ATFM); Graph coloring techniques; DSatur heuristic; Constraint programming (CP).

1 Introduction

With the rapid expansion of cultural, commercial, and tourism interactions at both the national and international levels, the air transport industry, as one of the most important connectivity infrastructures, plays a vital role in facilitating fast and safe movement. The dramatic reduction in travel time compared with other modes of transportation, and the ability to connect cities, countries, and even continents, have further increased the importance of this industry. The continuous rise in demand for air travel makes precise planning and the use of scientific solutions unavoidable in order to maintain safety, optimize resource utilization, and reduce operational risks.

At present, air traffic management in Iran relies heavily on human experience and decision-making. This dependence, while creating limits on the scalability of operations, can increase the likelihood of errors. The lack of intelligent systems based on mathematical models has made the need to develop novel methods, capable of automatically performing optimal allocation of flight resources with minimal human intervention, even more urgent.

One effective approach in this area is the use of graph modeling and combinatorial optimization algorithms, particularly graph-coloring algorithms. In this modeling, each route or section of airspace is represented as a vertex (node), and conflicts or concurrency constraints between flights are represented as edges. The problem of assigning flight levels and scheduling flights can be viewed as equivalent to the graph-coloring problem, in which colors denote safe altitudes (flight levels) or permissible time slots for flights. The objective is to minimize the number of colors (flight resources) while satisfying all safety and operational constraints.

To achieve this goal, real data obtained from radar control stations are used as system inputs. These data are prepared for modeling through the following steps:

1. Defining the problem domain and collecting raw data,
2. Preprocessing and cleaning the data,
3. Extracting patterns and mathematical relationships from the data,
4. Evaluating and applying the extracted patterns in the proposed system.

In this study, by employing a multi-objective mathematical programming model, an algorithm for graph coloring is presented that can simultaneously eliminate flight conflicts, ensure optimal utilization of resources, and enhance flight safety within the country's airspace. The structure of this paper is as follows: Section 2 reviews related research on the application of graph coloring in air traffic management. Section 3 introduces the dataset used, along with preprocessing steps and the construction of the flight graph model. Section 4 presents the proposed algorithm and details of its implementation. Section 5 discusses experimental results and performance analysis of the proposed approach in real-world scenarios. Finally, Section 6 provides conclusions and suggestions for future research.

2 Literature review

Graph coloring, as a mathematical modeling tool, has found remarkable applications in recent years in the field of Air traffic flow management (ATFM). In this section, two key studies in this area are reviewed.

Barnier and Brisset [1] proposed an innovative model for ATFM in France, based on the design of direct route networks and the static allocation of flight levels to intersecting routes. By reformulating the problem as a graph-coloring problem and leveraging a combination of constraint programming (CP) methods and greedy algorithms, the model aimed to completely eliminate flight conflicts. The key novelty of this study lies in the use of a greedy algorithm to identify large cliques in the intersection graph and apply distinct global constraints to them. Moreover, by designing a clique-based search strategy and breaking color symmetry, the model was able to obtain optimal solutions in a short time for most real-world cases. Findings showed that for flows with more than five flights, a maximum of 12 flight levels is required, and the approach outperforms the DSatur (Degree of saturation) algorithm in large instances. In addition, the model can be adapted to the current structure of European airspace (altitudes from 20,000 to

40,000 feet) and potentially scaled up to the European level. However, the authors emphasized that further development and consideration of additional operational constraints are necessary for practical application.

In [4], the problem of flight level allocation was modeled as a factorized optimization problem that simultaneously addresses two conflicting objectives: Reducing fuel consumption (airlines' interest) and maintaining air traffic safety (public interest). In this study, using threshold-based spectral graph coloring, each flight path was represented as a vertex and each path intersection as an edge, while colors denoted flight levels (FL10 to FL400). The interference matrix defined the safety cost based on the distance between flight levels, such that separations below the safety threshold imposed infinite interference, while larger separations incurred decreasing interference values. The fuel consumption model, defined for Airbus A320 data, included both the cost of climbing to the optimal altitude and the cruise cost as a function of distance and specific range. To solve the problem, a hybrid approach combining Simulated annealing (SA) and Hill Climbing (HC) algorithms was applied: SA allowed controlled acceptance of worse solutions to escape local optima, while HC accepted only better solutions. Results indicate that enforcing safety constraints leads to a 5% increase in fuel consumption but significantly improves safety separation between flights. The authors suggested that in the future, the model could be extended to real-time operations with dynamic positioning for managing both aircraft and UAV traffic, as well as applied to areas such as route-sharing optimization in urban transportation.

Taken together, these two studies demonstrate that graph-coloring-based approaches, particularly those using advanced or hybrid algorithms, can serve as a foundation for the design of modern air traffic management systems, while also highlighting the need for adaptability to each country's operational constraints and conditions.

A review of prior studies shows that although graph-coloring approaches, whether in the form of static models focused on eliminating flight conflicts or multiobjective models balancing safety and fuel efficiency, have yielded valuable results, several fundamental limitations remain:

1. Dependence on European airspace data and structure: Existing models are largely based on European flight data and route structures (notably France and Spain), which limits their direct applicability in other countries due to differences in route design, operational restrictions, and flight regulations.

2. Incomplete alignment with Iran's operational conditions: Iran's airspace, due to its geopolitical location and the high volume of international overflights, has features not considered in prior models, including differences in overflight density, temporal flight patterns, and specific security considerations.

3. Limited scalability under variable traffic conditions: Many existing approaches either function statically or lose efficiency in high-traffic scenarios. Iran's air traffic management requires a system capable of maintaining optimal flight-level allocation even under dynamic, real-time traffic fluctuations.

4. Incomplete utilization of real-time data: Although some studies mention real-time extensions, few operational models have been developed that can be deployed in control centers using live data with continuously updated flight graphs.

In light of these gaps, the present study aims to design and implement a multi-objective graph-coloring algorithm tailored to the characteristics and constraints of Iranian airspace, capable of operating under dynamic traffic conditions with real-time data. By using real radar control station data, this approach seeks not only to ensure flight safety but also to maximize the efficiency of air routes, serving as a foundation for the development of intelligent air traffic management systems in Iran.

3 Research methodology

In this study, the ATFM problem is modeled as a *graph coloring* problem. In this framework, the airspace or flight network is represented as a graph $G = (V, E)$, which is defined as follows:

- Each vertex $v \in V$ represents a flight, an airport, a way point, or a specific sector of airspace.

- Each edge $e \in E$ denotes a potential conflict between two vertices; for example, two flights entering the same airspace sector at the same time or altitude.

In the coloring process, each vertex is assigned a color corresponding to an operational resource (e.g., flight level, time slot, or route), such that no two adjacent vertices (conflicting flights) share the same color.

The application of graph coloring to ATFM leads to the optimal allocation of resources, preventing congestion and ensuring safety. Three key types of allocation problems can be identified:

1. **Time Slot Assignment**: conflicting flights are assigned to different time windows to avoid runway or sector congestion.

2. **Flight Level Assignment**: intersecting routes are assigned to different flight levels to maintain the minimum vertical separation.

3. **Route Assignment**: alternative routes are selected to reduce interference in high-density sectors.

3.1 Illustrative example

Example 1. Suppose that ten flights must pass through a congested sector within one hour, while the sector's capacity is only three flights every 15 minutes.

- Vertices: flights
- Edges: pairs of flights that are in the sector within the same 15-minute interval
- Colors: time slots (e.g., Color A: 0–15 minutes, Color B: 16–30 minutes, etc.)

By solving the graph-coloring problem, flights are assigned to different time slots such that no conflicting flights share the same color, thus reducing congestion.

3.2 Computational challenges

Several computational challenges are inherent to this problem:

- Due to its NP-Hard nature, heuristic and metaheuristic algorithms are required for large-scale networks.
- Unexpected changes caused by weather conditions or delays demand real-time recoloring (dynamic conditions).
- Simultaneous consideration of time, altitude, and route increases problem complexity (multi-resource constraints).

Air traffic control centers are responsible for ensuring safe aircraft separation. Ideally, all flights would travel on direct routes. However, if all flights used direct horizontal routes only, human control would become infeasible due to excessive intersections. A practical solution is to apply *vertical separation* (flight levels) to routes with horizontal intersections: Flights with the same origin-destination pair share the same direct path but use different flight levels to avoid conflicts. Thus, only flights with overlapping schedules require vertical separation.

Inspired by [1], this study adopts the concept of *flows*, where all flights with the same origin-destination are grouped into a single flow. The size of each flow (n_i) equals the number of aircraft within that flow. For each flow, the start and end times (based on the first and last flight) are calculated. Hence, for a given day, the model input is defined by the set of flows.

If the problem size becomes large, to control computational complexity, then only flows with more than a minimum threshold $n_{\min}$ can be considered. For instance, commercial scheduled flights usually form large flows that require higher flight levels (to save fuel), whereas private flights (Instrument flight rule (IFR)) typically form small flows operating at lower levels with fewer conflicts. Therefore, the model can be solved only for large flows (e.g., more than 5–10 flights), while small flows can be assigned by default below the main flight levels. In this research, however, we consider all flows ($n_{\min} = 1$).

If two flows intersect spatially and their time intervals overlap, then they cannot be assigned to the same flight level. Formally

$$\text{for all } i, j \quad (1 \le i < j), \quad L_i \neq L_j \quad \text{if} \quad \text{intersect}(r_i, r_j) \wedge \text{overlap}(t_i, t_j). \tag{1}$$

3.3 Experimental basis

The experimental basis of this research consists of real-world data from 52 flight flows at Tehran Mehrabad International Airport and Mashhad Hasheminejhad International Airport during weekly peak hours (Table 1). These data include detailed information on entry and exit times in controlled sectors, the size of each flow, and traversed routes, obtained from the national air traffic control center.

3.4 Problem modeling of ATFM with graphs

In this study, ATFM during the peak traffic period at Tehran Mehrabad and Mashhad Shahid Hasheminejhad Airports is formulated as a graph coloring problem. The considered airspace is divided into 11 flight corridors, each comprising several flows (flights with specific origin/destination pairs). Each vertex of the graph represents a flow, while each edge denotes the existence of a temporal or route conflict between two flows within the same corridor. Each corridor is thus modeled as an independent clique (a complete subgraph), since the flows within a corridor

Table 1: Flight flows at Tehran Mehrabad and Mashhad Shahi Hasheminejhad International Airport during peak hours.

Start	End	Corridor	Origin–Destination	n_i	Flow
0500	0700	1	Tehran–Bandar Abbas	5	f_1
0500	0700	1	Tehran–Zahedan	3	f_2
0500	0700	1	Tehran–Zabol	1	f_3
0500	0700	1	Tehran–Iranshahr	1	f_4
0500	0700	1	Tehran–Yazd	1	f_5
0500	0700	1	Tehran–Qeshm	2	f_6
0500	0700	1	Tehran–Kerman	2	f_7
0500	0700	1	Tehran–Chabahar	2	f_8
0500	0700	1	Tehran–Sirjan	1	f_9
0500	0700	2	Tehran–Asaluyeh	6	f_{10}
0500	0700	2	Tehran–Shiraz	5	f_{11}
0500	0700	2	Tehran–Kish	5	f_{12}
0500	0700	2	Tehran–Isfahan	2	f_{13}
0500	0700	2	Tehran–Bushehr	3	f_{14}
0500	0700	2	Tehran–Dubai	1	f_{15}
0500	0700	3	Tehran–Ahvaz	2	f_{16}
0500	0700	3	Tehran–Abadan	1	f_{17}
0500	0700	3	Tehran–Muscat	1	f_{18}
0500	0700	4	Tehran–Tabriz	5	f_{19}
0500	0700	4	Tehran–Urmia	1	f_{20}
0500	0700	5	Tehran–Ardabil	2	f_{21}
0500	0700	5	Tehran–Rasht	1	f_{22}
0500	0700	6	Tehran–Mashhad	11	f_{23}
0500	0700	7	Tehran–Kermanshah	2	f_{24}
0500	0700	7	Tehran–Ilam	1	f_{25}
0500	0700	7	Tehran–Najaf	3	f_{26}
0500	0700	7	Tehran–Baghdad	2	f_{27}
0500	0700	8	Mashhad–Ahvaz	1	f_{28}
0500	0700	8	Mashhad–Abadan	1	f_{29}
0500	0700	8	Mashhad–Yazd	1	f_{30}
0500	0700	8	Mashhad–Shiraz	1	f_{31}
0500	0700	8	Mashhad–Bushehr	1	f_{32}
0500	0700	8	Mashhad–Isfahan	1	f_{33}
0500	0700	9	Mashhad–Bandar Abbas	2	f_{34}
0500	0700	9	Mashhad–Kerman	1	f_{35}
0500	0700	9	Mashhad–Kish	2	f_{36}
0500	0700	9	Mashhad–Asaluyeh	1	f_{37}
0500	0700	9	Mashhad–Qeshm	1	f_{38}
0500	0700	9	Mashhad–Dubai	2	f_{39}
0500	0700	10	Mashhad–Zahedan	1	f_{40}
0500	0700	10	Mashhad–Zabol	1	f_{41}
0500	0700	10	Mashhad–Chabahar	2	f_{42}
0500	0700	10	Mashhad–Iranshahr	1	f_{43}
0500	0700	11	Mashhad–Ardabil	1	f_{44}
0500	0700	11	Mashhad–Urmia	1	f_{45}
0500	0700	11	Mashhad–Tabriz	2	f_{46}
0500	0700	11	Mashhad–Kermanshah	1	f_{47}
0500	0700	11	Mashhad–Ilam	1	f_{48}
0500	0700	11	Mashhad–Rasht	2	f_{49}
0500	0700	11	Mashhad–Tehran	6	f_{50}
0500	0700	11	Mashhad–Baghdad	2	f_{51}
0500	0700	11	Mashhad–Najaf	3	f_{52}

are mutually conflicting, while flows belonging to different corridors do not interfere with one another. Figure 1 shows the graph constructed from the data presented in Table 1.

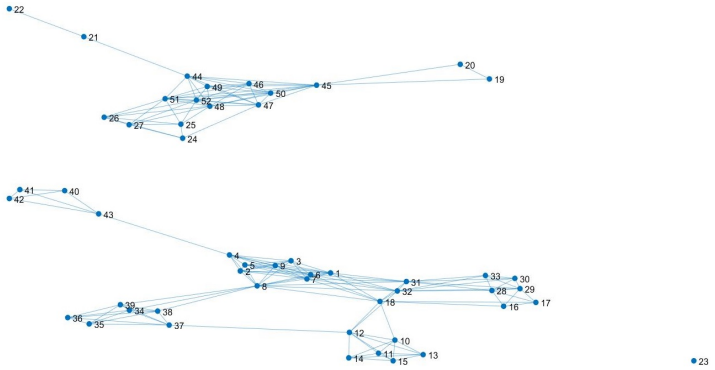

Figure 1: Graph corresponding to the data in Table 1.

The initial model, based on relation (1), includes only local constraints between each pair of flows. However, in graphs resulting from intersecting routes, large complete subgraphs (cliques) are frequently observed. In such cases, instead of formulating multiple pairwise constraints, one can impose an all-different constraint over the clique members, which is both simpler and more effective in detecting inconsistencies. Since finding the maximum clique or all maximal cliques is an NP-hard problem, greedy or approximate algorithms are typically employed. The algorithm adopted in this work (inspired by Bomze et al. [2]) starts from each node and incrementally builds a clique by selecting, at each step, the node with the largest neighborhood intersection. This procedure generates only maximal cliques, after which duplicate cliques or trivial ones (of size 2) are eliminated. The theoretical complexity of the algorithm is $O(n^4)$ in the worst case (for complete graphs). Nevertheless, in larger real-world instances, the graphs are sparse and the effective complexity growth is significantly lower [2].

4 Proposed algorithm

The proposed algorithm, employing a combination of greedy strategies for initial assignment and CP for final optimization, is designed to achieve resource allocation at an operational scale using real-world data.

4.1 Initial solution generation

In this paper, the DSatur algorithm is employed to assign flight levels to the flights listed in Table 1. DSatur is a graph coloring algorithm introduced by Daniel Brélaz in 1979 [3]. Similar to greedy coloring algorithms, DSatur colors the vertices of a graph one by one, adding a new color whenever necessary. After coloring a new vertex, the algorithm identifies the uncolored vertex with the largest number of distinct colors already assigned to its neighbors, and this vertex is selected for the next step. Brélaz [3] defined this number as the saturation degree of a vertex, from which the algorithm derives its name.

Although DSatur is a heuristic graph coloring algorithm, it provides exact results for bipartite, cyclic, and wheel graphs [3]. It is also known in the literature as LF saturation [5].

To generate a quick initial solution, the algorithm proceeds as follows:

1. Select the vertex with the highest saturation degree (the largest number of distinct colors among its neighbors).
2. In case of a tie, choose the vertex with the highest degree (the most incident edges).
3. Assign the smallest feasible color not used by its neighbors.
4. Update the saturation degrees of the uncolored neighbors.
5. Repeat until the graph is fully colored.

4.2 CP Model

In the proposed approach, the output of DSatur is used as an upper bound for the number of colors in a CP model. Then

- Large cliques in the graph are identified.
- Global coloring constraints are imposed on these cliques.
- The objective is to minimize the total number of colors (flight levels).

Algorithm 1 Clique-based graph coloring with all different constraints

Require: F (set of flows),
For each flow $i \in F$: route r_i, time window t_i,
$n_{\min}$ (minimum flow size to enter the model),
$LEVELS = \{1, \ldots, N\}$ (initial domain of colors (flight levels)).
Ensure: Mapping $L : F \to \mathbb{N}$ minimizing number of distinct colors.

$F \leftarrow \{i \in F \mid n_i \geq n_{\min}\}$ (remove small flows).
$N \leftarrow |F|$ (upper bound on colors).
$V \leftarrow F$ (one vertex per flow).
for All $i, j \in F$, $i < j$ **do**
  if intersect(r_i, r_j) **and** overlap(t_i, t_j) **then**
    add edge (i, j) to E
  end if
end for
Detect maximal cliques by Greedy algorithm ($\mathcal{C}$).
Remove from $\mathcal{C}$ any clique with $|C| < 3$ or duplicates.
$k_{\max} \leftarrow \max_{C \in \mathcal{C}} |C|$ (lower bound).
for All $i \in V$ **do**
  $Domain(i) \leftarrow \{1, \ldots, N\}$.
end for
for All $C \in \mathcal{C}$ **do**
  impose AllDifferent$(\{L_i : i \in C\})$.
end for
for All $(i, j) \in E$ not covered by a clique constraint **do**
  impose $L_i \neq L_j$.
end for
Sort the cliques in descending order of size.
if there is a remaining clique **then**
  A variable from that clique is selected, otherwise a variable with the smallest domain is selected.
  if No colors in the variable's domain have been used previously **then**
    the smallest unused color is assigned deterministically, otherwise the algorithm proceeds with branching.
  end if
end if

4.3 Solution procedure

Considering the explanations in the previous sections, the steps for solving the problem are as follows:

1. Identify flows and corridors from the real-world data (Table 1).

2. Construct the graph based on the interference matrix.

3. Detect cliques (each corridor forms an independent clique).

4. Execute DSatur to obtain an initial estimate of the number of colors.

5. Apply CP combined with clique constraints to optimize and reduce the number of colors.

6. Assign the final colors (Table 2).

7. Validate the results against operational constraints.

The "Clique-based graph coloring with all different constraints" algorithm was implemented in MATLAB 2020 and applied to the data presented in Table 1. The corresponding output is shown in Table 2. It can be observed that, for the flows in Table 1, five flight levels are sufficient to ensure vertical separation and maintain flight safety during the peak hour under consideration.

Table 2: Flow color assignment after executing the algorithm

Corridor	**Flow**	**Flow Color**	**Corridor**	**Flow**	**Flow Color**
1	f_1	1	7	f_{27}	4
1	f_2	2	8	f_{28}	3
1	f_3	3	8	f_{29}	4
1	f_4	4	8	f_{30}	1
1	f_5	5	8	f_{31}	5
1	f_6	6	8	f_{32}	9
1	f_7	7	8	f_{33}	2
1	f_8	8	9	f_{34}	3
1	f_9	9	9	f_{35}	1
2	f_{10}	1	9	f_{36}	2
2	f_{11}	2	9	f_{37}	4
2	f_{12}	3	9	f_{38}	5
2	f_{13}	4	9	f_{39}	6
2	f_{14}	5	10	f_{40}	1
2	f_{15}	6	10	f_{41}	2
3	f_{16}	1	10	f_{42}	3
3	f_{17}	2	10	f_{43}	5
3	f_{18}	4	11	f_{44}	2
4	f_{19}	1	11	f_{45}	3
4	f_{20}	2	11	f_{46}	1
5	f_{21}	1	11	f_{47}	4
5	f_{22}	2	11	f_{48}	5
6	f_{23}	1	11	f_{49}	6
7	f_{24}	1	11	f_{50}	7
7	f_{25}	2	11	f_{51}	8
7	f_{26}	3	11	f_{52}	9

4.4 Results and analysis

To evaluate the performance of the proposed algorithm compared to DSatur, a simple flight network scenario was designed as Example 2.

Example 2. Consider a scenario with eight flight routes that intersect at several points. Each route is modeled as a vertex in the graph, and each intersection between two routes is represented as an edge. In total, this network includes eight flight routes and 24 intersections. The corresponding adjacency matrix is given by

$$A = \begin{bmatrix} 0\,1\,0\,1\,0\,1\,0\,0 \\ 1\,0\,0\,0\,1\,1\,1\,0 \\ 0\,0\,0\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,0\,1 \\ 0\,1\,0\,1\,0\,1\,0\,0 \\ 1\,1\,0\,0\,1\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0\,1\,1\,0 \end{bmatrix}.$$

The graph corresponding to this hypothetical network is illustrated in Figure 2.

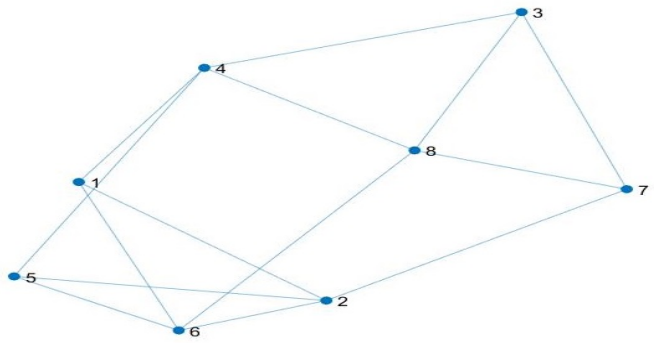

Figure 2: Graph corresponding to the hypothetical flight network in Example 2.

The objective is to assign the minimum number of distinct flight levels (colors) to the routes, such that no two intersecting routes share the same flight level.

Our proposed algorithm (based on complete search techniques combined with maximal clique guidance) is able to solve the problem optimally using only three flight levels. In other words, three flight levels are sufficient to eliminate all conflicts among the routes.

In contrast, applying the DSatur algorithm to the same dataset—although significantly faster—resulted in the use of four flight levels. This implies that DSatur, in this case, reached a suboptimal solution by introducing an unnecessary additional flight level into the network.

This example clearly demonstrates that in real-world applications such as air traffic network management, relying solely on greedy algorithms like DSatur can lead to resource waste (flight levels). In contrast, our proposed algorithm, by guaranteeing optimality, is able to provide a more efficient allocation of flight levels.

The analysis of results shows that the proposed CP + Clique approach achieved a reduction in the number of required flight levels compared to DSatur. This reduction can lead to

- more efficient utilization of airspace capacity,
- reduction of delays,
- fuel savings and lower operational costs,
- improved flight safety.

Moreover, modeling each corridor as a complete graph (due to the absence of inter-corridor conflicts) simplified the model and increased the accuracy of the coloring allocation.

5 Conclusion and recommendations

To the best of our knowledge, this research represents the first scientific study in Iran that directly employs real operational data from two major Airports (Mehrabad and Hasheminejhad) during peak traffic hours to model and solve the ATFM problem using graph coloring algorithms.

Based on the collected data, it was observed that, at present, route allocation for maintaining flight safety and handling high-demand flights within the same time window is performed manually. This manual approach has limitations; for example, the Tehran–Mashhad and Mashhad–Tehran flights are treated differently, and similar cases exist that require manual adjustments.

The proposed methods in this study can overcome such limitations and enable more efficient utilization of alternative routes. The released capacity can then be used for other purposes such as humanitarian services, military operations, political programs, and other special applications.

Strengths

- Utilization of real-world data,
- Potential generalization to other major and secondary airports in the country,
- Optimal combination of greedy algorithms with constraint-based optimization methods.

Weaknesses

- Limitations due to security and military restrictions in accessing full airspace data,
- Lack of weather simulation and dynamic flight conditions in the current model.

Recommendations

- Development of an intelligent ATFM software tool for real-time allocation of flight levels,
- Collaboration with security authorities to optimize restricted airspace zones,
- Incorporating the effects of weather conditions and dynamic modeling.
- Considering the constraints on the number of available flight levels, future research should focus on developing methods to reduce the required number of levels through strategies such as

    1. Flight time adjustment (rescheduling, delaying, or advancing),
    2. Route modification (re-routing),
    3. Increasing resources/capacity (e.g., adding runways or allowing higher concurrency subject to safety separation),
    4. Vertex splitting or cost-based decomposition (e.g., sequestering a service or dividing a flight into smaller time slots).

Figure 3 presents the potential future research directions for optimizing flight level allocation.

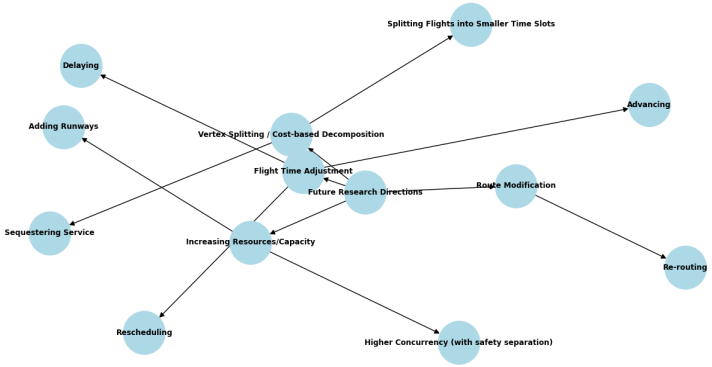

Figure 3: Suggested future research directions for optimizing flight level allocation

Acknowledgements

The research of the second author, Ahmad Erfanian, was supported in part by the Ferdowsi University of Mashhad. The research of the third author, Narjes Sabeghi, was supported in part by the Velayat University.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Barnier, N., and Brisset, P. *Graph coloring for air traffic flow management*, Ann. Oper. Res., 130(1) (2004), 163–178.

[2] Bomze, I.M., Budinich, M., Pardalos, P.M., and Pelillo, M. *The maximum clique problem*, In Handbook of Combinatorial Optimization: Supplement Volume A (pp. 1–74). Boston, MA: Springer US, 1999.

[3] Brélaz, D. *New methods to color the vertices of a graph*, Commun. ACM., 22(4) (1979), 251–256.

[4] Gimenez-Guzman, J.M., Martínez-Moraian, A., Reyes-Bardales, R.D., Orden, D., and Marsa-Maestre, I. *Flight level assignment using graph coloring*, Appl. Sci. 10(18) (2020), 6157.

[5] Yekezare, N., Zohrehbandian, M., Maghasedi, M., and Bonomo-Braberman, F. *Optimality of DSatur algorithm on chordal graphs*, Oper. Res. Lett., 57 (2024), 107185.