International Journal of Engineering, Science & Mathematics
Vol. 11 Issue 08, August 2022,
ISSN: 2320-0294 Impact Factor: 6.765
Journal Homepage:www.ijesm.co.in, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Aircraft Scheduling at Runway Based on Distributed Constraint Satisfaction

Ranjan Kumar Thakur
Assistant Professor, Department of Mathematics, H. P. S. College,
Madhepur, Madhubani – 847408,L. N. M. U., Darbhanga
Ranjan.ku.thakur@gmail.com

Nawin Kumar Agrawal
Professor, Department of Mathematics, L.N.M.U., Darbhanga, India
drnkumaragrawal@gmail.com

ABSTRACT
Runways are the scarce resource at airports. To utilize this resource optimally is essential for any traffic control decision at airports. Different methods have been used to solve this scheduling problem. In this paper a distributed constraint satisfaction problem (DCSP) approach is used to schedule departure sequence of aircrafts at runway. First the problem is formulated as a DCSP and then a solution strategy based on asynchronous backtracking is proposed.
Keywords–DCSP, Aircraft Scheduling.

Introduction

The management of traffic at airports is very complex in nature. Aircraft departure sequence is controlled by three controllers namely "preflight", "taxiways" and "runway" controllers. Management of departure sequence at airports consists of assignment of startup time to flights for preflight phase at gates, assignment of taxiway to flights for taxiing it to runways and lastly assigning takeoff time to flights at the assigned runways. Each controller is responsible for the scheduling in its own area. The scheduling plan of one controller assist the scheduling plan of other controllers. Controlling runways is last in the scheduling plan but runways are the scarcest resources to deal with at airports as it is very demanding to add any new runway at airports. Therefore, scheduling flights at runway is vital for any traffic control decision. So, to utilize runways optimally it is better that the schedule of runway controller will assists its previous controllers.

There are two major concerns while scheduling aircraft at runways. First is, wake vortex (the air turbulence) separation, it is the separation time that is needed between two consecutive flights taking off from the same runway. Flights are categorized based on its weight and speed class and certain separation time needed to be followed between two consecutive flights based upon their weight and speed class when they are to use same runway. Another concern is the restriction set by Central Flow Management Unit (CFMU). The CFMU assign about 15 minutes of time intervals to each flight during which the flight should takeoff. The coordination made by CFMU ensures constant traffic flows by restricting number of flight takeoff from airports.

Problem Statement:

The runway scheduling problem has three major scheduling tasks. A flight that is going to takeoff from an airport requires to use a runway then in its initial climbing phase it takes a Standard Instrument Departure (SID) route, then it exits from the airport using the specified Terminal Maneuvering Area (TMA) exit point (see: [1],[2]). A schematic topology of Prague airport is as below in figure 1.

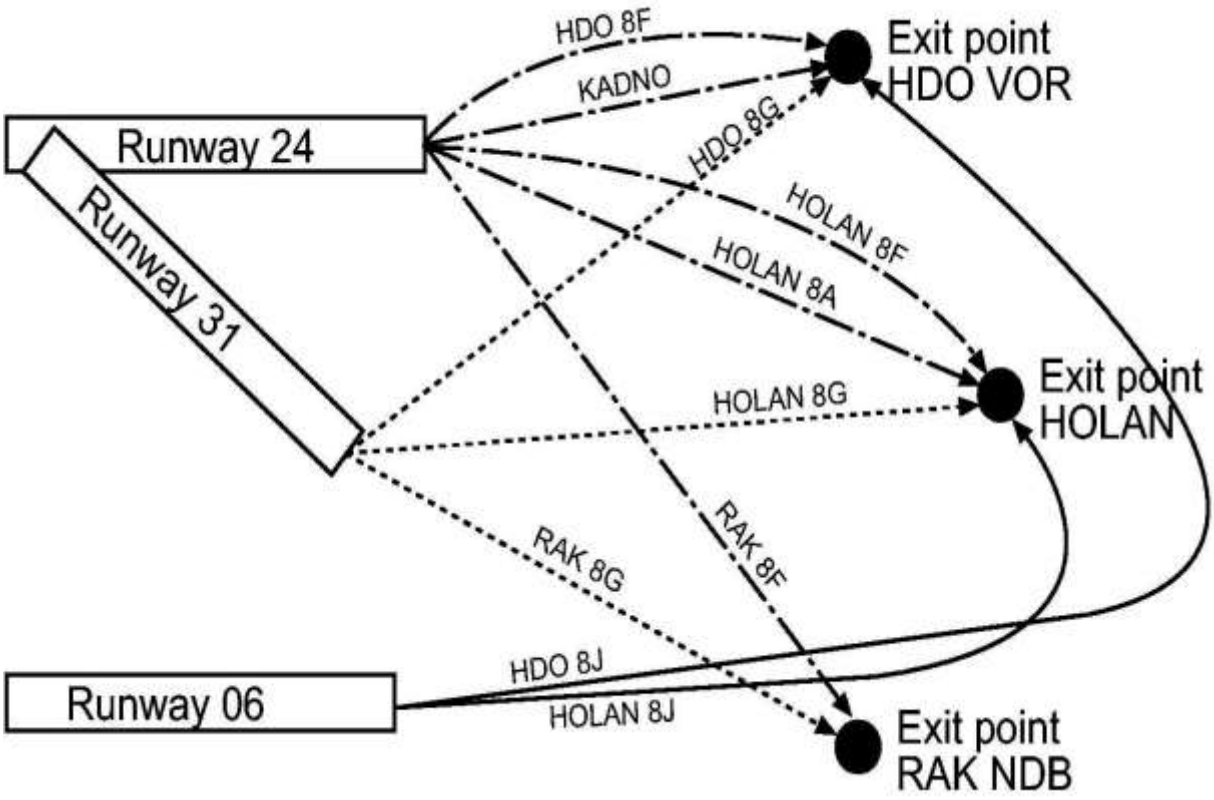

Figure 1 Schematic representation of part of Prague airport: runways, SID routes and exit points.

For an airport, the runway scheduling problem consists of assigning

- Takeoff time to flights
- Runway to flights
- SID routes to flights
- TMA exit point

While assigning the above to flights we need to consider the following constraints:

- Takeoff time must be in the specified CFMU time interval for the flights
- Minimum separation time must be followed between two consecutive flights depending upon their weight and speed class if they use the same runway
- While assigning runway SID routes and TMA exit points to flights, topology of airport must be followed
- Other tower control constraints that provide specific order for flight departure.

Related Work

The nature of the problem is constraint based and constraint satisfaction problem-based approach is well suited for our problem (see [5],[6]). Leeuwen (see [3]) uses constraint satisfaction problem-based approach and ILOG solver to solve the problem. Thakur (see [4]) also uses constraint satisfaction problem – based approach and C++ to solve the problem. In both the cases, firstly the problem is formulated as a Constraint Satisfaction Problem and backtracking is used to solve it.The changes made to the variables are in a sequence, means to say that there is one control over the whole processing. The search for

the solution may be reduced if the processing is distributed among automated agents (see, [8],[11]). Such a distributed approach is done in reducing "A truth maintenance system "(see, [9]) in a paper titled "Multiagent truth maintenance" by Hunus (see, [10]).
Reducing our problem into distributed constraint satisfaction problem and then solving it using asynchronous backtracking [see [7]] will reduce the time to get to the solution as compared to CSP based approach. In this paper we, first model the problem as a DCSP then provide a solution strategy based on asynchronous backtracking.

Proposed Model

In order to get a departure sequence of flights, we require to reduce our problem as a Distributed Constraint Satisfaction Problem (DCSP).

A Constraint Satisfaction Problem (CSP) is defined as a triplet (V,D,C) where V represents the set of variables, D represents the set of corresponding domains of the variables and C represents the set of constraints that needed to be fulfilled for a feasible solution. Several problems can be reduced to CSP and then can be solved using backtracking and heuristics. For examples n-queens problem, map coloring problem, different scheduling problems etc.

A DCSP is a CSP consists of automated agents where the variables and constraints are distributed among these automated agents. These agents communicate with each other by sending messages. We also considered that there is a finite delay in message delivery. Some protocols are needed before going for a solution which ensures that the DCSP can be solved. To solve the DCSP we need to establish the following protocols:

- A topology of agent's communication channel is created indicating the directed link between agents where two agents are linked if they needed to communicate with each other. The direction of edge can be established on the basis certain priority like alphabetical order to the names of agents etc.
- Agents' instantiation is to be done concurrently. After their instantiation, agents send messages to the agent which is connected with it by outgoing link. Then they wait for messages where waiting time is finite and respond to messages. Here the responding agent is the one which is connected by outgoing link and termed as constraint evaluating agent.
- The constraint evaluating agents get values from agents through incoming links. These values constitute agent view of the constraint evaluating agent – agent_view{(agentX, agentX value),(agentY, agentY value),…}. Here agentX, agentY, … are the agents through which the constraint evaluating agent is connected via incoming links.
- Agents communicate with each other with two types of messages: an ok message in the form - (ok?, (agent, agents value)) sent by the agent to another constraint evaluating agent to which it is linked. Another type of message is a nogood message in the form – (nogood, (agentX, agentX value), (agentY, agentY value),…) sent by the constraint evaluating agent to the value sending agents in its link of least priority.
- If the constraint evaluating agents find that it is does not have any value that is compatible with its agent_view then it send a nogood message to the agent to which it is connected via incoming link with least priority attaching its agent_view with the nogood message - (nogood, (agentX, agentX value), (agentY, agentY value),…).

Compatibility

- A nogood message is received by an agent, checks for any compatible value in its domain that is consistent with its agent_view.
- If a subset of agent_view be such that the agent is unable to find any value that is compatible with this subset, then such a subset is termed as nogood.
- If an agent finds a nogood, then one of the associated agent must change its value. So, agent initiate a backtrack and send nogood to one of the associated agents which has least priority.

Avoiding loops

- We can avoid loops by using a total order relation among agents.

Handling asynchronous changes

- It can be delt by introducing context attachment with the nogood message indicating the cause of nogood that triggered it.
- A nogood may be considered as a new constraint and can be used to create links dynamically.

A detailed formalization and solution strategy for DCSP is mentioned in an article by Yokoo (see [7]).

Stating the Runway Scheduling problem as a DCSP:

Each flight is considered as an automated agent. If there are "n" flight is to be scheduled at an airport then there are "n" agents. So, flights F1, F2, … are the agents. We create Flight class with the following details:

Flight Class:

String flightNUMBER	Contains detail of flight number
String runway	Contain assigned runway
String SID	Contain assigned SID
String TMA	Contain assigned TMA
String speedCLASS	Contain speed class of the flight
String weightCLASS	Contain weight class of the flight
Int cfmuSTART	Contains start time of allotted CFMU time interval
Int cfmuEND	Contains end time of allotted CFMU time interval
Int takeoff	Contains assigned takeoff time

Constraints:

consider an airport has "r" number of flights, "s" number of SID routes and "t" number of TMA exit points with associated topology of connection between runways and SID routes and between SID routes and TMA exit points.

- Resource constraint: It represents the possible assignments of runways, SID routes and TMA exit points to flights. For example, consider flight F1 can be assigned m_1 runways say runway(1), runway(2), …, runway(m_1), s_1 SID routes say SID(1), SID(2), …,SID(s_1) and t_1 TMA exit points say TMA(1), TMA(2), …, TMA(t_1). Then any assignment to F1 must be from these available option for F1. This we termed as resource constraint for F1. Each flight has its own resource constraint. We create a set D of possible triplets (R,S,T) where R represents possible runway

assignments, S represents possible SID routes assignment and T represents possible TMA exit point assignments. For F1, D1 contains ($r_1.m_1.t_1$) number of triplets.

- CFMU Constraint: To assign takeoff time to a flightwe make sure that the takeoff time must lie in the CFMU time interval of the flight. For any flight cfmuSTART<= takeoff<=cfmuEND. This we termed as CFMU constraint.
- Separation Constraint: Here we consider two types of separation time between consecutive flights, the default separation time and vortex separation time. If an aircraft is taking off is in a smaller weight class or a larger speed class than its previous flight that takes off from the same runway then vortex separation time is observed otherwise default separation time is to followed. This we considered as Separation Constraint.
- Topology Constraint: For each airport, there is a topology that represents the connection among runways, SID routes and TMA exit points.The resource assignment to flights must follow this connection topology. This we termed as topology constraint.
- Tower Control Constraint: These are the constraints that priorities the takeoff scheduling among flights like flight F4 must takeoff before flight F1. These we termed as Tower Control Constraint.

Once we obtain the constraints and agents, our next step is to get a directed graph where agents are the nodes and the there is an edge between two agents(nodes) if (Large Separation time + CFMU time interval) of agents overlap. The direction of edge between any two nodes is based on alpha numeric order that is, if the flight F4is connected with flight F7, then direction of edge is from F4 to F7, where F7 is the constraint evaluating agent. A dynamic link is created among agents if they are instantiated with the same runway.

Ok? Message format – (Ok?, (F1, F1_value)) here F1_value is a triplet.

Nogood message format – (nogood,(F1, F1_value), (F2, F2_value), …)

Constraint evaluation is done for takeoff time only keeping in view of dynamic link when created while using the same runway to adjust separation constraint. For resource constraint checking and topology constraint checking, it is to be done by the automated agents itself and does not require intervention from other agents.

Analysis

Here in our model, flights takeoff time is instantiated concurrently in the allotted CFMU time interval. The constraint checking is done at two stages, first at individual agent level where each agent check resource and topology constraints during instantiation. At the second stage, the separation constraint is checked by the constraint evaluating agent. Dynamic link may be created when nogood is created and runway sharing is observed.

Change in values to variables is asynchronous, so, every time the process is instantiated the run time varies.

Conclusion

The model presented in this paper is well suited for implementation by any object-oriented programming language. Since, the changes to associated variables is distributed among agents so there is less idle time for agents to wait for change. Hence, it reaches to the solution faster than CSP models. Since, the changes in the variable values are asynchronous, every other time when the method runs, it provides the solution in different time. Sometimes it may reach to solution early depending upon which type of

asynchronous changes have been occurred during run phase. Further, the method suits to a verity of scheduling problems like timetable scheduling of trains, buses, etc.

The solution strategy may further be enhanced by different solution approach by asynchronous weak-commitment search method.

References

[1] S. Zelinkski and T. Romer, "An Airspace Concept Evaluation System Characterization of National Airspace System Delay," AIAA 4th Aviation Technology, Integration and Operations(ATIO), September 2004.

[2] Gupta, G., Malik, W., and Jung, Y. \Incorporating Active Runway Crossings in Airport Departure Scheduling," AIAA Guidance, Navigation and Control Conference, Toronto, Canada, August 2-5, 2010.

[3] P. van Leeuwen, H.H. Hesselink and J.H.T. Rohling," Scheduling aircraft using constraint satisfaction,"electronic notes in theoretical computer science 76(2002).

[4] Ranjan Kumar Thakur, Ram Baksh, Arabind Kumar and Aditya PratapSingh," Constraint Satisfaction Problem (CSP) Based Implementation of Scheduling aircraft at Runway," International Journal of Science and Research (IJSR) volume 3 Issue 3, March 2014.

[5] Hesselink, H. and S. Paul, Planning aircraft movements in airports with constraint satisfaction, in: Proceedings of the Third IMA Conference on Mathematics in Transport Planning andControl (1998)

[6] Roberts, S. and E. Foster, "D-MAN User Guide," (2001).

[7] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms," IEEE Trans. On Knowledge and DATA Engineering, vol. 10, No. 5 September 1998.

[8] A. H. Bond and L. Gasser, Eds., Readings in distributed Artificial Intelligence, Morgan Kaufmann, 1988.

[9] J. Doyle, "A truth maintenance system", Artificial Intelligence, vol. 2, pp. 231-272, 1979.

[10] M. N. Huhns and D. M. Bridgeland, "Multiagent truth maintenance", IEEE Transactions on systems, Man and Cybernetics, vol. 21, no. 6, pp. 1437-1445, 1991.

[11] S. E. Conry, K. Kuwabara, V. R. Meyer, "Multiagent negotiation for distributed constraint satisfaction", IEEE Tracsactions on systems, man and Cybernetics, vol. 21, no. 6, pp. 1462-1477, 1991.