Optimizing Integrated Flight-Crew-Aircraft Scheduling: Simulated Annealing for Effective Solutions in the Face of Irregular Operations

Gürkan Güven Güner[1(✉)] and Serpil Erol[2]

[1] Department of Industrial Engineering, University of Turkish Aeronautical Association, Ankara, Turkey
gguner@thk.edu.tr
[2] Department of Industrial Engineering, Gazi University, Ankara, Turkey
serpiler@gazi.edu.tr

Abstract. In the face of increasing irregular operations due to global pandemics and other disruptions, effective scheduling of flight-crew-aircraft resources has become crucial for the efficient operation of airlines. This work presents a novel approach for solving the integrated airline scheduling problem. The proposed solution approach provides a comprehensive solution to small-sized real-world instances of this challenging problem. The solution methodology comprises an initial heuristic algorithm, followed by a simulated annealing heuristic method, to efficiently generate and optimize scheduling solutions. The initial heuristic ensures a strong starting point, while the simulated annealing algorithm dynamically explores the search space to find optimal solutions. The studies in the literature on the application of simulated annealing in airline scheduling, have demonstrated the effectiveness of the simulated annealing approach in finding optimal and near-optimal solutions. Through extensive experimentation, our approach consistently produces high-quality schedules with reduced operational costs. The significance of this work lies in its ability to address the urgent need for efficient airline scheduling amidst the irregular operations caused by global pandemics, which disrupt traditional planning approaches. By integrating the scheduling of flight, crew, and aircraft resources, our solution optimizes resource utilization, enhances operational efficiency, and mitigates the impact of unexpected events. Thorough review of existing literature has been conducted and the uniqueness of our approach is ensured. By presenting our findings, it is aimed to contribute to the advancement of the field and foster discussions on addressing the challenges of modern airline operations.

Keywords: Integrated airline scheduling · Irregular operations · Resource optimization · Simulated annealing

N. M. Durakbasa and M. G. Gençyılmaz (Eds.): ISPR 2023, LNME, pp. 360–373, 2024.
https://doi.org/10.1007/978-3-031-53991-6_28

1 Introduction

In the dynamic and complex world of aviation, the effective scheduling of flight crews and aircraft is a critical aspect that directly impacts operational efficiency, passenger satisfaction, and airline profitability. The intricate interplay between crew members, their qualifications, aircraft availability, and the ever-changing nature of air travel demands sophisticated optimization techniques to navigate the challenges posed by irregular operations. This study explores the application of Simulated Annealing (SA)—a powerful metaheuristic algorithm—to optimize integrated flight-crew-aircraft scheduling in the face of such disruptions, aiming to achieve effective solutions that mitigate the impact of unforeseen events and maintain operational resilience. Irregular operations, including weather disturbances, air traffic congestion, mechanical issues, and crew unavailability, can cause disruptions to airline schedules, leading to delays, cancellations, and missed connections. These disturbances not only result in increased costs for airlines but also have a profound impact on customer satisfaction. As a result, airlines strive to minimize the adverse effects of irregular operations by developing robust scheduling solutions that allow for rapid adaptations and optimal resource allocation.

SA, a metaheuristic optimization technique inspired by the annealing process in metallurgy, has proven to be a valuable tool in addressing complex scheduling problems. SA has gained popularity in various domains for its ability to find near-optimal solutions in large solution spaces, even in the presence of highly nonlinear and non-convex objective functions. By simulating the gradual cooling and reorganization of atoms in a solid, this algorithm emulates the search for an optimal configuration through iterative adjustments and stochastic transitions, allowing for the exploration of a wide range of solutions. In the context of integrated flight-crew-aircraft scheduling, SA can effectively handle the complexities arising from the need to align crew qualifications, contractual regulations, and aircraft availability. By considering a multitude of constraints, such as crew rest requirements, duty time limitations, and aircraft maintenance needs, SA provides a robust framework to optimize the allocation of flight crews to aircraft in the face of irregular operations. Furthermore, this technique can account for various factors, such as minimizing delays and cancellations, maximizing crew utilization, and ensuring efficient resource utilization, all while maintaining compliance with regulatory standards.

This study delves into the intricacies of integrating SA with flight-crew-aircraft scheduling, highlighting its potential in mitigating the disruptive impact of irregular operations. By harnessing the power of this optimization method, airlines can enhance their operational performance, reduce costs, and improve customer satisfaction by efficiently managing the allocation of crews and aircraft during challenging circumstances. Through a comprehensive exploration of the SA approach, we aim to provide valuable insights and practical guidelines for aviation professionals seeking effective solutions in the complex realm of integrated flight-crew-aircraft scheduling.

2 Literature Review

There are many studies in the literature that propose SA method for different airline scheduling problems. These problems are sometimes considered individually and sometimes in an integrated manner. The following Table 1 indicates just the studies, which are mainly considered to implement SA for the integrated airline scheduling problem, in the literature.

Table 1. The considered studies to use SA for the ıntegrated airline scheduling problem

Study	Problem
Zheng et al. [1]	Aircraft scheduling
Jamili [2]	Large scale aircraft routing problem
Chen et al. [3]	Crew scheduling
Jungai and Hongjun [4]	Flight delay scheduling

The inception of the mathematical model was sparked by a study conducted by Stojković and Soumis [5, 6], aiming to present an optimization framework addressing the challenge of concurrent operational flight and crew scheduling. Subsequently, the comprehensive integrated flight crew scheduling model was formulated with a primary focus on aircraft scheduling considerations. Therefore, the studies [1, 2, 7–28] focusing on aircraft scheduling (fleet assignment and/or aircraft routing) problems with integrating flight and/or crew scheduling problems in the literature are considered to suggest an optimization model for the integrated flight-aircraft-crew scheduling problem. To develop the optimization problem about only the crew scheduling perspective, the latest studies [29–37] in the literature are considered.

3 Methodology

In this study, the formidable task of optimizing integrated flight-crew-aircraft scheduling in the challenging context of irregular operations was approached with a powerful tool—SA, implemented using the R programming language. R's versatility in handling data and conducting complex optimization procedures played a pivotal role in our research. By leveraging R, we were able to develop and execute an efficient SA algorithm, enabling us to tackle the intricate scheduling problem faced by the aviation industry. The flexibility and ease of use of R allowed us to adapt our code to the specific needs of this demanding domain, ensuring that our solutions were not only effective but also tailored to the unique constraints of the airline industry. Through this study, R has demonstrated its significance as a valuable resource in the quest for more resilient and cost-effective integrated flight operations. The steps involved in the SA algorithm can be summarized as below:

Initialize: Start with an initial solution. This solution can be generated randomly or based on some heuristics.
Set Parameters: Define parameters like initial temperature and cooling rate. These parameters control the annealing schedule.
Define a Stopping Criterion: Decide when to stop the algorithm. This could be a maximum number of iterations, a specific temperature threshold, or other criteria related to the problem.
Main Loop:
Repeat the following steps until the stopping criterion is met.
Generate a Neighbor Solution: Create a neighboring solution by making a small change to the current solution. The nature of this change depends on the problem, but it should be based on some probabilistic criteria.
Calculate Cost: Compute the cost or objective function value of the new solution and the current solution.
Evaluate Acceptance: Determine whether to accept the neighbor solution. Typically, if the neighbor solution has a lower cost, it's accepted. However, it may also be accepted with some probability if it has a higher cost, based on a probabilistic criterion (Metropolis criterion).
Update Current Solution: If the neighbor solution is accepted, set it as the new current solution.
Update Best Solution: If the current solution is better than the best solution found so far, update the best solution.
Cooling: Reduce the temperature according to the cooling rate. This reduces the likelihood of accepting worse solutions as the algorithm progresses.
Output: Once the stopping criterion is met, or after a predefined number of iterations, output the best solution found during the search.

A new mathematical model, which is constructed by using Mixed Integer Non-linear Programming (MINLP) in this study includes the following objectives:

- Minimizing the total flight-crew assignment cost,
- Minimizing the total penalty cost of using flights that have long connection/waiting time (in-active time),
- Minimizing the total flight delay cost and
- Minimizing the total aircraft assignment cost. The aircraft assignment cost includes two main cost items: the total operating cost and the total passenger spill cost.

The considered constraints are related to:

- Flight coverage constraints
- Flight precedence constraints
- Task assignment constraints
- Network flow balance constraints
- Unit flow constraints
- Constraints for the requirement for nonlinear compatibility between flow and time variables.
- Time windows constraints

The small-sized problem of the AnadoluJet airline company is considered in this study. The integrated problem modelled by using MINLP was solved by SA algorithm in the R programming language. The suggested code in R programming language in this study is as the following. The required packages to run the code and the explanations of each line (considered parameters, decision variables, objective function, SA steps, etc.) are shown below.

```
#install.packages("optimization")
#library("optimization")
#Install packages to convert minutes to Hour-Minute-Second format
#install.packages("tidyverse")
#install.packages("magrittr")
#install.packages("lubridate")
#install.packages("dplyr")
#install.packages("hms")
#library(lubridate)
#library(dplyr)
#library(hms)
#library(magrittr)
#library(tidyverse)

# Load the 'gtools' package for generating combinations
#install.packages("gtools")
library(gtools)
#library(stringr)
num_nodes <- 5  # Number of flights
min_node <-1
max_node <-5
num_crews <- 7  # Number of crews
# Define Parameters to Calculate Aircraft Cost
#SC (Seat Capacity)
SC <- c(189, 189, 176, 176, 189, 189, 189)
#CASM (Cost per Available Seat Mile)
CASM <- c(0.042, 0.042, 0.038, 0.038, 0.042, 0.042, 0.042)
#RASM (Revenue per Available Seat Mile)
RASM <- c(0.15, 0.15, 0.12, 0.12, 0.15, 0.15, 0.15)
#RR (Recapture Rate)
RR <- 0.15
#ENPS (Expected Number of Passenger Spill)
ENPS <- c(0, 1000, 2000, 1500, 3000)
# Create a 2-dimensional array to represent the flight duration parameter FD
FD <- array(0, dim = c(num_nodes, num_nodes))  # Initialize the array with zeros
FD[1,1] <- 0
```

```
FD[1,2] <- 70
FD[1,3] <- 63
FD[1,4] <- 80
FD[1,5] <- 75
FD[2,1] <- 70
FD[3,1] <- 63
FD[4,1] <- 80
FD[5,1] <- 75
# Define the hub airport
i <- sample(setdiff(min_node:max_node,2:5),1,rep=FALSE)
i
# Generate all combinations of four different numbers between 2 and 5 without replication
combinations_j <- permutations(n = 4, r = 4, v = 2:5, repeats.allowed = FALSE)
combinations_j
combinations_k <- permutations(n = 7, r = 4, v = 1:7, repeats.allowed = FALSE)
combinations_k
combinations_a <- permutations(n = 7, r = 4, v = 1:7, repeats.allowed = FALSE)
combinations_a
# Get the number of combinations in combinations_j
num_combinations <- nrow(combinations_j)
num_combinations
# Generate a random index between 1 and num_combinations
random_index <- sample(1:num_combinations, 1)
# Assign j as the randomly selected combination
j <- combinations_j[random_index, ]
j
# Get the number of combinations in combinations_k
num_combinations_k <- nrow(combinations_k)
num_combinations_k
# Generate a random index between 1 and num_combinations
random_index_k <- sample(1:num_combinations_k, 1)
random_index_k
# Assign j as the randomly selected combination
k <- combinations_k[random_index_k, ]
k
# Set the binary decision variable Xijk to 1 for crew k assigned to flight arc (i, j)
# Create a 3-dimensional array to represent the binary decision variable Xijk
X <- array(0, dim = c(num_nodes, num_nodes, num_crews))  # Initialize the array with zeros
X
for(t in 1:4) {
  X[i, j[t], k[t]] <- 1
  t <- t+1
}
X[i, j[1], k[1]]
X
for(t in 1:4) {
  X[j[t], i, k[t]] <- 1
  t <- t+1
}
# Create a 3-dimensional array to represent the binary decision variable Zijk
num_aircrafts <- 7  # Number of crews
Z <- array(0, dim = c(num_nodes, num_nodes, num_aircrafts))  # Initialize the array with zeros
# Get the number of combinations in combinations_k
num_combinations_a <- nrow(combinations_a)
num_combinations_a
# Generate a random index between 1 and num_combinations
random_index_a <- sample(1:num_combinations_a, 1)
random_index_a
# Assign j as the randomly selected combination
a <- combinations_a[random_index_a, ]
a
# Set the binary decision variable Zija to 1 for aircraft k assigned to flight arc (i, j)
for(t in 1:4) {
  Z[i, j[t], a[t]] <- 1
  t <- t+1
}
Z[i, j[1], a[1]]
```

```
for(t in 1:4) {
  Z[j[t], i, a[t]] <- 1
  t <- t+1
}
# Print the value of the binary decision variable Zija
print(Z[i, j[1], a[1]])
# Create an array to track the arrival times of aircraft at the hub
ArrivalTime <- array(0, dim = c(num_aircrafts))
ArrivalTime[a] <- 0  # Initialize the array with zeros
# Create a 3-dimensional array to represent the integer decision variable Tija for the departure times
DT <- array(0, dim = c(num_nodes, num_nodes, num_aircrafts))  # Initialize the array with zeros
# Generate the departure times for flights
for (t in 1:4) {
  # Check the availability of the aircraft before assigning a departure time
  if (ArrivalTime[a[t]] < 30) {
    DT[i, j[t], a[t]] <- 30
  } else {
    DT[i, j[t], a[t]] <- ArrivalTime[a[t]]
  }
  ArrivalTime[a[t]] <- DT[i, j[t], a[t]] + FD[i, j[t]]
  DT[j[t], i, a[t]] <- ArrivalTime[a[t]] + 30
}
DT
ArrivalTime
#Calculate the objective value
# Define the objective function
calculateObjective <- function(X, Z, DT) {
  TotalCost <- 0
  for (t in 1:4) {
    TotalCost <- TotalCost +
      10 * sum(X[i, j[t], k[t]]) +
      10 * sum(DT[i, j[t], a[t]]) +
      sum(Z[i, j[t], a[t]] * (
        (FD[i, j[t]] * CASM[a[t]] * SC[a[t]]) +
          (ENPS[j[t]] * RASM[a[t]] * FD[i, j[t]] * (1 - RR))
      ))
  }
  TotalCost
}
# Call the calculateObjective function to calculate the total cost
TotalCost <- calculateObjective(X, Z, DT)
# Print the value of TotalCost
print(TotalCost)
InitialSolution <- TotalCost
#Simulated Annealing
# Define the neighborhood function
# This function generates a neighbor solution given the current solution
# Generate a neighbor solution based on the current solution (schedule)
# Return the modified schedule
NeighborTotalCost <- array(0, dim = 23)
NeighborTotalCost
for(tt in 1:24){
  X <- array(0, dim = c(num_nodes, num_nodes, num_crews))  # Initialize the array with zeros
  Z <- array(0, dim = c(num_nodes, num_nodes, num_aircrafts))  # Initialize the array with zeros
  DT <- array(0, dim = c(num_nodes, num_nodes, num_aircrafts))  # Initialize the array with zeros
  TotalCost <- 0
  j <- combinations_j[tt,]
  for(t in 1:4) {
    X[i, j[t], k[t]] <- 1
    t <- t+1
  }
  for(t in 1:4) {
    X[j[t], i, k[t]] <- 1
    t <- t+1
  }
  for(t in 1:4) {
    Z[i, j[t], a[t]] <- 1
```

```
    t <- t+1
  }
  for(t in 1:4) {
    Z[j[t], i, a[t]] <- 1
    t <- t+1
  }
  for(t in 1:4) {
    DT[i, j[t], a[t]] <- 30
    t <- t+1
  }
  for(t in 1:4) {
    DT[j[t], i, a[t]] <- DT[i, j[t], a[t]] + FD[i,j[t]] + 30
    t <- t+1
  }
  for(t in 1:4) {
    TotalCost <- TotalCost + 10*sum(X[i, j[t], k[t]]) + 10*sum(DT[i, j[t], a[t]]) + sum(Z[i, j[t],
a[t]]*((FD[i,j[t]]*CASM[a[t]]*SC[a[t]])+(ENPS[j[t]]*RASM[a[t]]*FD[i,j[t]]*(1-RR))))
    t <- t+1
  }
  NeighborTotalCost[tt] <- TotalCost
  tt <- tt + 1
}
NeighborTotalCost
# Define the simulated annealing steps
# These steps perform the simulated annealing algorithm to find the optimal solution
# Set the initial solution, initial temperature, cooling rate, and maximum iterations
Temperature <- 90
coolingRate <- 0.1
maxIterations <- 200
currentSolution <- InitialSolution
bestSolution <- currentSolution
# Create a vector to store the best solutions at each iteration
bestSolutions <- numeric(maxIterations)
for (iteration in 1:maxIterations) {
  # Generate a neighbor solution based on the current solution (schedule)
  # Return the modified schedule
  # You can consider swapping flights, reassigning crews or aircraft, etc.
  r <- sample(2:24, 1)
  newSolution <- NeighborTotalCost[r]
  # Define the acceptance probability function
  # This function determines whether to accept a neighbor solution or not
  AcceptanceProbability <- exp((bestSolution - NeighborTotalCost[r])) / Temperature
  if (newSolution < bestSolution) {
    bestSolution <- newSolution
  } else if (AcceptanceProbability > runif(1)) {
    bestSolution <- newSolution
  }
  # Store the best solution at the current iteration
  bestSolutions[iteration] <- bestSolution
  Temperature <- Temperature * coolingRate
}
bestSolutions
bestSolution
# Plot the graph
plot(1:maxIterations, bestSolutions, type = "l", xlab = "Iteration", ylab = "Best Solution")
# Add a title to the plot
title(main = "Best Solution over Iterations")
# Find the last arrival times for each crew and aircraft
lastArrivalTimes <- numeric(num_aircrafts)
for (t in 1:4) {
  if (X[j[t], i, k[t]] == 1) {
    arrivalTime <- DT[j[t], i, a[t]] + FD[j[t], i]
    if (arrivalTime > lastArrivalTimes[a[t]]) {
      lastArrivalTimes[a[t]] <- arrivalTime
    }
  }
}
```

```
lastArrivalTimes <- numeric(num_crews)
for (t in 1:4) {
  if (X[j[t], i, k[t]] == 1) {
    arrivalTime <- DT[j[t], i, a[t]] + FD[j[t], i]
    if (arrivalTime > lastArrivalTimes[a[t]]) {
      lastArrivalTimes[a[t]] <- arrivalTime
    }
  }
}
# Print the table of optimal decision variables, departure times, crew, and aircraft
optimalTable <- data.frame(X = character(), DepartureTime = numeric(), Crew = numeric(), Aircraft = numeric())
# Function to convert minutes to Hour-Minute format with base time at 08:00
convertToHM <- function(minutes) {
  minutes <- minutes + 480  # Add 480 minutes (8 hours) to the input minutes
  hours <- floor(minutes / 60)
  minutes <- minutes %% 60
  sprintf("%02d:%02d", hours, minutes)
}
# Create a vector to store the arrival times
arrivalTimes <- numeric(length = nrow(optimalTable))
# Calculate the arrival times for each flight in the optimal table
for (t in 1:4) {
  if (X[i, j[t], k[t]] == 1) {
    arrivalTime <- DT[i, j[t], a[t]] + FD[i, j[t]]
    arrivalTimes[optimalTable$Flight == paste("From hub (", i, ") to", j[t])] <- arrivalTime
  }
  if (X[j[t], i, k[t]] == 1) {
    arrivalTime <- DT[j[t], i, a[t]]
    arrivalTimes[optimalTable$Flight == paste("From", j[t], "to hub (", i, ")")] <- arrivalTime
  }
}
# Convert the arrival times to HM format
arrivalTimes <- sapply(arrivalTimes, convertToHM)
# Add the arrival times as a new column to the optimalTable
optimalTable <- cbind(optimalTable, ArrivalTime = arrivalTimes)
# Add flights from hub to destination nodes
for (t in 1:4) {
  if (X[i, j[t], k[t]] == 1) {
    row <- c(paste("From hub (", i, ") to", j[t]), DT[i, j[t], a[t]], k[t], a[t])
    optimalTable <- rbind(optimalTable, row)
  }
}
# Add flights from arrival nodes to hub
for (t in 1:4) {
  if (X[j[t], i, k[t]] == 1) {
    row <- c(paste("From", j[t], "to hub (", i, ")"), DT[j[t], i, a[t]], k[t], a[t])
    optimalTable <- rbind(optimalTable, row)
  }
}
colnames(optimalTable) <- c("Flight", "DepartureTime", "Crew", "Aircraft")
print(optimalTable)
# Create a new column for days in the optimal table
optimalTable$Day <- ""
# Define the base day as Monday at 08:00 o'clock
baseDay <- as.POSIXct("2023-06-05 08:00:00")
# Function to calculate the day based on departure time
calculateDay <- function(departureTime) {
  departureDateTime <- baseDay + minutes(departureTime)
  weekdays(departureDateTime)
}
# Calculate the day for each departure time in the optimal table
optimalTable$Day <- sapply(optimalTable$DepartureTime, calculateDay)
# Convert the "DepartureTime" values from factors to numeric
optimalTable$DepartureTime <- as.numeric(as.character(optimalTable$DepartureTime))
# Convert the numeric "DepartureTime" values to HM format in the optimalTable
optimalTable$DepartureTime <- sapply(optimalTable$DepartureTime, convertToHM)
# Print the updated table with "DepartureTime" in HM format
```

```
print(optimalTable)
# Create a new column for flight duration in the optimal table
optimalTable$FlightDuration <- 0
# Add flight durations from hub to destination nodes
for (t in 1:4) {
  if (X[i, j[t], k[t]] == 1) {
    flightDuration <- FD[i, j[t]]
    optimalTable$FlightDuration[optimalTable$Flight == paste("From hub (", i, ") to", j[t])] <- flightDuration
  }
}
# Add flight durations from arrival nodes to hub
for (t in 1:4) {
  if (X[j[t], i, k[t]] == 1) {
    flightDuration <- FD[j[t], i]
    optimalTable$FlightDuration[optimalTable$Flight == paste("From", j[t], "to hub (", i, ")")] <- flightDuration
  }
}
# Add flight indexes to the existing table
flightIndexes <- paste("F", 1:nrow(optimalTable), sep = "")
optimalTable <- cbind(Flight = optimalTable[, 1], Flight_Index = flightIndexes, optimalTable[, 2:ncol(optimalTable)])
# Define the desired column order
column_order <- c("Flight_Index", "Flight", setdiff(colnames(optimalTable), c("Flight_Index", "Flight", "Crew", "Aircraft")),
"Crew", "Aircraft")
# Reorder the columns in the optimal table
optimalTable <- optimalTable[, column_order]
# Combine the "DepartureTime" and "Day" columns into a single column called "DepartureTime":
optimalTable$DepartureTime <- paste(optimalTable$DepartureTime, optimalTable$Day)
arrivalTimes <- numeric(length = nrow(optimalTable))
for (t in 1:4) {
  if (X[i, j[t], k[t]] == 1) {
    arrivalTime <- DT[i, j[t], a[t]] + FD[i, j[t]]
    arrivalTimes[optimalTable$Flight == paste("From hub (", i, ") to", j[t])] <- arrivalTime
  }

  if (X[j[t], i, k[t]] == 1) {
    arrivalTime <- DT[j[t], i, a[t]] + FD[i, j[t]]
    arrivalTimes[optimalTable$Flight == paste("From", j[t], "to hub (", i, ")")] <- arrivalTime
  }
}
# Function to convert minutes to Hour-Minute format with base time at 08:00
convertToHM <- function(minutes) {
  minutes <- minutes + 480  # Add 480 minutes (8 hours) to the input minutes
  hours <- floor(minutes / 60)
  minutes <- minutes %% 60
  sprintf("%02d:%02d", hours, minutes)
}
# Convert the arrival times to HM format
arrivalTimes <- sapply(arrivalTimes, convertToHM)
optimalTable <- cbind(optimalTable, ArrivalTime = arrivalTimes)
# Combine the "ArrivalTime" and "Day" columns into a single column called "ArrivalTime":
optimalTable$ArrivalTime <- paste(optimalTable$ArrivalTime, optimalTable$Day)
# Remove the "Day" column from the optimal table:
optimalTable$Day <- NULL
# Calculate crew utilization percentage
crewUtilization <- sum(X) / (num_nodes * num_crews) * 100
# Calculate aircraft utilization percentage
aircraftUtilization <- sum(Z) / (num_nodes * num_aircrafts) * 100
optimalTable <- optimalTable[, c("Flight_Index", "Flight", "DepartureTime", "ArrivalTime", "FlightDuration", "Crew",
"Aircraft")]
# Display the updated table with modified column names
print(optimalTable)
bestSolution
# Create vectors to store duty and flight times for each crew
dutyTimes <- numeric(num_crews)
flightTimes <- numeric(num_crews)
# Calculate duty and flight times for each crew
for (crew in 1:num_crews) {
```

```
  # Initialize duty and flight times for the current crew
  dutyTime <- 0
  flightTime <- 0
  # Iterate over flights
  for (t in 1:4) {
    if (X[i, j[t], crew] == 1) {
      dutyTime <- dutyTime + FD[i, j[t]]  # Add flight duration to duty time
      flightTime <- flightTime + FD[i, j[t]]  # Add flight duration to flight time
    }
    if (X[j[t], i, crew] == 1) {
      dutyTime <- dutyTime + FD[i, j[t]]  # Add flight duration to duty time
      flightTime <- flightTime + FD[i, j[t]]  # Add flight duration to flight time
    }
  }
  # Add resting time, briefing time, and debriefing time to duty time
  if (flightTime > 0) {
    dutyTime <- dutyTime + 60 + 30 + 30  # Assuming 60 minutes resting time, 30 minutes briefing time, and 30 minutes
debriefing time
  }
  # Store duty and flight times for the current crew
  dutyTimes[crew] <- dutyTime
  flightTimes[crew] <- flightTime
}
# Calculate the total flight times of each aircraft in the optimal solution
totalFlightTimes <- array(0, dim = num_aircrafts)
for (t in 1:4) {
  if (X[i, j[t], k[t]] == 1) {
    totalFlightTimes[a[t]] <- totalFlightTimes[a[t]] + FD[i, j[t]]
  }
  if (X[j[t], i, k[t]] == 1) {
    totalFlightTimes[a[t]] <- totalFlightTimes[a[t]] + FD[i, j[t]]
  }
}
# Display the updated table with modified column names
print(optimalTable)
bestSolution
# Print the utilization percentages
cat("Crew Utilization: ", crewUtilization, "%\n")
cat("Aircraft Utilization: ", aircraftUtilization, "%\n")
# Print duty and flight times for each crew
for (crew in 1:num_crews) {
  cat("Crew", crew, "Duty Time:", dutyTimes[crew], "minutes\n")
  cat("Crew", crew, "Flight Time:", flightTimes[crew], "minutes\n")
}
# Print the total flight times of each aircraft
cat("Total Flight Times:\n")
for (a in 1:num_aircrafts) {
  cat("Aircraft", a, ":", totalFlightTimes[a], "\n")
}
# Print the last arrival times for each crew and aircraft
for (a in 1:num_aircrafts) {
  cat("Last arrival time for Aircraft", a, ":", convertToHM(lastArrivalTimes[a]), "\n")
}
for (crew in 1:num_crews) {
  print(paste("Crew", crew, "Last Arrival Time:", lastArrivalTimes[crew]))
}
```

The following Fig. 1 gathered by R code shows the best solutions over iterations.

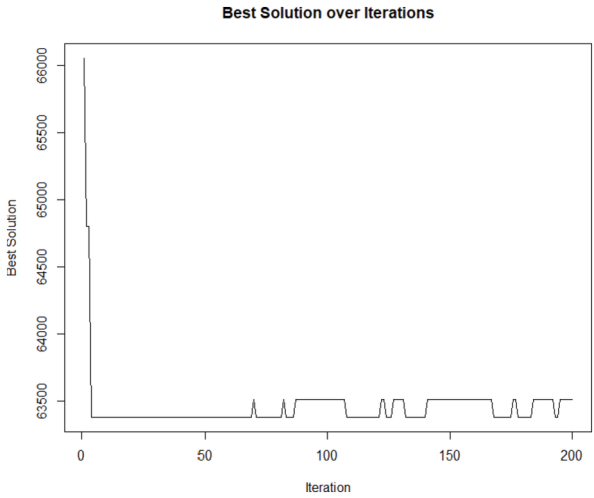

Fig. 1. The best solutions over iterations

After applying the SA algorithm using the R programming language to solve the complex integrated flight-crew-aircraft scheduling problem, which is often modelled as a MINLP problem, we have observed the remarkable utility of this approach in delivering highly efficient solutions. The combination of SA adaptability and the computational capabilities of R proved to be a formidable force in addressing the intricate scheduling challenges posed by the aviation industry. The algorithm, driven by R, successfully navigated the complex solution space, yielding solutions that not only met operational constraints but also demonstrated cost-effectiveness and adaptability in the face of irregular operations. This underscores the power of SA as a practical and effective tool in optimizing integrated flight-crew-aircraft scheduling, contributing to more streamlined and resilient operations in the aviation sector.

4 Conclusions

In the fast-paced and dynamic world of aviation, optimizing integrated flight-crew-aircraft scheduling is a paramount challenge. The intricate web of variables, from crew availability to aircraft maintenance, becomes even more complex when irregular operations disrupt the well-laid plans. In our exploration of this complex problem, we turned to the powerful optimization technique known as SA to find effective solutions that can withstand the turbulence of unexpected disruptions.

SA emerges as a potent ally in the intricate realm of integrated flight-crew-aircraft scheduling, especially when confronted with the turbulence of irregular operations. Its capacity to swiftly adapt to changing circumstances, explore diverse solutions, and minimize costs positions it as a valuable tool for the aviation industry. As airlines continue to grapple with the ever-evolving challenges of scheduling, SA offers a path towards more efficient, resilient, and cost-effective operations, ultimately ensuring smoother journeys for both airlines and passengers in the face of unexpected disruptions. It is aimed that

the applied methodology can be source of inspiration for the decision makers working on the integrated airline scheduling problems.

References

1. Zheng, S., Yang, Z., He, Z., Wang, N., Chu, C., Yu, H.: Hybrid simulated annealing and reduced variable neighbourhood search for an aircraft scheduling and parking problem. Int. J. Prod. Res. **58**(9), 2626–2646 (2020)
2. Jamili, A.: A robust mathematical model and heuristic algorithms for integrated aircraft routing and scheduling, with consideration of fleet assignment problem. J. Air Transp. Manag. **58**, 21–30 (2017)
3. Chen, X., Chen, X., Zhang, X.: Crew scheduling models in airline disruption management. In: IEEE 17th International Conference IE&EM, Xiamen, China, 2010, pp. 1032–1037 (2010)
4. Jungai, T., Hongjun, X.: Optimizing arrival flight delay scheduling based on simulated annealing algorithm. Phys. Procedia **33**, 348–353 (2012)
5. Stojković, M., Soumis, F.: The operational flight and multi-crew scheduling problem. Yugosl. J. Oper. Res. **15**(1), 25–48 (2005)
6. Stojković, M., Soumis, F.: An optimization model for the simultaneous operational flight and pilot scheduling problem. Manag. Sci. Informs **47**(9), 1290–1305 (2001)
7. Awalivian, M.R.F., Sa'Adah, S.: Optimization of aircraft flight scheduling and routing problem using multi-objective antlion optimization. In: ICAICST, pp. 1–6, June 2021
8. Birolini, S., Antunes, A.P., Cattaneo, P., Malighetti, P., Paleari, S.: Integrated flight scheduling and fleet assignment with improved supply-demand interactions. Transp. Res. B: Methodol. **149**, 162–180 (2021)
9. Parmentier, A., Meunier, F.: Aircraft routing and crew pairing: updated algorithms at Air France. Omega **93**, 1–17 (2020)
10. Ahmed, M.B., Mansour, F.Z., Haouarii, M.: Robust integrated maintenance aircraft routing and crew pairing. J. Air Transp. Manag. **73**, 15–31 (2018)
11. Kenan, N., Jebali, A., Diabat, A.: An integrated flight scheduling and fleet assignment problem under uncertainty. Comput. Oper. Res. **100**, 333–342 (2018)
12. Cadarso, L., de Celis, R.: Integrated airline planning: robust update of scheduling and fleet balancing under demand uncertainty. Transp. Res. Part C Emerg. Technol. **81**, 227–245 (2017)
13. Özener, O., Örmeci Matoğlu, M., Erdoğan, G., Haouari, M., Sözer, H.: Solving a large-scale integrated fleet assignment and crew pairing problem. Ann. Oper. Res. **253**(1), 477–500 (2017)
14. Sandamali, G.G.N., Su, R., Zhang, Y., Li, Q.: Flight routing and scheduling with departure uncertainties in air traffic flow management. In: Proceedings of the 13th IEEE ICCA (2017)
15. Weide, O., Ryan, D., Ehrgott, M.: An iterative approach to robust and integrated aircraft routing and crew scheduling. Comput. Oper. Res. **37**(5), 833–844 (2010)
16. Yan, S., Tseng, C.H.: A passenger demand model for airline flight scheduling and fleet routing. Comput. Oper. Res. **29**, 1559–1581 (2002)
17. Yan, S., Tu, Y.: Multifleet routing and multistop flight scheduling for schedule perturbation. Eur. J. Oper. Res. **103**(1), 155–169 (1997)
18. Yan, S., Young, H.-W.: A decision support framework for multi-fleet routing and multi-stop flight scheduling. Transp. Res. A Policy Pract. **30**(5), 379–398 (1996)
19. Papadakos, N.: Integrated airline scheduling. Comput. Oper. Res. **36**, 176–195 (2009)
20. Erdem, E., Aydın, T., Erkayman, B.: Flight scheduling incorporating bad weather conditions through big data analytics: a comparison of metaheuristics. Expert Syst. **38**(8), 1–19 (2021)
21. Kiarashrad, M., Pasandideh, S.H.R., Mohammadi, M.: A mixed-integer nonlinear optimization model for integrated flight scheduling, fleet assignment, and ticket pricing in competitive market. J. Revenue Pricing Manag. **20**, 596–607 (2021)

22. Wei, M., Zhao, L., Ye, Z., Jing, B.: An integrated optimization mode for multi-type aircraft flight scheduling and routing problem. Math. Biosci. Eng. **17**(5), 4990–5004 (2020)
23. Thomaz, S.: Effects of asymmetric demands on airline scheduling decisions in a network. Econ. Transp. **22**, 1–9 (2020)
24. Prakash, R., Piplani, R., Desai, J.: An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput. Transp. Res. Part C Emerg. Technol. **95**, 570–581 (2018)
25. Abdelghany, A., Abdelghany, K., Azadian, F.: Airline flight schedule planning under competition. Comput. Oper. Res. **87**, 20–39 (2017)
26. Bennell, J.A., Mesgarpour, M., Potts, C.N.: Dynamic scheduling of aircraft landings. Eur. J. Oper. Res. **258**(1), 315–327 (2017)
27. Ng, K.K.H., Lee, C.K.M., Chan, F.T.S., Qin, Y.: Robust aircraft sequencing and scheduling problem with arrival/departure delay using the min-max regret approach. Transp. Res. E: Logist. Transp. Rev. **106**, 115–136 (2017)
28. Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D.: Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas. Transp. Res. C: Emerg. Technol. **80**, 485–511 (2017)
29. Zhang, C., Gu, C., Gong, M., Wu, K., Xia, H., Zhang, F.: An improved fast search multi-objective genetic algorithm for airline crew scheduling problems. In: Proceedings of the 40th Chinese Control Conference, pp. 1900–1904, July 2021
30. Bayliss, C., De Maere, G., Atkin, J.A.D., Paelinck, M.: Scheduling airline reserve crew using a probabilistic crew absence and recovery model. J. Oper. Res. Soc. **71**(4), 543–565 (2020)
31. Quesnel, F., Desaulniers, G., Soumis, F.: A branch-and-price heuristic for the crew pairing problem with language constraints. Eur. J. Oper. Res. **283**(3), 1040–1054 (2020)
32. Antunes, D., Vaze, V., Antunes, A.P.: A robust pairing model for airline crew scheduling. Transp. Sci. **53**(6), 1751–1771 (2019)
33. Deveci, M., Demirel, N.Ç.: Evolutionary algorithms for solving the airline crew pairing problem. Comput. Ind. Eng. **115**, 389–406 (2018)
34. Quesnel, F., Desaulniers, G., Soumis, F.: A new heuristic branching scheme for the crew pairing problem with base constraints. Comput. Oper. Res. **80**, 159–172 (2017)
35. Zeren, B., Özkol, İ: A novel column generation strategy for large scale airline crew pairing problems. Expert Syst. Appl. **55**, 133–144 (2016)
36. Soykan, B., Erol, S.: An optimization-based decision support framework for robust airline crew pairing process. In: Ocalir-Akunal, E.V. (Ed.) Using Decision Support Systems for Transportation Planning Efficiency, IGI Global, Hershey, PA (2016)
37. Deng, G.-F., Lin, W.-T.: Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Syst. Appl. **38**(5), 5787–5793 (2011)