POLITÉCNICO DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Executive Summary - Milestone 1

Ana Paula Rabelo – 1222618

Regiana Barbosa - 1222617

This report presents some information about the algorithm for the aircraft scheduling problem (ASP) developed using the Scala programming language. The domain model consists of the entities Aircraft, Agenda, Runway, Operation, and ScheduleAircrafts, which are used to represent the problem constraints and solutions. The AgendaIO object is responsible for parsing the XML input file and creating an Agenda object that contains all necessary information. The Constraints object defines auxiliary functions used in the algorithm.

To better represent the problem constraints, we defined two opaque types, positiveInteger and nonNegativeInteger, along with their companion objects.

The algorithm uses a recursive function to allocate aircraft to runways, with an immutable Queue data structure used to store Aircraft and Runway objects. The algorithm dequeues Aircraft objects and calculates the maximum and minimum landing window times, accounting for emergency situations. If the current time is within this window, it looks for an available runway with the minimum delay time. If a runway is available, it creates an Operation object representing the aircraft allocation and calls the recursive function with the remaining Aircraft objects, the same runways queue, the new operations list, the target time of the new Aircraft object, and the same maxDelayTime. If no runway is available, the function checks whether it is an emergency aircraft or not. If the current time is not within the minimum and maximum window times, it inserts the dequeued Aircraft object into a new queue and calls itself recursively. The function returns a list of Operation objects, which are used to create a ScheduleAircrafts object.

We chose to use an immutable Queue due to its first-in-first-out (FIFO) property, which is suitable for the problem's scheduling requirements. Queues are easily implemented in Scala and provide efficient access to the first element in the queue, enabling processing of the aircraft in the order they were added to the queue.

The project also includes error-handling strategies and unit and functional tests to evaluate the algorithm's functionality. Besides that, we also used high-order functions, pattern matching, immutability, and the frameworks scalatest and scalacheck for the tests.

However, the algorithm currently does not consider the limitation of certain runways to specific classes of aircraft, which may lead to imprecise results. Further improvements are necessary to address this limitation.