1 Abstract

APACHE is an automated stand allocation system for large airports developed with the constraint programming language CHIP. It is an interactive, incremental tool which takes numerous user constraints and preferences into account. APACHE handles resource allocation and maximizes passenger flow through contact gates. Scheduling time is below one minute, allowing quasi real-time use.

2 Introduction

In this paper we describe APACHE (Airport Parking Assignment in a Constraint Handling Environment), an automated stand allocation system for large airports with several terminals. Different types of constraints, e.g. parking restrictions, safety/security requirements, preferred terminals and resource allocation for ground movements can be combined and are taken automatically into account. Constraints can be either imperative, i.e. they must be satisfied, or preferences, i.e. they should be satisfied if possible. The system searches for minimal resource usage and maximal passenger flow through direct contact gates.

APACHE is an interactive, incremental tool. Constraints and user decisions, like resource down-times or flight delays, are either entered graphically or by database updates. Scheduling time for a complete day of flight operations is below one minute. This allows for automated updating of the parking plan in a continually changing environment. Manual assignment of stands is possible, but not required. In another mode of utilisation, the program can be used for forecasting/planning of the impact of new/changed parking stands or terminal gates.
APACHE was developed with CHIP, a powerful constraint handling language based on logic programming. Constraint handling techniques allow the solution of complex combinatorial problems with a fraction of the development effort required for design and maintenance of conventional systems.

The paper is structured as follows: we begin with a description of the stand allocation problem and the constraints which are taken into account. We then list design objectives which were defined for the APACHE system. In section 5, we discuss the general systems architecture and the information/data flow of the program. The user interface forms a central part of the complete design. It is covered in section 6. We then describe the problem solver, which allows a flexible, automated scheduling using the constraint programming language CHIP. Key aspects of the APACHE approach are then evaluated in section 8. Several other approaches to stand allocation have been proposed. We compare them to the constraint based approach in section 9.

3 Problem Description

APACHE solves the problem of stand allocation for large airports. Planes arriving at the airport must be parked at one of the available stands, either in contact at one of the terminal gates, or on the apron, the field off the terminal. If the plane is parked in contact, passengers can disembark or board the plane directly from the terminal through a jetty. If the plane is parked on the apron, passengers must be transported to the plane either by bus or travel lounge. A solution to the stand allocation problem has to find an assignment of all arriving and departing planes to compatible stands and should take several cost criteria into account.

3.1 Plane Movement

Depending on the time a plane stays at the airport we can distinguish three situations:

The plane stays only for a short time (typically below 3 hours). Then the plane should arrive and depart from the same stand, i.e. it should not be moved between arrival and departure (figure 1a).

If the plane stays somewhat longer, it is possible to move the plane once between arrival and departure (figure 1b).

If the plane stays much longer (typically more than 4 hours) then the plane can be moved twice and may be parked empty on the apron (figure 1c). This way, no terminal gate is blocked during the waiting period with an empty plane.
3.2 Constraints

Any solution to the stand allocation problem must take all existing hard constraints into account, i.e. it can not park two planes at the same stand at the same time, and should satisfy as many soft constraints, user preferences, as possible. At the same time, the solution should also respect certain cost constraints on resource use or scheduling quality.

We will now discuss this classification of constraints in more detail.

3.2.1 Hard Constraints

These constraints form the core of the system. Any solution must respect such conditions. They can therefore be used actively to prune the search space for possible solutions. Examples of hard constraints are:

**Restricted Parking** Not all types of aircraft can be parked at every gate. Sometimes a parking place can be used either by one big plane, or two smaller ones. A parking position may temporarily block other places if it is used. The type of jetty installed also restricts which types of aircraft are compatible with a stand.

**Customs Status** Often gates are restricted for only domestic or international flights. Others may be able to operate both domestic and international flights. Sometimes, certain groups of gates can be switched between international and domestic status, but only together and only during certain periods of the day.

**Special Flights** Some flights are restricted to specific gates for security reasons or passenger comfort.

3.2.2 Preferences

Preferences are soft constraints. The system should respect them, if possible. Violations of some of the soft constraints can be accepted in a solution.
**Preferred Terminals** Arriving or departing flights have preferred terminals. A plane should park in contact at one of the contact gates of its preferred terminal. If this is not possible, it is allowed to park it on the apron of this terminal. If this is also not possible, it can be parked on the apron of another terminal.

**Largest First** Planes with many passengers should be assigned to contact gates with preference. This is a simple heuristic to improve the number of passengers which can board directly.

### 3.2.3 Cost Model

The cost model takes two conflicting factors into account.

**Passenger Movements** A main criteria judging the quality of a schedule is the number of passengers who can board directly via a jetty. A high overall percentage improves passenger comfort, and also decreases costs and delays due to bus transfer.

**Towing Resources** The number of movements of planes can be restricted on a global or on a hourly basis. Such towing operation are an important cost factor, since they do not only require resources, but also add to the limited amount of ground traffic. On the other hand, towing of planes can free up space on the terminals, so that more passengers can board directly.

Since the different cost factors are contradictory, we can not satisfy both in an optimal way at the same time. Currently, special emphasis is given for passenger flow, while the towing resource limit is handled as a hard constraint.

### 4 Design Objectives

We now want to describe some of the design objectives behind the APACHE system. For this, we first have to define the intended use of the program.

#### 4.1 Intended Use

The system can be used for two different purposes:

An **online scheduling system** must handle the actual scheduling during a day of operations. The system must react quickly to all changes in flight plans and resource availability. For this, stability and fast reaction time are most important.

Another use of the program is as a **planning tool** to study the impact of changes in the airports configuration. This allows to compare different approaches to handle for example increased flight operations or new terminals. This use of APACHE requires the flexibility to change the system quickly to experiment with new situations.
4.2 Performance Criteria

From the intended use, we can infer several key design objectives. The system has to be

- flexible
- interactive
- incremental
- integrated
- efficient

We now discuss each point in more detail:

4.2.1 Flexibility

The configuration of airports is constantly changing. New terminals or gates may become available. New aircraft types require special resources. A decision support tool must be adaptable to these new situations in a very short time.

4.2.2 Interactive Tool

The stand allocation problem must take many different constraints into account. Some of these conditions are contradictory, i.e. they can not be achieved at the same time. No completely automated system can hope to handle all these conflicts by itself. Experienced human controllers have to control the system. On the other hand, computationally expensive or error prone tasks should be taken over by the computer. The aim is a decision support system, not a completely automated scheduler. A graphical environment allows the user to interact with the system in an intuitive way.

4.2.3 Incremental Solutions

The stand allocation is not a static problem, which can be solved once for a whole day. Changes in flight plans, delays or cancelations, require the constant updating of the proposed plan. On the other hand, the planning should not be changed at random. Only changes which are required by new constraints or which can improve the quality of the schedule should be considered.

4.2.4 Integration

The data for the stand allocation problem are usually available in other parts of the IS (Information Systems) environment. The results of the planning must be forwarded to other parts of the environment. This requires a tight integration into existing information processing systems.
4.2.5 Efficiency

If the program is to be used as an interactive tool, it must react to changes in the data very quickly. A rescheduling time below one minute was considered to be acceptable. This can be guaranteed by the APACHE system.

5 Systems Architecture

In previous sections we have described the problem formulation and the design objectives. In this section we will now discuss the overall architecture of APACHE. We present two views of the system. One is based on the physical components of the system, the other is concerned with the flow of information through the program.

5.1 Physical Model

We have split the functionality of the APACHE system into two main components. One part is the actual problem solver, a constraint based program, which computes the solution to the stand allocation problem. The other part contains the user interface, which controls the scheduler and provides the connection to the user and other parts of the system. Both parts are written in CHIP, a constraint logic programming language.

Fig 2: Physical Model
The overall architecture of the APACHE system is shown in figure 2. The interface and the scheduling program run on UNIX workstations. It is possible, but not required, to distribute the programs over a local area network (LAN). Using network services, access to remote databases on mainframes via wide area networks (WAN) or to other parts of the airport information systems can be provided. Multiple display stations can be connected to one APACHE scheduler. It is also possible to use high end personal computers for the display of scheduling information.

5.2 Data Flow

The flow of information through the APACHE system is shown in figure 3.

![Data Flow Model](image)

**Fig 3: Data Flow Model**

The main input sources are:

- **flight database**: This database contains information about flight plans, delays, cancellation of flights, etc. The information is usually obtained from an existing database system.

- **airport configuration**: This input describes the layout of the airport, which gates are compatible with which flights, etc.

- **resource events**: Events on this input describe down-time of jetties, resource restrictions and other changes in the current state of the airport's facilities.

- **policy parameters**: This file describes global parameters under which the system operates. It covers safety margins, or passenger limits. Changes in this input allow planning experiments to evaluate how the assignment is affected.
• cost objectives: This covers hard limits for resource consumption or optimization criteria for a search among alternative solutions.

The scheduler produces a parking assignment which is also fed back to the scheduler for incremental modification.

6 User Interface

The APACHE system is completely embedded in an interactive user environment. Based on the industry standard Xwindow system™, it allows a direct manipulation of the parking information.

Several views on the data are available in different windows:

6.1 Gantt Chart

A Gantt chart (figure 4) displays the parking assignment over a period of time (typically 2 days). The user can zoom into the plan for more details. By clicking onto objects, flight information and constraints become available for manipulation. Gate restrictions can be modified online. Possible alternative parking places for planes can be shown. After rescheduling, the necessary modifications of the plan are highlighted for emphasis.

![Fig 4: Gantt chart display](image)
6.2 Parking View

The current parking situation is displayed in another window (figure 5). This provides another interface to the scheduling data. Information on flight arrival/departure, etc can be obtained by graphical interaction. A textual display of pending events (arrivals, departures) complements this view of the current parking state.

Fig 5: Parking view at specific point in time

6.3 Resource View

The limits on the use of tugs and their projected use can be displayed as well (figure 6). The resource constraints can be modified by directly manipulating the display.

Fig 6: Resource limit and actual use
7 Problem Solver

The actual scheduler is a program written in CHIP. CHIP was developed at the European Computer-Industry Research Centre (ECRC), located in Munich. It is a constraint logic programming language which has been designed to solve complex decision making problems. CHIP is an advanced programming language which combines powerful modeling capability, flexibility and versatility of advanced Artificial Intelligence tools with the efficiency of conventional algorithmic approaches.

CHIP brings together techniques from Logic Programming, Artificial Intelligence, Mathematics and Operations Research to allow powerful symbolic and numerical constraint manipulation. It combines in one system the following three computation domains: finite domains (for discrete combinatorial problems), boolean terms (for propositional calculus problems) and rational terms (for linear programming type applications). It is used mainly for developing decision support tools in the areas of

- Planning and manufacturing
- Distribution and Logistics
- Economics and Finance

In APACHE, we are using several of the constraint mechanisms of the CHIP system to model the assignment problem. Using data read from the flight database and the different description files mentioned above, the program generates the constraints which describe the problem. The hard constraints are specified first, thus restricting the search space for possible solutions. The user preferences are used to further restrict the search space. The program may decide to ignore a preference if it can not be satisfied together with the other conditions.

A set of heuristics is used to find solutions with good cost values. This use of heuristics is very important, since it allows to incorporate knowledge available for manually solving this problem. Such heuristic information is easily expressed with rules in the CHIP language.

We prefer to implement the problem in a very high-level language like CHIP rather than an implementation in a language like C. From a programmer's point of view, this approach has several important advantages:

- Development time is short. Different modelling techniques and implementations of problem specific constraints can be tested without much work.

- The scheduling program is modular, rather small and concise. The formulation of the constraints is very close to a mathematical framework.

- The program can be understood and maintained more easily than conventional programs. This results immediately from the high-level problem description and the expressive power of the CHIP system.
Changes and extensions can be provided without rewriting major parts of the program. This is especially important for problems like stand allocation which have continuously changing requirements.

Due to an optimized implementation of the constraint solver, the program reaches a very good performance.

8 Evaluation

The APACHE system has been tested on actual flight data from a major European airport with around 400 flights per day. The existing stand allocation system consists of a mainframe package running overnight, which is manually updated during the day. Considering the number of changes in flight operations occurring during a day, it is clear that the original plan does not reflect reality at the end of the day. Due to the high cost of software maintenance, the package does not handle all new constraints, which are due to changes in the airport configuration. These changes are considered manually.

The APACHE program in comparison shows several advantages:

- All constraints can be considered. The flexibility of the system allows to express many different constraints quite easily. No manual post processing is required.
- All planes are assigned to stands by the automatic scheduler.
- A better terminal utilization is obtained. In comparison with the mainframe package, an increase in passenger flow through contact gates has been observed.
- The scheduler can be run interactively. This allows keeping track of all changes during the day without manual rescheduling. The required rescheduling time of one minute can be guaranteed.
- The APACHE system reacts to new requirements in a flexible manner. For example, for adding a new terminal building the necessary changes and allocation strategies for the APACHE system can be integrated within one week.

9 Comparison

Several other approaches to the stand allocation problem have been reported. We will discuss main ideas of such systems and compare them to the constraint based approach discussed here.

9.1 Operations Research Packages

One possibility is the use of Operations Research packages to model the problem as a linear programming or integer linear programming prob-
lem. This allows to use well known techniques implemented in stable, robust programs.

This approach has several disadvantages. The problem must be formulated in terms of the solution method used, not in a natural way. Quite often, some constraints cannot be expressed at all in such a framework. Reformulation can also lead to a huge increase in the number of problem variables and a resulting inefficiency of problem resolution. Perhaps the biggest disadvantage is that heuristics and strategies known for solving the application problem cannot be easily integrated. This important source of information can thus not be used.

9.2 Dedicated, conventional programs

It is quite possible to develop a bespoke system for stand allocation in a conventional language. The main obstacle is the lack of flexibility in this approach. To obtain an automated stand allocation, many of the constraint handling algorithms provided in CHIP would have to be reimplemented in some way in a conventional language. The development effort for such a system will be significantly higher than for the APACHE system, which uses general techniques already provided in the constraint language CHIP.

9.3 Rule based systems

Rule based systems allow to capture heuristics and strategies for solving difficult problems like stand allocation. They are based on the assumption that this allows to emulate human planning behaviour for such problems. The major problem is that integrating constraint handling features into rule based application can be very complex. The required rule system is difficult to define, and not very efficient. Algorithmic parts of the problem can only be specified in an rather awkward manner.

9.4 Constraint based approach

The APACHE system tries to combine several of the advantages of other methods, while reducing the limitations. It is based on a very high-level constraint language, thus providing powerful constraint solving methods in an expressive environment. Heuristics and other strategies can be defined in rules of the underlying logic programming framework. This mix of constraint solving methods with user defined constraints allows for a very flexible system, which can be rapidly developed and extended.

10 Summary

In this paper we have discussed the APACHE system, a decision support tool for airport stand allocation. It provides an automated scheduler, which takes many different constraint types into account. It is based on the constraint logic programming language CHIP, which allows to
develop compact, flexible and efficient problem solutions with very low development cost.

11 Acknowledgements

We would like to thank E. Odier, P.O. Thibault and P. Le Naour of Air France and H. Chevrinais for many interesting discussions and valuable feedback.

CHIP is used under licence from ECRC GmbH, Munich.