Proceedings of the Seventh Int'l Conf. on Artificial Intelligence and Expert Systems Applications,
San Francisco, November, 1995.

CHECK-IN COUNTER ALLOCATION USING GENETIC ALGORITHM

Bryan Kit Wang Yeung & Hon Wai Chun

City University of Hong Kong
Department of Electronic Engineering
Tat Chee Avenue
Kowloon, Hong Kong
email: eehwchun@cityu.edu.hk

ABSTRACT

This paper describes research in developing an airport check-in counter allocation system using a genetic algorithm (GA) approach. The task is to allocate a set of check-in counters for a particular duration such that all the passengers of each departing flight are adequately serviced in time. We have formulated this problem as a two-dimensional placement problem with time and counter as the dimensions. The system was implemented with Koza's genetic programming kernel in Common Lisp. This paper describes how fitness-directed scheduling can be applied to airport check-in counter allocation and the three knowledge representations that we have experimented with.

KEYWORDS

genetic algorithm, constraint-based scheduling, resource allocation, expert system

1. INTRODUCTION

This paper describes research in developing an airport check-in counter allocation system using a genetic algorithm (GA) approach [12]. The main objectives of our research is to design and implement a system to automate and optimise check-in counter allocation in an international airport where check-in counters are centrally allocated. The GA approach [4, 5, 6] was investigated as an alternative to other forms of scheduling, such as constraint-based scheduling [1, 7, 8, 11, 13, 14]. Currently at this airport, the allocation schedule is constructed by hand. This takes several days to more than a week during seasonal planning. Due to increasing traffic, a new system is needed that can automatically create an allocation plan based upon preset rules and criteria to more effectively utilise the somewhat limited check-in counter resources. We implemented a system called CCAS using Koza's Common Lisp genetic programming kernel [9, 10].

CCAS (see Fig. 1) is a planning tool to generate two key output - (1) an allocation Gantt chart for each day of the week, and (2) reports to notify each airline of their check-in counter assignments. The input to CCAS is a master seasonal schedule which lists all the flights for the season and the check-in counter requirement for each flight. Information includes the airline, departure time, destination, passenger load factor, type of check-in, number of counters required, etc.

Since knowledge representation is a key issue in GA work, this is a main focus in our research. We have experimented with three different approaches to representing flights and check-in counters within the GA framework.

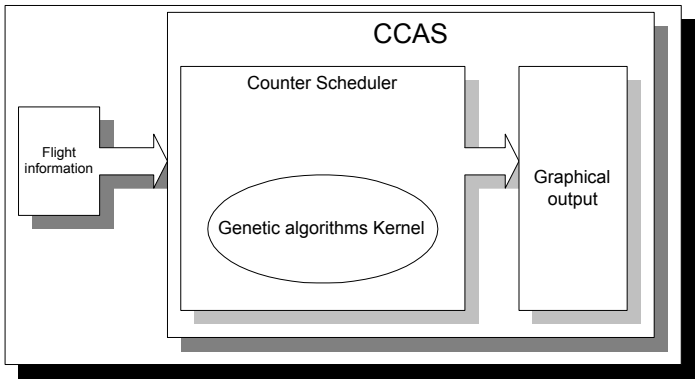

Figure 1. The System Architecture of CCAS.

2. THE GENETIC ALGORITHM APPROACH

In GA, populations of individuals are genetically bred. This breeding is done using the Darwinian principle of survival [2, 3], reproduction of the fittest, and a genetic recombination (crossover) operation. An individual that solves (or approximately solves) a given problem may emerge from this combination of Darwinian natural selection and genetic operations. In CCAS, individuals represent alternative allocation plans for one day - a two-dimensional Gantt chart of check-in counters versus time where check-in counter assignment blocks for given flights are randomly placed. The fittest individual is the "best" allocation plan for that day.

Fitness measure - Each individual in the population is measured in terms of how well it performs in the particular environment. This measure is called the fitness measure. The CCAS fitness measure indicates the overall effectiveness of the allocation plan and how well allocation criteria are met (see Section 3 for details).

Reproduction - As the first step in GA, a population of individuals is randomly created - this is generation 0. Unless the problem is so small and simple that it can be easily solved by blind random search, the individuals in generation 0 will have very poor fitness. Nonetheless, some individuals in the population will turn out to be somewhat fitter than others. These differences in performance are then exploited. The fittest individual will be selected for reproduction.

The Darwinian principle of reproduction, survival of the fittest, and the genetic operation of sexual recombination (crossover) are used to create a new offspring population of individuals from the current population. The reproduction operation involves selecting, in proportion to fitness, an individual from the current population, and allowing it to survive by copying it into the new population. In CCAS, good allocation plans will continue to survive, reproduce, and hopefully result in better offspring.

Crossover - The genetic process of reproduction between two parental individuals is used to create new offspring from two allocation plans selected in proportion to fitness. Intuitively, if two allocation plans are somewhat effective in solving the check-in counter problem, then some of their check-in counter assignments probably good. By recombining randomly chosen assignments of somewhat effective allocation plans, we may produce new allocation plans that are even more optimised in solving the problem.

After the operations of reproduction and crossover are performed on the current population, the population of offspring (i.e. the new generation) replaces the old population

(i.e. the old generation). Each individual in the new population is then measured for fitness, and the process is repeated over many generations.

Algorithm - The GA used by CCAS (see Fig. 2) can be summarised to solve the check-in counter allocation problem by executing the following three steps:

Step 1: Generate an initial population of allocation plans (GA individuals) consisting of random assignments of check-in counters to flights.

Step 2: Iteratively perform the following sub-steps until the termination criterion has been satisfied:

(a) Evaluate each allocation plan and assign to it a fitness value according to how well it fulfils the allocation criteria.

(b) Create a new population of allocation plans by applying the following two primary operations. The operations are applied to allocation plans in the population chosen with a probability based on fitness.
   (i) Copy existing allocation plan to the new population.
   (ii) Create new allocation plans by genetically recombining randomly chosen parts of two existing plans.

Step 3: The best allocation plan that appeared in any generation (i.e. the best-so-far individual) is designated as the result of CCAS.

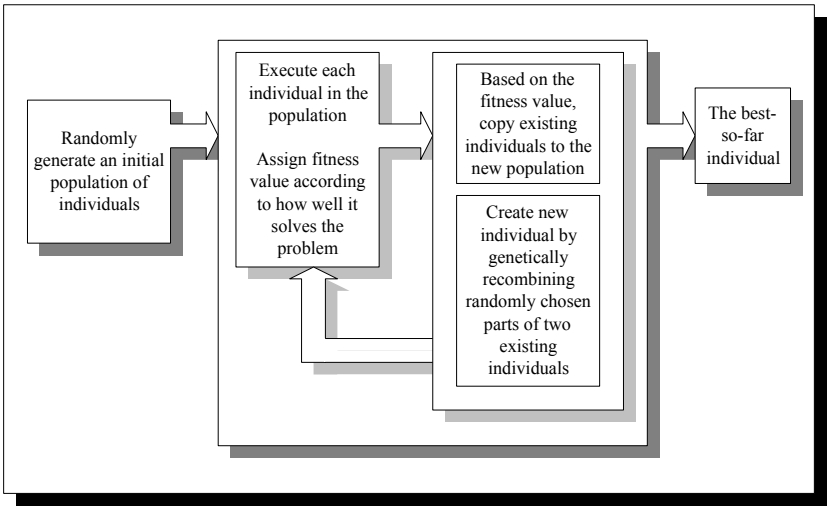

Figure 2. Flowchart of GA used by CCAS

3. CCAS FITNESS MEASURE

Each GA individual in CCAS is represented as an two dimensional plan in which check-in counters act as the vertical axis versus the time. In the Fig. 3 example, J1, J2, J3 and etc., are the check-in counters in the airport and the horizontal axis is the regular opening time of all counters.

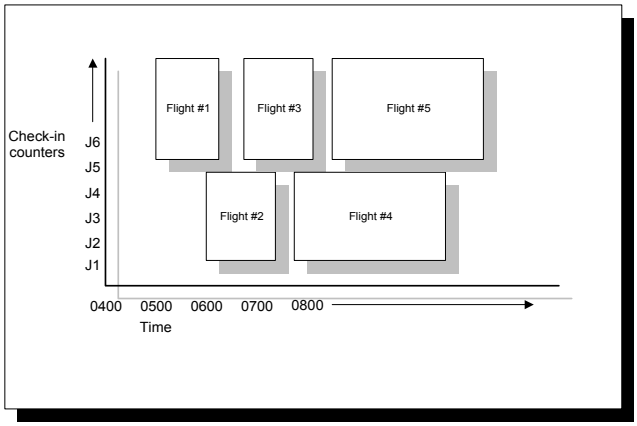

Figure 3. Logical representation of an GA individual in CCAS

CCAS allocates and optimises check-in counters allocation by fulfilling the following allocation rules and criteria. Firstly, the system should allocate check-in counters to each flight in accordance to the number of counter required by the flight as well as the opening and the closing times of the check-in counters for a flight. Secondly, the system should satisfy the requested number of counter of each flight as much as possible. Thirdly, the system should consider the passengers arrival profile whenever conflicts happen.

Conflicts occur whenever there are overlapping of blocks in the allocation plan. It is a time conflict in which a counter is assigned to more than one flight at the same time. Very often, the number of check-in counter is inadequate to satisfy the demand from the flights, especially in the peak hour of the airport. Therefore, overlaps may be unavoidable. Statistics are provided by the airport to indicate when passengers will usually arrive to check-in during the check-in time duration of a flight. Very often, the peak is somewhere in the middle of the check-in duration. Therefore, conflicting counters are allowed in periods where less passengers are expected to check-in. The number of check-in counters to be assigned to the overlapping flights will be reduced during the overlapping period.

Fitness measure is the driving force behind GA; it defines which individual can survive and successfully reproduce. In *CCAS*, *fitness* is measured in terms of the number of *overlaps* found in the allocation plan (individual). In other words, it is desirable for an individual to have less number of *overlap* of blocks of flight so as to satisfy the allocation rules and criteria mentioned before.

In *CCAS*, two types of *overlap* are defined: *the homogeneous overlap* and the *heterogeneous overlap*. *Homogeneous overlap* is the time conflict between flights of the *same* airline companies while *heterogeneous overlap* is the time conflict between flights of *different* airline companies. In airport, it is common for flights of the same airline to share the same counters so as to spare more counters for the use by other flights of different airline especially in the peak hours of the airport. However, it is impossible for the flights of different airline to commonly use the same counters allocation. Therefore, the goal of the optimisation of the check-in counter allocation is to, whenever there exists unavoidable overlap in the individual, minimise the heterogeneous overlap by allowing minimum amount of the homogeneous overlap in the individual.

4. THREE CCAS REPRESENTATIONS

Representation is a key issue in genetic algorithm work because GA directly manipulate a coded representation of the problem and the representation scheme can severely limit the window by which a system observes its world.

In CCAS, an individual is a two dimensional plan, with check-in counters plotted versus time, into which blocks of flight are placed. In GA, an individual is randomly created according to pre-defined structure of the representation. Therefore, apart from the GA kernel, a *counter set* is defined so that elements in the set are chosen randomly to build the individual. Obviously, the *counter set* is the set of all check-in counters to be used. Three encodings of the CCAS representation were tried and evaluated.

4.1. Version 0

When an individual was encoded, apart from the *counter set*, a *flight set* was also defined to contain all flights to be scheduled. An individual was then created by randomly choosing flights from the *flight set*. In accordance with the number of counter required by that flight, counters from the *counter set* were assigned to the flight at random. However, the problem of this approach is that duplicated flights would appear in the same individual which is illegal since no flight should be scheduled more than once in an individual. It is caused by the random assignment of flights in the *flight set*.

4.2. Version 1

To amend the deficiency of Version 0, Version 1 does not use *flight set*. Instead, flights to be scheduled are stored in an array. Like Version 0, according to the number of counter required by the flight, counters are assigned to the flight randomly from the *counter set*. Under this approach, only the counters are chosen at random; flights are scheduled one at a time. Nonetheless, when using Version 1, additional function is required to check the adjacency of check-in counters assigned as one of the *fitness*-value assignment functions. Certainly, the search space is very large for this encoding scheme which results in inefficiency and large time consumption.

4.3. Version 2

To improve the performance of Version 1, Version 2 was implemented. Version 2 differs in that no matter how many counters are requested by the flight, only one counter is chosen randomly at first. Automatically, consecutive counters will be assigned to the flight by CCAS. Under this approach, the adjacency of the counters is ascertained. For instance, counter ‘J6’ is chosen to ‘Flight #1’. Assuming that four counters are required in ‘Flight #1’, automatically, three consecutive counters after ‘J6’, namely, ‘J7’, ‘J8’ and ‘J9’ are assigned to the ‘flight #1’. This scheme greatly reduce the search space of the problem and, by all means, improve the efficiency of the system.

5. CONCLUSION

A check-in counter allocation system using genetic algorithms was successfully designed and implemented. It is used to amend the deficiency of the current manual approach by providing automation and optimisation for check-in counter allocation and a graphical user interface. In applying genetic algorithms, fitness-directed scheduling was derived and was used to achieve the goal of optimisation. Moreover, three knowledge representations was tried and evaluated.

REFERENCES

1. Atabakhsh, H., "A survey of constraint based scheduling systems using an artificial intelligence approach", *Artificial Intelligence in Engineering*, 1991, Vol 6, No 2,
2. Booker, B., D.E. Goldberg and J.H. Holland, "Classifier Systems and Genetic Algorithms", *Artificial Intelligence* 40 (1-3):235-282, September 1989,
3. Davis, Lawrence(editor), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991,
4. Davis, Lawrence, editor, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann, 1989,
5. Filho, Jose L. Ribeiro, Philip C. Treleaven, and Cesare Alippi, "Genetic-Algorithm Programming Environments", *IEEE Computer* 27(6):28-43, July 1994,
6. Goldberg, David E*., Genetic Algorithms in Search, Optimization, and Machine learning*, Addison-Wesley, Reading, MA, 1989,
7. Kempf, Karl, Claude le Pape, Stephen F. Smith and Barry R. Fox, "Issues in the Design of AI-Based Schedulers: A Workshop Report", *AI Magazine*, 11(5):37-46, 1991,
8. Kempf, Karl, Bruce Russell, Anjiv Sidhu and Stu Barrett, "AI-Based Scehdulers in Manufacturing Practice: Report of a Panel Discussion", *AI Magazine*, 11(5): 46-55, 1991,
9. Koza, John R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992,
10. Koza, John R., *Genetic ProgrammingII: Automatic Discovery of Reusable Subprograms*, MIT Press, 1994,
11. Kumar, Vipin, "Algorithms for Constraint-Satisfaction Problems: A Survey", *AI Magazine*, 12(2):32-44, 1992,
12. Srinivas, M. and Lalit M. Patnaik, "Genetic Algorithms: A Survey", *IEEE Computer* 27(6): 17-26, July 1994,
13. Steffen, Mitchell S., "A Survey of Artificial Intelligence-Based Scheduling System", Institute of Industrial Engineers, *1986 Fall Industrial Engineering Conference Proceedings*, 1986.
14. Suresh, V. and Dipak Chaudhuri, "Dynamic scheduling - Survey of research", *International Journal of Production Economics*, 32(1993) 53-63,