Published in the Tenth International Conference on
Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, June 1997

CONSTRAINT-BASED RESOURCE ALLOCATION FOR AIR CARGO TRANSFER PLANNING

Hon Wai Chun
Department of Computer Science, City University of Hong Kong,
Tat Chee Avenue, Kowloon, Hong Kong
email: andy.chun@cityu.edu.hk

ABSTRACT

In this paper, we documented results produced by a prototype resource allocation system that performs transfer planning for an air cargo handling facility using a constraint-based approach. The key problem is to decide, for each export and import air cargo container, which transfer deck should it be placed at before or after transfer from the aircraft, through which path within the building facility should it be transferred and at which time, and how the empty container should be transferred out to or from the storage area. This resource allocation problem is modelled as a constraint-satisfaction problem (CSP) in this research. The key objective is to optimise the use of the air cargo handling facility, to minimise any delay to aircraft departure, and to fulfil customer service commitments. Another objective is to be able to perform reactive scheduling if flight arrival or departure time has changed, and if the number and type of containers have changed. This paper describes how this problem can be modelled, the constraints that are involved, and the scheduling algorithm used.

INTRODUCTION

This paper describes a constraint-based model and algorithm for an air cargo transfer planning prototype system. This system determines how each export or import air cargo container, or *unit loading device* (ULD), is handled before or after it is taken to or from the aircraft. Several resources must be scheduled for each ULD – the conveyor belt, the hoist or crane, the transfer deck on the ground floor, the transfer times, and the transfer path for the empty ULD. Since resources are valuable and limited within the air cargo handling facility, they must be allocated efficiently taking into consideration all the necessary constraints. Constraints such as the type and priority of a ULD, the arrival or departure time of the flight, the availability and capacity of the resources, and the service commitments to the clients of the air cargo handling company. This research was performed using flight data from the Hong Kong International Airport; one of the busiest international airport in the world.

Our research focuses on designing and developing a constraint-based resource allocation algorithm [PUGE94b] that can assist the human planner by generating a detailed transfer plan for each export and import ULD. This plan will allow the human planner to estimate resource requirements such as the number of workers needed to build up or break down containers at different times of the day. Currently, due to the complexity of the problem, a human planner is not capable of producing a long-term transfer plan that is detailed down to the ULD level.

Figure 1 is a diagram of the simplified air cargo handling facility that was used as the test case for this research. Although the building structure is fictitious, it was modelled after and contains all the essential characteristics of a real air cargo handling facility. For simplicity, the building has three main floors – the workstation floor used for building up and breaking down of containers, the ground floor to take or received the container to or from the aircraft, and the empty container floor to store empty ULDs. The building contains three main transfer paths – the stacker crane which is used for long-term storage, the outer hoist for normal traffic, and the inner hoist for high priority or late cargo. Not shown in Figure 1 is the floor that is used to store empty ULDs. For export, an empty ULD must first be transferred to the workstation floor for container build-up. For import, after ULD break-down, the empty ULD must be transferred to the empty ULD storage floor. The transfer planning system will also schedule the transfer of these empty ULD.

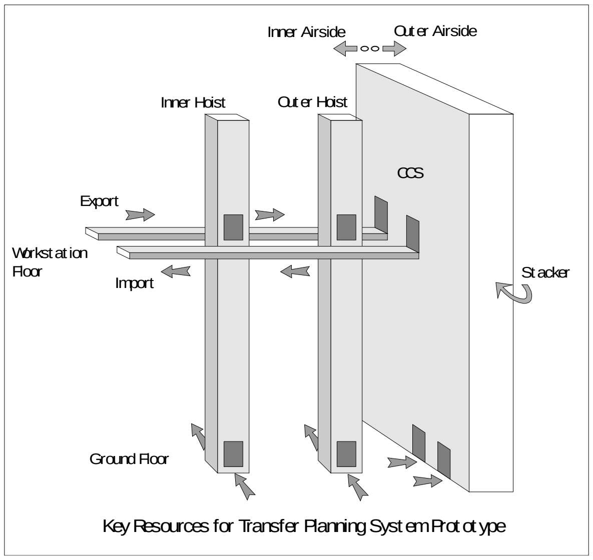

Figure. 1 The simplified building structure used by the transfer planning system.

Figure 2 is a floor plan of the simplified workstation level used by our transfer planning system. The key resources are the two conveyor belts, the two hoists, and the stacker crane. In our simplified building, there is one conveyor belt for import and one for export. Each conveyor belt can access the two hoists and stacker crane. The transfer rate for each resource may be different.

Figure 3 is a diagram of the simplified ground floor layout used by the transfer planning system. There are seven transfer decks that can be used for export containers and three transfer decks for import containers. However, two of the three import transfer decks feeds to both the outer hoist and the stacker crane. A key constraint is that each transfer deck can only be used by containers of the same flight at any one time-period. This is to avoid potential human error in transferring a container to a wrong flight.

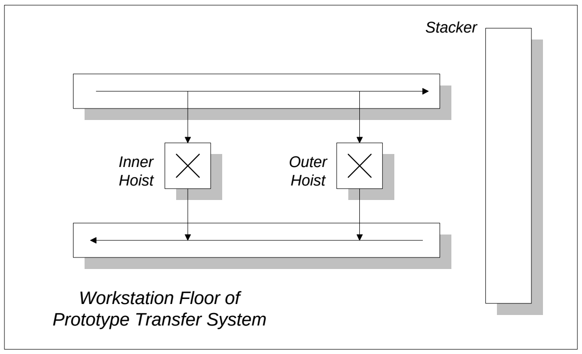

Figure. 2 The workstation level floor plan used by the transfer planning system.

The input to the transfer planning system is a flight schedule that contains the airline code, flight number, flow (either export or import), the scheduled time of arrival or departure, and number and types of ULDs in the aircraft.

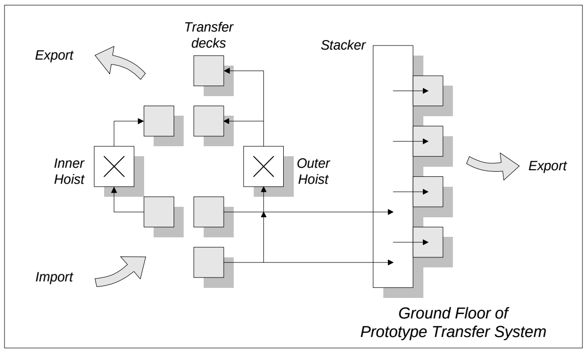

Figure. 3 The ground level floor plan used by the transfer planning system.

Our constraint-based scheduling algorithm allocates five main resources to each ULD - (1) a transfer path within the air cargo handling facility, (2) the transfer time, (3) the transfer deck on the ground level, (4) the empty ULD transfer path, and (5) the empty ULD transfer time.

THE OBJECT-ORIENTED CONSTRAINT-BASED MODEL

To solve air cargo transfer planning, we represented the resource allocation task as an object-oriented [LEPA93] constraint-satisfaction problem (CSP) [KUMA92]. Although CSP or constraint-programming has a relatively long history [STEE80], with constraint language extensions found in Prolog [COLM90, VANH89], and Lisp [SISK93], it is only recently that constraint-programming became more popular with the availability of the ILOG's C++ class libraries [PUGE94a]. This library provided a very efficient and clean implementation of constraint-based programming features in a conventional language. It also integrated constraint programming with object-orientation. Our scheduling algorithm was implemented using the ILOG C++ class libraries.

In general, any scheduling and resource allocation problems can be formulated as a constraint-satisfaction problem (CSP) which involves the assignment of values to variables subjected to a set of constraints. CSP can be defined as consisting of a finite set of *n* variables $v_1, v_2, ..., v_n$, a set of domains $d_1, d_2, ..., d_n$, and a set of constraint relations $c_1, c_2, ..., c_m$. Each d_i defines a finite set of values (or solutions) that variable v_i may be assigned. A constraint c_j specifies the consistent or inconsistent choices among variables and is defined as a subset of the Cartesian product: $c_j \subseteq d_1 \times d_2 \times ... \times d_n$. The goal of a CSP algorithm is to find one tuple from $d_1 \times d_2 \times ... \times d_n$ such that *n* assignments of values to variables satisfy all constraints simultaneously.

The constrained variables used by the transfer planning scheduling algorithm are represented as attributes of a *ULD Assignment* class. Figure 4 is a Unified Modelling Language (UML) class diagram with the key domain classes used by the transfer planning system. There are seven main domain classes:

- **Flight** - Each instance of the *Flight* class represents an individual flight. Attributes of this class store information loaded from the flight schedule. In particular, there are two types of flight -- export and import.

- **ULD** - Each *ULD* object represents one single ULD in a flight. Attributes of this class include the type of ULD, the priority, and the ULD identity number.

- **Flight Assignment** - This class contains resource assignment information for one particular flight. It contains a reference to a flight object and lists of ULD assignment objects. There are three types of ULD assignments, depending on the priority and type of ULD.

- **ULD Assignment** - This class contains the resource assignment information for one ULD. All the constrained variables used by the CSP scheduling algorithm is stored within objects of this class. This includes a constrained variable to represent the transfer path the ULD should take. The domain of this variable includes all the possible transfer paths that are available to this type of ULD. There is a constrained variable to represent the transfer start time. The domain of the start time depends on whether the ULD is an export or import ULD and will range between an earliest and latest time. There is a constrained variable to represent the transfer deck to place the ULD. The domain of the transfer deck variable depends on whether the ULD is an export or import ULD. There is also a constrained variable to represent the transfer path for the empty ULD, and a constrained variable to represent the empty ULD transfer time.

- **Resource** - Each *Resource* subclass has a throughput attribute. The subclass includes the *Transfer Path* and the *Transfer Deck*. Objects of this class keep track of resource availability.

- **Transfer Deck** - Each instance of the *Transfer Deck* class represents one transfer deck on the ground level.

- **Transfer Path** - Each instance of the *Transfer Path* class represents one transfer path within the air cargo handling facility.

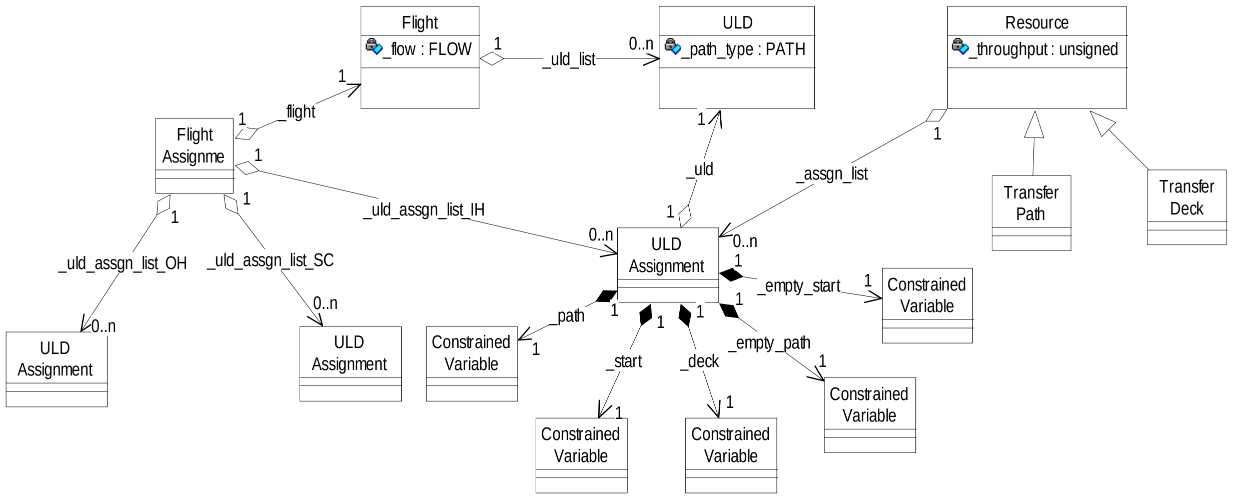

Figure. 4 The UML class diagram of some of the classes used in the transfer planning system.

TRANSFER PLANNING CONSTRAINTS

Some of the constraints that are considered by the transfer planning system were mentioned in the previous sections. The following are other constraints that are also considered by the scheduling algorithm:

- **No resource conflict -** This is of course the most basic constraint. Resources assigned to different ULDs cannot conflict with each other. For example, the same hoist cannot be used to transfer two different ULDs at the same time.

- **Temporal ordering -** The sequence which ULDs are dispatched are sequentially ordered for two reasons. First, for export ULDs, dispatch from the stacker crane is performed before normal ULDs and urgent ULDs. The second reason to order the ULDs is to reduce the search time by eliminating equivalent redundant solutions.

- **Empty ULDs transfer times -** For export flights, an empty ULD must be transferred to the workstation floor before the earliest transfer start time. Empty ULDs from import flights can be transferred after the latest transfer completion time.

- **Number of allocated transfer decks -** The number of transfer decks to be allocated to a flight will depend on the total number of ULDs for that flight.

- **Transfer deck rule -** This is a constraint to encode the transfer deck rule which forces the transfer deck to process only one flight's ULD at a time.

The following are constraint diagrams developed by the author to illustrate temporal constraints. The horizontal line ranges from the earliest transfer start time (ETST) to the latest transfer completion time (LTCT). This line also represents the domain of the temporal constrained variables, shown as upward arrows. Arrows sliding along this line is analogous with variables taking different values from a domain. Each temporal constrained variable represents the transfer start time of one ULD. The rectangular box represents the duration of the transfer that depends on the selected transfer path. The "less than" symbol within the circle represents the temporal constraints.

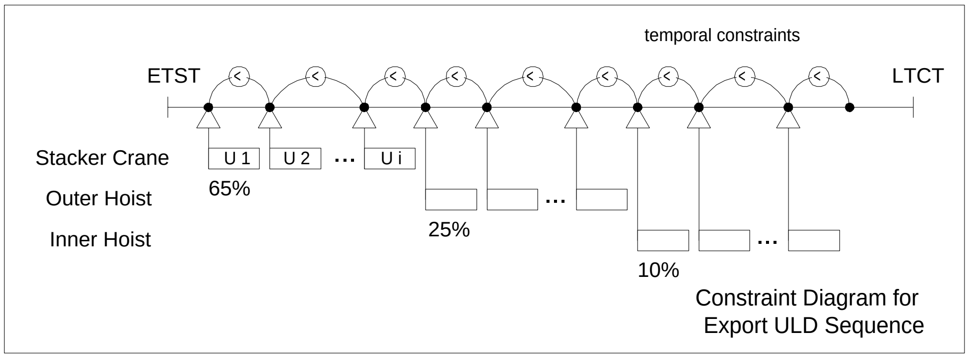

Figure. 5 Constraint diagram for export ULDs.

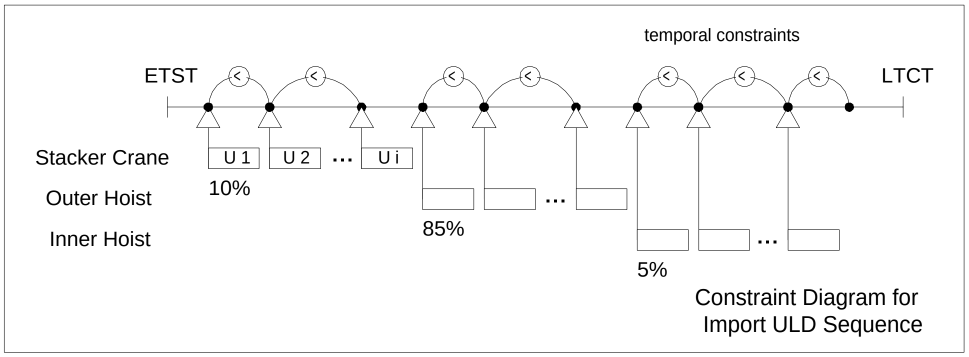

Figure. 6 Constraint diagram for import ULDs.

For export ULDs within the same flight, all ULDs are temporally sequenced in a chain as illustrated in Figure 5. The temporal constraints for ULDs of the same type are used as "redundant constraints" to reduce search time. However, for import flights, ULDs are only temporally sequenced within the same transfer path, as illustrated by Figure 6.

The percentage in the diagram indicates the statistical percentage of ULDs of different type. For example, for export flights, around 65% of the total ULDs for any one flight are taken from long term storage, around 25% are normal ULDs, and around 10% are late arrival ULDs.

THE TRANSFER PLANNING SCHEDULING ALGORITHM

The scheduling algorithm developed for this research consists of a collection of goals. The CSP search algorithm tries to solve each goal sequentially. The main goal is defined as follows:

```
void main() {
  …
  IlcSolve(TransferPlanSearch());
  …
};
```

IlcSolve is the built-in CSP search algorithm provided by the ILOG Solver C++ class library. TransferPlanSearch is the main goal and is defined as follows:

```
ILCGOAL0(TransferPlanSearch) {
  // Export flights have higher priority.
  IlcAnd(AllocateExportULD(),
         // then import flights
         AllocateImportULD(),
         // store empty ULDs away
         AllocateImpEmptyULD(),
         // retrieve empty ULDs for export
         AllocateExpEmptyULD()
         ) ;
}
```

TransferPlanSearch itself consists of four other goals that are ordered according to priority – export has highest priority since flight departure time should not be delayed, next is import flights, and then the empty ULDs. ILCGOAL0 is a macro to define goals using ILOG Solver. As an example, the first goal of allocating resources to export ULDs can be defined as follows:

```
ILCGOAL0(AllocateExportULD) {
  …
  ULDAssignment* pU ;
  while (pU=(ULDAssignment*)iter()) {
    // assign transfer path for ULD,
    IlcAnd(AssignTransferPath(pU),
          // then the transfer time,
          AssignTransferTime(pU),
          // the transfer deck,
          AssignTransferDeck(pU),
          // and then fire demon to check
          // additional constraints
          CheckExpConstraints(pU)
          );
  }
}
```

Besides the transfer plan generation, our prototype also performs reactive scheduling to handle to following two types of dynamic changes:

- **Change in number of ULD** - The number of ULDs in a flight might not be accurate and may change. Changing this number will cause the transfer planning system to recompute the number of ULDs in each category, and reschedule all affected ULDs.

- **Equipment breakdown** - If a resource breaks down, the transfer planning system reschedules all the ULDs that were previously assigned to that resource according to constraints.

The transfer planning prototype displays the results of the schedule as a Gantt chart. Each row represents one resource. A resource is either a transfer path, transfer deck, or unassigned transfer path. Each rectangle in the Gantt chart represents the allocation of a resource to a ULD for a period of time.

SYSTEM IMPLEMENTATION

The object-oriented constraint-based [LEPA93] allocation system was implemented in C++ using the ILOG Solver class library [PUGE94a] and the RTL Scheduling Framework developed by Resource Technologies Limited [CHUN96a, CHUN96b]. The graphic user interface that displays the generated schedule as a Gantt chart was developed using C++ graphic components provided by ILOG Views. The system was developed using platform independent coding and can execute within Windows 95/NT or Unix environment. ILOG was chosen since it is the fastest implementation of CSP in the market and, with a C++ environment, the system can interface easily to external systems.

The performance of the scheduling algorithm was an important consideration for this research and a considerable amount of time was spent in improving the system performance. The performance improved from the initial 5 to 6 minutes to half a minute required to schedule one whole day's flight on a Pentium-based 133MHz personal computer. Most of the performance gains are obtained through the addition of redundant constraints.

Our test cases had on the average around 300 ULDs daily and since each ULD requires three resources to be allocated - the transfer path, the empty transfer path, and the transfer deck, the system had to allocate around 900 items total in half a minute - while considering all the constraints defined. Reactive scheduling only takes a few seconds to perform as well. Due to the large number of items to be scheduled, it is currently not feasible for a human to accurately produce a daily transfer plan down to the ULD level.

Reactive scheduling is the ability to react to changes such as changes in aircraft loading, additional flights, flight cancellations, or flight delays. In our system, reactive scheduling only takes a fraction of the time needed by the initial scheduling. Traditionally, reactive scheduling involves regenerating a new schedule all over again. In our approach, the reactive scheduling algorithm reuses the current schedule as much as possible and only makes changes to the original schedule, if absolutely necessary, in order to solve problems caused by unexpected changes.

Although the users would like an "optimal" solution, in reality an "optimised" solution will be sufficient. The reason is that an "optimal" solution is guaranteed to be non-optimal as soon as it is put into use – flights are not always punctual, import and export cargo loading will change, flights might be added, etc. Our approach is to use a scheduling algorithm that generates a heuristically optimised solution and couple that with a quick reactive scheduling component, also using the same heuristics, to deal with last minute changes.

In certain cases, depending on the flight schedule and aircraft loading, the scheduling problem might be over-constrained. In those cases, the system first relaxes constraints related to import ULDs since they are less time critical. In cases where the system is still over-constrained after import constraints are relaxed, the system must then violate export ULD constraints and warn the user. For these cases, the physical cargo handling facility is over its maximum capacity and there is not much a scheduling algorithm can do to improve the situation.

CONCLUSIONS

This paper documents our research in modelling air cargo transfer planning as a constraint-satisfaction problem. The constraint-based scheduling algorithm was tested using air cargo data from the Hong Kong International Airport. The results showed that the scheduling algorithm was able to generate a more efficient schedule in much less time than a human scheduler. Currently, the scheduling task is too complex and time consuming to be produce manually down to ULD precision. Long-term planning is, of course, out of the question.

ACKNOWLEDGEMENTS

Research described in this paper was performed in cooperation with Mr. Albert Pang of Resource Technologies Limited (http://www.rtl.com.hk/~rtl) in Hong Kong.

REFERENCES

[CHUN96a] H.W. Chun, K.H. Pang, and N. Lam, "Container Vessel Berth Allocation with ILOG SOLVER," The Second International ILOG SOLVER User Conference, Paris, July, 1996.

[CHUN96b] H.W. Chun, M.P. Ng, and N. Lam, "Rostering of Equipment Operators in a Container Yard," The Second International ILOG SOLVER User Conference, Paris, July, 1996.

[COLM90] A. Colmerauer, *An Introduction to Prolog III*, Communications of the ACM, 33(7), pp.69-90, 1990.

[KUMA92] V. Kumar, "Algorithms for Constraint Satisfaction Problems: A Survey," In *AI Magazine*, 13(1), pp.32-44, 1992.

[LEPA93] C. Le Pape, "Using Object-Oriented Constraint Programming Tools to Implement Flexible "Easy-to-use" Scheduling Systems," In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling for Manufacturing*, Cocoa Beach, Florida, 1993.

[MACK77] A.K. Mackworth, "Consistency in Networks of Relations," In *Artificial Intelligence*, 8, pp.99-118, 1977.

[PUGE94a] J.-F. Puget, "A C++ Implementation of CLP," In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.

[PUGE94b] J.-F. Puget, "Object-Oriented Constraint Programming for Transportation Problems," In *ILOG Solver Collected Papers*, ILOG SA, France, 1994.

[SISK93] J.M. Siskind and D.A. McAllester, "Nondeterministic Lisp as a Substrate for Constraint Logic Programming," In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, pp.133-138, July, 1993.

[STEE80] G.L. Steele Jr., *The Definition and Implementation of a Computer Programming Language Based on Constraints*, Ph.D. Thesis, MIT, 1980.

[VANH89] P. Van Hentenryck, *Constraint Satisfaction in Logic Programming*, MIT Press, 1989.

[WALT72] D.L. Waltz, "Understanding Line Drawings of Scenes with Shadows," In *The Psychology of Computer Vision*, McGraw-Hill, pp.19-91, 1975.