MACPLAN, a Mixed-Initiative Approach to Airlift Planning

Kirsten Y. Kissmeyer, Anne M. Tallant

The MITRE Corporation
Burlington Road
Bedford, MA 01730

Abstract

This paper describes an object-oriented model for a mixed-initiative airlift planning aid. The system provides a workbench of tools that help airlift planners build effective plans faster. The system maintains consistency among objects in its knowledge base by enforcing constraints. Forward-chaining through domain rules is used to identify problems in the routing network. MACPLAN has improved the productivity of airlift planners by an order of magnitude.

Introduction

Air Force airlift planners develop airlift plans for mobilizing forces in contingency operations. Orders for such mobilizations are given as a series of time-phased force movements. These movement requirements can often number in the thousands. The planners must determine the feasibility of an operation under the constraints of limited numbers of aircraft and airfields to support the move.

MACPLAN is a mixed-initiative, knowledge-based system that helps airlift planners develop resource-effective airlift plans quickly. A mixed-initiative system provides the user with automated help, yet allows the user to drive the problem-solving process. The user can select what parts of a plan to develop himself, and what parts to let the system develop.

MACPLAN provides an object-oriented model of airlift entities, a number of heuristic and algorithmic tools for analyzing a plan, and graphical plan manipulation tools.

Background

The airlift planner's task is made especially difficult due to the overwhelming quantities of data involved in the planning process. This includes information about cargo and passengers, aircraft, operators of aircraft, airfields, and their related timing and constraint factors.

A planner's overall task is to allocate and schedule resources to satisfy potentially thousands of movement requirements. A *movement requirement* is some quantity of passengers and/or cargo that must be moved from one port to another in a specified time window (see figure 1). There are three transportation-related categories of cargo: bulk, oversize, and outsize. Bulk cargo fits on a pallet; oversize is cargo that is too large to fit on a pallet and can only be moved on certain aircraft types; outsize is the largest category of cargo and will only fit on a C-5 aircraft.

Port of Embarkation: Airfield A
Port of Debarkation: Airfield B
Tons Bulk: 30
Tons Oversize: 0
Tons Outsize: 0
Number Passengers: 10
Available to Load Date: Day 0
Earliest Arrival Date: Day 1
Latest Arrival Date: Day 3

Figure 1. A sample movement requirement.

Each class of aircraft has certain characteristics that effect its ability to support the movement. These include its capacities (by cargo type and passengers), range, air speed, takeoff and landing requirements, and average allowed utilization rate. Civilian and military operators supply the aircraft apportioned for the movement. Restrictions prevent operators from using certain airfields, and thereby reduce the number of aircraft that can use the airfield.

Airfields also have a number of characteristics which constrain their use. Cargo and passenger throughput constraints limit the daily amounts of cargo and passengers that can onload or offload at an airfield. Real-world parking and runway characteristics, as well as political considerations, impose additional constraints on airfield utilization in terms of allowable aircraft types and operators.

The Concept of a Plan

The goal of airlift planning is to develop a plan that provides alternative ways to deliver its requirements on time. Flexibility is important since airfields can be closed and aircraft can fail.

We do not use the word *plan* in the conventional sense of a sequence of actions to perform in order to reach a goal (Wilkins 1984). An airlift plan embodies guidelines for airfield and aircraft utilization. The following are examples of plan guidelines:

- Use airfields A and B for refueling and reassignment of military aircraft after mission completion.
- Disallow civilian aircraft at airfield C.
- Use civilian aircraft for the bulk of the passenger loads and route them through group 1.
- From the first to the ninth day of the plan, use 3 747-100s from operator A, and increase the amount to 4 from day 10 to the end of the plan.

The movement requirements and the plan, in the form of these guidelines, then serves as the basis for developing a detailed proposed schedule of airlift movements. As described in the next section, the current approach to developing such a schedule is to run a discrete-event simulation of the operation.

Airlift Planning Prior to MACPLAN

Prior to MACPLAN, airlift planning was primarily a manual process. The only electronic aid was a simulation that executed as a batch job on a computer main frame. The planner developed the plan on paper, and then keyed the information into the computer prior to running the simulation. To do this, they reviewed stacks of computer-generated paper reports, plotted onload and offload airfield locations on a map, and drew out intended routing strategies on paper charts, incorporating enroute stops as necessary to accommodate planes with limited ranges.

The simulation requires that the planner specify routes for each requirement to move from its onload to its offload airfield. To reduce the complexity of the routing network, airfields are grouped geopolitically. In this way, it is only necessary to specify a routing from every requirement's onload airfield group to its offload airfield group. Figure 2 shows the reduction of complexity realized when airfields are grouped in this manner. The dots represent individual airfields, the thin arrows represent requirements that must move, and the ovals describe groupings. The thicker arrows represent the only routes that are necessary to specify routing at the group level. Route distances are then calculated by using the longest airfield to airfield distance between groups.

The simulation operates upon the resource utilization and routing guidelines set by the planner to produce a detailed schedule of movements. The simulation "loads" cargo onto aircraft and "moves" the aircraft from airfield to airfield using a set of heuristics to guide the process. The system chronicles any late arrivals or bottlenecks that occur. After the simulation has completed, planners analyze the simulation's results to see how well their specification of resource use moved the cargo and passengers. If something moved late, or not at all, the planner must figure out why, and what to do about it.

While the simulation produces accurate predictions on the ability of the plan to execute the operation, it is cumbersome to use for several reasons. No qualitative checking of the plan is performed, and thus plans that could be identified as logistically infeasible or inconsistent prior to a simulation, are simulated anyway. Moreover, the simulation's reports on plan execution tend to be cryptic, making it difficult to determine the source and scope of problems encountered. The combination of the lack of pre-simulation plan development aids and qualitative checking, as well as difficulty in understanding the simulation's output, necessitated many iterations of modification and simulation before a good plan was developed.

The high turnover rate of expert airlift planners also contributes to lengthening plan development time. By the time a planner becomes efficient and expert at airlift planning, it is often time to leave for another assignment. For this reason, the Air Force needed a system that would retain some planning expertise and help new planners learn their job quickly.

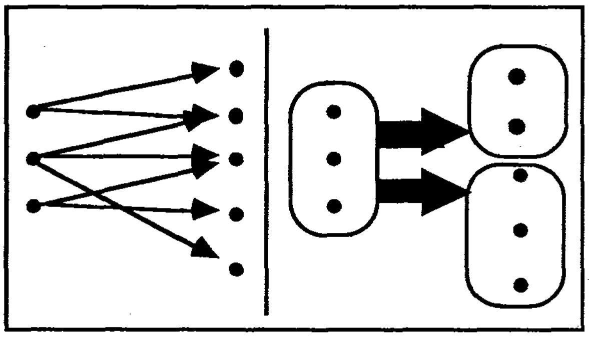
Figure 2. Routing Abstraction

The MACPLAN Concept

When the Air Force asked us to help them improve the planning process, it became evident to us that the simulation, although outdated in its technology, still performed its intended function of dynamic analysis quite well. We instead focused our attention on the plan development phase, with emphasis on preparing a logically consistent and complete plan to be ultimately checked by the simulation. We also realized that, for the purposes of deliberate planning, development of the "best plan" for a given operation was unnecessary. For this reason we designed MACPLAN to be a knowledge-based planning aid that would help produce feasible plans faster than before, and could operate as a front-end to the simulation process. MACPLAN's quality control in plan development reduces the number of lengthy simulation runs to only one or two. MACPLAN provides a suite of automated tools for quick plan development and "what-if" analysis. To further ensure that the plan is ready for detailed simulation, MACPLAN contains its own cruder internal simulation to detect dynamic difficulties in a plan. Special attention was given to the user interface so that the system could be learned quickly.

Modelling of Airlift Planning World

MACPLAN bases its model of the airlift planning domain on the Flavors[1] object-oriented extension to LISP. Flavors can be used to represent an extensive amount of declarative knowledge, and its natural hierarchical organization of classes (Winston & Horn 1989) allows many relationships to be derived by inheritance. In addition, the modular construction of Flavors gave us the critical ability to expand and modify MACPLAN incrementally. This was particularly important since MACPLAN had to be able to evolve as our understanding of the complex military airlift domain evolved. A meta-language was also built on top of Flavors and the LISP substrate to provide constraint checking capabilities (Abelson & Sussman 1987b).

Objects in MACPLAN

The objects in the MACPLAN model consist of requirements (cargo and passengers to be transported), aircraft, operators of aircraft, airfields, and routing networks. The plan element knowledge base is represented as a class structure hierarchy in which each class of objects can be treated as an object itself. All objects in the model descend from the flavor *plan element*. The plan element object infrastructure contains an identifier name and a method for enforcing slot documentation. A plan element class is created by first defining a flavor of its class name with the desired instance variables. The macro *def-plan-element* is then called to create the following class structure:

[1] Flavors is a trademark of Symbolics Corporation

```
(defstruct (plan-element (:include
                              kernel)
        (:type :named-array)
        (:predicate PLAN-ELEMENT?)
        conc-name)
name            ;from kernel
documentation   ;from kernel
database-mapper ;map for indexing
                ;instances in file and memory
instances       ;list of all existing
                ;instances of this object class
inferiors       ;all classes that inherit
                ;attributes from this class
superiors       ;all classes that this class
                ;inherits attributes from
inst-vars       ;list of all instance
                ;variables(slots) local to this
                ;class and inherited from
                ;superiors
        )
```

All plan element class objects are either *resources* or *tasks*. A *resource* is any entity, such as an airfield or aircraft, that helps to complete a task. A *task* is something that must be done, in this case something to be moved. They each contain specific key attributes that are used for their respective roles in resource allocation and task execution.

The description of each plan element class object in the knowledge base consists of three parts: a description of the class of objects to which the object belongs, the flavor instance representing the object to be used for creating instances of the class, and its attributes. Each class description defines its own attributes and inherits any that are defined by ancestors in the hierarchy.

In addition to defining class attributes, each plan element class description maintains a list of its instances. This allows the knowledge base to quickly retrieve all instances of a particular class, and to collect all instances of a class's descendents by traversing the class hierarchy.

An instance of a plan element class is instantiated or retrieved by calling the following function:

(get-object
plan-element-class-name
object's-official-name
attribute-initialization-list
documentation)

First the knowledge base looks to see if an object of this plan element class and *official-name* is already instantiated in memory. If not, the knowledge base then looks in its permanent files of information on plan elements. If an object of this class with this *official-name* is found in the permanent knowledge-base, the object is instantiated with the specific attributes recorded in its permanent description. Otherwise the object is unknown to MACPLAN, and it receives default attribute values from its class's flavor definition, or uses values for attributes as specified in the *attribute-initialization-list*. In addition, *get-object* fires any initialization methods that have been defined for this class. Initialization methods might be used to create associated display objects, or other instances or structures that this class refers to in its slots. *Get-object* returns two values, the object instance and an indicator of whether or not it was newly instantiated.

Each attribute, or instance variable, of a plan element class is defined as an object itself by the macro *def-plan-slot*. These slot objects are stored in the *inst-vars* slot of its corresponding plan element class's structure. By defining slots in this way we are able to introduce constraints on slots (Huang, Unger, & Fan 1988). Every slot has at least one constraint, a datatype constraint required by *def-plan-slot*. For instance, a slot can be constrained to contain one of several types of flavor instances, a number within a certain range, a list comprised of certain types of items, etc. Constraints are discussed in more detail in the next section.

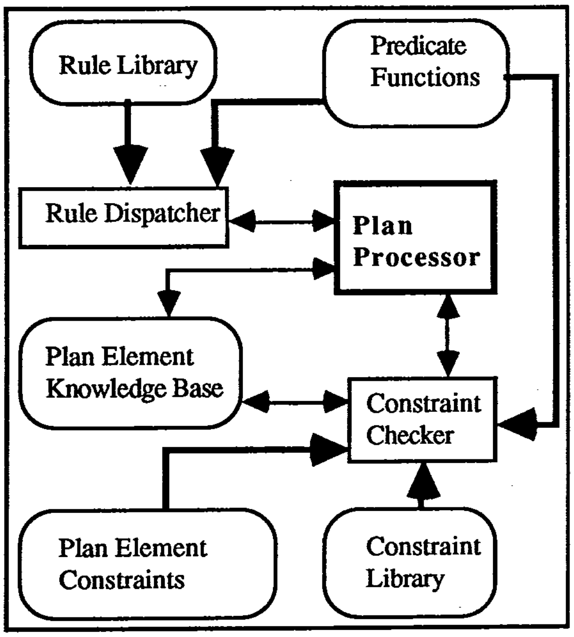

Figure 3. MACPLAN Core Architecture

How MACPLAN Reasons

Reasoning mechanisms in MACPLAN are closely coupled with MACPLAN plan elements (see figure 3). Predicates, rules, and constraints form the basic reasoning mechanisms in MACPLAN. MACPLAN predicates are hand-coded LISP functions that return two values, either *true* or *false*, and a justification for the veracity (a string value). Constraints and rules contain declarative plan element information. Predicates are called only by constraints and rules, but are themselves independent of plan element representation. Thus, predicates are not effected by changes in plan element structure.

Forward-chaining (Winston 1984) comprises one of the major mechanisms by which MACPLAN reasons. The forward chainer uses rules that infer route planning errors, inadequacies, or conflicts. Rules are a natural medium for describing the event-driven nature of the network routing problem, because collections of rules correspond to network conditions (Abelson & Sussman 1987a). The rules are derived from route planning experts (Hoffman 1987) and describe the relationships between routes and cargo movement. The domain rules are categorized by the cargo type to which they apply, and sub-categorized by degree of criticality. Rules are true statements such as:

• A critical problem exists if there are passengers to be moved along a specific route and there are no aircraft that can carry passengers.
• A critical problem exists if there are passengers to be offloaded at a specific airfield, but no operator supplying passenger carrying aircraft is allowed at that airfield.

Such rules are expressed declaratively as follows:

```
(defnetrule critical-pax
        passenger crit-pax critical
        :documentation "Can airfield
        handle at least one passenger
        aircraft and corresponding
        operator?")
```

This macro creates a rule object, places it into the appropriate category of similar rules (critical passenger), and places a pointer to the predicate function *crit-pax* in its predicate slot. This particular predicate determines whether the specified airfield can handle passenger aircraft and a legitimate operator.

Rules are applied to the routing network when the user invokes the network analysis tool. The rule dispatcher is activated when the consistency of a route is questioned. It invokes the appropriate category of rules, based on cargo type expressed in the antecedents, and returns the results of the associated predicate function that includes an explanation of any problem encountered. Rules within each category are fired by the rule dispatcher in order of decreasing criticality.

We kept MACPLAN's rule-base small and manageable by limiting rules to only represent network routing knowledge. Knowledge about plan element relationships is instead expressed in constraints. Constraint satisfaction (Charniak & McDermott 1986, Rich 1983) comprises the second major reasoning mechanism in MACPLAN. The airlift planning process can be viewed as filling slots subject to constraints (Brown 1987). A constraint says if a certain condition is true, then the contents of certain slots must satisfy certain conditions and certain other conditions cannot be true. Each slot of a plan element object is in itself an object containing slots for its value and any constraints on its value. A constraint defines the relationship which holds among a set of slots (Stefik 1986). Constraints may also be maintained between slots of different plan element classes. A plan element slot may have many constraints, which are run automatically when the slot is altered.

Constraints are built as objects, so different constraints are defined as different classes of objects. Instances of the same type of constraint can thereby be used for different classes of objects in the plan element hierarchy. The constraint object is pushed onto the constraint slot of the plan element slot(s) whose value(s) it will constrain. Constraints are declaratively expressed as follows:

```
(defconstraint min-max-launch-
                    interval
      requirements
      :min-launch-interval
      :max-launch-interval
      :shorter-eq
      "min-max")
```

The predicate *:shorter-eq* determines whether the time interval for the slot *:min-launch-interval* is longer than the time interval for its related slot *:max-launch-interval* for any object of class *requirements.*

Enforcing constraint checking at the slot level was accomplished by meta-linguistic abstraction, or the creation of a language on top of another language. We built upon LISP's innate message-sending facilities to base constraint checking upon plan element slot access. Each time a message that sets a plan element's slot is sent, the constraint checker is fired and works through the set of constraints identified for that slot. A message to a slot is built as follows:

```
(SEND plan-element-instance
      slot-message
      [new-value]
      [setting-indicator]
      [justification])
```

The *setting-indicator* determines if constraint-checking should be performed and how to apply the new value to the slot. The default value indicates that constraint checking should be performed and, provided that all constraints are satisfied, the new value should be placed into the slot. Other *setting-indicator* values indicate that constraint checking should be 1) performed but the new value should not be placed into the slot, 2) bypassed with the new value placed into the slot, 3) performed and, provided that no constraints are violated, the new-value should be deleted from the slot, 4) performed on new value as a member of a set and, provided that no constraints are violated, the new value should be inserted into the set, 5) performed on new value as a member of a set and, provided that no constraints are violated, the new value should be excluded from the set. What happens when a particular constraint is violated depends on what type of constraint it is. In some cases, a resumable error, with an explanation of the violation, is signalled. In other cases the new value is simply not applied to the slot.

MACPLAN Features

Plan Development Aids

MACPLAN provides the planner with several features that facilitate fast plan development.

• Movement requirements can be viewed graphically or textually, in the same format they were reviewed previously on paper listings, but with the added ability to select the level of viewing detail.

• MACPLAN can group airfields geo-politically to reduce the complexity of the routing network.

• The system can generate direct routes for each movement requirement to travel from its onload group to its offload group. Any legs that cannot be flown by all the aircraft in the plan are displayed in red.

• MACPLAN allows the planner to select a region on the map in which he would like to use some airfields for stop-overs. Any qualified enroute airfields in the geographic area selected are brought into the plan.

• A graphical plan routing tool, called the *Abstract,* provides an interactive electronic version of the previously described method of developing a network on paper charts.

• MACPLAN automatically adds all known information about referenced entities to the plan.

• MACPLAN tolerates incomplete information. If the system encounters an airfield that it does not know about, it accepts the airfield with default attribute values, allowing the planner to complete the information at his leisure.

- Planners can expand and modify MACPLAN's permanent knowledge base of aircraft, operators, and airfields.
- MACPLAN tracks available versus requested aircraft totals.
- MACPLAN provides the ability to save and load plans at any point in development. This was accomplished by writing a generic module that can save a list of directly or indirectly referenced objects, arrays, lists, etc, to file. The file can then be loaded so that the objects in it are re-instantiated to their previous state.
- Planners can download a plan to the external simulation by directing MACPLAN to convert its object-oriented plan format to the simulation's record-based format.

Plan Analysis Tools

For quick plan checking, a set of analysis tools was implemented.

- The airlift vs. requirements comparison determines the ability of the different types and quantities of available aircraft to move the requirements as a function of time.
- Extraneous network components can be viewed and selectively deleted from the plan.
- A network checker identifies inconsistencies and potential conflicts in the plan. Rules incorporating the planner's knowledge are used to ensure the consistency of the routing network.
- A workload estimator provides a rough estimate of the airfields' abilities to sustain the planned movement.
- An internal simulator generates approximate movement schedules, and identifies more precisely the times and locations of bottlenecks in the plan. Although some backtracking is required during the simulation, it is minimized by identifying the repeated invariant constraint checks and performing these checks once prior to scheduling.

User Interface

The user interface was implemented first so that we could effectively evaluate the functionality of the system with the users as it was developed. We made a special effort to provide a natural approach, or interface, to the problem. Thus, the planners played a key role in designing the interface.

MACPLAN utilizes a dual-headed, tiled-screen approach to maximize the viewing area and to better organize information. Menus pop-up in a predictable, context-dependent manner. Color graphics, with structured techniques in the visualization of data, portray the plan and results of analysis tools. Layered levels of abstraction and detail are used to represent entities and information in the plan, with the top level of display containing the least amount of detail. Mousing on specific icons reveals further levels of detail. Icons can be directly manipulated to alter the information they represent. Graphs and maps, with extensive use of color coding, snapshot movement tasks versus movement capability at any given time or place in the plan.

The user interface organizes all of MACPLAN's tools and displays. Natural language, implemented via a semantic grammar, enhances the intuitive nature of the interface. Plan guidelines can be easily entered using natural language, as illustrated below:

Disallow C-5s at airfield A.
Recover all military from group 1 to
group 2 using airfield C
and airfield D.

Implementation Status

MACPLAN runs as a standalone, multi-process, dual-screen system, on a Symbolics 3600 Lisp Machine. It is written in LISP using Symbolics' object-oriented extension, Flavors. Two and a half years elapsed from conception of MACPLAN to its deployment in an operational environment. MACPLAN has been operationally evaluated by planners for the past six months.

Evaluation

MACPLAN has enabled planners to develop airlift plans an order of magnitude faster than before. Planners often comment that they would have overlooked important plan details if it had not been for MACPLAN's help. New planners, even those with little or no experience using computers, come up to speed quickly, and they find MACPLAN extremely easy to use.

Conclusion and Future Work

MACPLAN employs an integration of AI, numerical formulae, and graphics technologies to support a mixed-initiative decision aid. MACPLAN has successfully enabled new planners to learn their job more quickly.

Through the success of MACPLAN, our belief in the viability of artificial intelligence has been reinforced. Interest in MACPLAN and the problem it addresses has identified new fields of research in resource allocation and scheduling at MITRE. Case-based reasoning and problem partitioning, within the discipline of linear programming, comprise two of these areas. These areas will continue where MACPLAN leaves off. Case-based reasoning will perform more of the work for the planner. While MACPLAN allows the planner to reuse a plan for different movement requirements, the planner has to do all the work in identifying an appropriate plan and modifying it to move the new set of requirements. Given a set of requirements, case-based reasoning will be used to select and adapt a suitable existing plan for the planner. This type of reasoning also provides a richly-detailed knowledge base for producing explanations about behaviors within the domain.

While case-based reasoning reduces the amount of planning work necessary by utilizing plans already made, problem partitioning can be used to optimize those plans. MACPLAN generates a plan which is feasible but not necessarily optimal. Problem partitioning will be employed to optimize the plan by decomposition.

In addition to identifying further research areas in artificial intelligence, MACPLAN has confirmed our beliefs in rapid prototyping with emphasis on developing the user interface up front. Without an interface to serve as a sounding board, the user typically loses sight of the benefits that the system could provide. In addition, the participation of the user is critical throughout the life of the prototype. Reviewing functionality periodically enables the user to see his problem more clearly and to suggest additional functionality.

Acknowledgements

We would like to acknowledge the following technical staff who developed MACPLAN: Tom Mitchell, Ruben Kleiman, Joe Kartje, Terry Gleason, Tom Doehne. We would also like to thank the following people for their various contributions: project leaders Murray Daniels and Joe Katz; our sponsor ESD at Hanscom Air Force Base in Bedford, MA; our project officer 1Lt. Joe Besselman ESD/AV-2; the Air Force planning experts, in particular: Major Brian Jones, Major Steve Gordon, and Major Greg Sensiba for their invaluable participation in the project; and General D. Cassidy, Commander in Chief, United States Transportation Command, and Commander in Chief, Military Airlift Command, for his support of the project.

References

Abelson, H., and Sussman, G.J. 1987a. Lisp: A Language for Stratified Design. AI Memo 986, MIT.

Abelson, H., and Sussman, G.J. 1987b. *Structure and Interpretation of Computer Programs.* Cambridge, MA: The MIT Press.

Brown, R.H. 1987. Allocation of Resources in the Knowledge-based Planning Architecture of CAMPS. Technical Report M87-63, The MITRE Corporation.

Charniak, E. and McDermott, D. 1986. *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley.

Hoffman, R.R 1987. The Problem of Extracting Knowledge from the Perspective of Experimental Psychology. AI Magazine 8(2): 53-67.

Huang, Y.W., Unger, E.A., and Fan, L.T. 1988. Object-Oriented Database Design Using Relational Database Methodology. In Proceedings of the Workshop on Databases in Large AI Systems, 78-91. St. Paul, MN: American Association for Artificial Intelligence.

Rich, E. 1983. *Artificial Intelligence*, New York: McGraw Hill, Inc.

Stefik, M.J. 1981. Planning with Constraints. Artificial Intelligence 16(2): 111-140.

Wilkins, D.F. 1984. Domain-independent Planning: Representation and Plan Generation. Artificial Intelligence 22: 269-301.

Winston , P.H. and Horn, B.K.P. 1989. *LISP.* Reading, MA: Addison-Wesley.

Winston, P.H. 1984. *Artificial Intelligence*, Reading, MA: Addison-Wesley.