AALPS
A Knowledge-Based System for Aircraft Loading

Debra Anderson and Charles Ortiz
SRI International

The automated air load planning system (AALPS)—a knowledge-based expert system using the expertise of highly trained loadmasters—helps users perform the complex task of loading military cargo planes. SRI developed the system through close interaction with users and domain experts; the Army now employs it daily. In this article, we (1) describe the complex set of constraints driving the AALPS planning function, (2) outline the steps in the system's development, (3) review the problem representation and reasoning methods used, and (4) discuss the system's success as shown by its widespread use.

AALPS grew out of the US Army's data distribution system testbed (ADDS), established by the Army and DARPA at Fort Bragg, North Carolina in 1979. During this period, Army users at Fort Bragg identified the need to develop an automated tool for planning and executing aircraft loads in real time. Previous approaches for developing such a tool had been unsuccessful because they relied on computationally intensive optimization techniques, and because they lacked the flexibility for modifying either the loading strategy or the many constraints restricting sets of allowable cargo arrangements. By employing AI techniques and a sufficiently flexible spatial representation, however, we were able to model the heuristic techniques commonly used by experts to load aircraft.

Automatically generating valid cargo loads (that is, ones satisfying all constraints while making good use of available space), AALPS assists transportation officers in their day-to-day planning for mission contingencies as well as for real-time load planning at the airfield. Typically, users specify a list of equipment to be loaded and the types of aircraft to be used. For each aircraft type, the system then outputs an arrangement of equipment satisfying equipment, aircraft, and mission requirements. Once at the airfield, preplanned missions can be brought up by the system, and planning parameters can be replaced with actual information obtained at the marshaling area. In this way, the system revalidates generated loads as required by runtime mission requirements.

The system also evaluates the consequences of particular loading strategies—a key factor in long-range planning and feasibility studies comparing current Army policies and equipment with proposed replacements to see how changes will affect deployment. For example, a new and slightly larger truck model might replace two of an older model—thereby affecting the number or type of aircraft needed to carry them.

Finally, AALPS verifies that user-derived loads satisfy all problem-specific constraints. AALPS’
inherent flexibility enables users to repeat load planning by changing certain parameters slightly, executing the program with new parameters, and comparing results with prior executions. For example, a typical aircraft load planning mission involving 1000 aircraft normally takes a loadmaster about a week to perform—whereas AALPS can do the same work in a matter of hours.

**The problem description**

AALPS creates cargo manifests when given a list of cargo to be transported, the aircraft types to use, and the specific type of mission. A *manifest* lists valid arrangements of cargo for each aircraft. Figure 1 shows a sample AALPS-generated manifest. Prior to the development of AALPS, expert loadmasters produced such manifests by shuffling templates of items on graph paper and manually calculating constraint values—a tedious and time-consuming process.

The Army uses three standard types of aircraft to transport equipment and personnel: the C130, the C141, and the C5 (in order of increasing capacity). We can load each aircraft with cargo including any or all of the following items:

- Wheeled vehicles,
- Palletized cargo,
- Tracked vehicles,
- Helicopters,
- Miscellaneous equipment, and
- Personnel.

Usually, we classify each mission by both aircraft type and delivery method. Missions may contain an aircraft mix, depending upon availability and need. We then consider the following loading methods:

- **Strategic air/land**: This delivery method stresses maximum utilization of aircraft space, ignoring
tactical considerations such as offloading.

- **Tactical air/land**: In this method, the ease with which items can be loaded and offloaded must be taken into consideration when preparing a load. For example, palletized cargo is loaded aft of wheeled cargo to make loading and offloading easier. Trailers are loaded hitched to trucks to speed offloading.

- **Airdrop**: In this delivery method, personnel and equipment are offloaded by parachute. Items that cannot be airdropped are loaded by one of the other methods. Items to be airdropped are mounted on special platforms. Aircraft balance throughout the airdrop process is a prime consideration.

- **The low-altitude parachute extraction system (LAPES)**: This method allows for the aerial delivery of items that cannot be airdropped. It is executed by flying the aircraft very close to the ground over a flat, cleared area. An extraction parachute is deployed from the rear of the aircraft, and the aircraft "flies away" from the item. The item is mounted on a special platform enabling it to skim to a stop.

In addition to general spatial considerations related to fitting the maximum cargo into a given space, four constraint classes restrict an item's location on an aircraft. These classes relate to

1. **Aircraft type**: A load's center of gravity must fall within a center-of-balance window for each aircraft and for all loading methods, although that window's location and extent differs from plane to plane. Aircraft floor limitations restrict the gross weight of items, the axle weight of items, the PSI values of tires, and the like; such constraints frequently involve more than one item and relationships to surrounding items. For example, a weight limitation may be imposed on a given region and, within that region, individual items may be limited to a certain fraction of the overall limit.

2. **Delivery method**: Strategic air/land missions stress maximum use of aircraft space. In a tactical air/land mission, however, speed and the ease with which cargo can be on-loaded and off-loaded takes precedence. Airdrop and LAPES missions must maintain aircraft balance while dropping cargo and must also provide space between items for parachute cables.

3. **Equipment type**: Numerous considerations are associated with each item's classification as well as with the particular item itself. Classes of items such as trucks and trailers require special consideration; these are best loaded in pairs so that the truck can unload the trailer. On the other hand, certain restricted articles (or items containing hazardous cargo) cannot be loaded with other hazardous items on the same aircraft. Finally, some items must be loaded at particular locations within the aircraft (for venting or refrigeration).

4. **Particular mission requirements**: When loading cargo for several planes, we must take into account the mission's tactical and logistical considerations. For instance, some types of cargo must be distributed
among different planes in a mission to ensure that critical assets are available in case some aircraft are lost—a tactic called “cross-loading.”

This is by no means an exhaustive list of constraints that the system must consider. Anderson et al. discuss the subject further.2

**The AALPS functional architecture**

The design of AALPS' functional architecture enables the system to interact with users when automatically considering and fulfilling the above constraints. This architecture serves three basic functions: the automatic generation of valid loads by the system, user generation of new loads, and user modification of existing loads. Figure 2 illustrates the architecture supporting these functions.

The user interface is composed of two modules—the command interpreter and the typeload editor. The command interpreter provides a friendly interface between the user and AALPS, incorporating a dynamic user model and several other congenial and adaptive interface features that vary the presentation and that support the remaining AALPS application software.3 The typeload editor maintains a graphical image of aircraft and equipment as items are loaded. It treats these items as active objects, enabling users to select them with a mouse and graphically manipulate them on the aircraft. The system can later revalidate loads altered in this way. This tool is useful

---

**Figure 2. The AALPS functional architecture.**
at deployment, when we know with certainty actual cargo characteristics (such as axle weights and center of gravity).

The equipment database contains both dimensional information and special characteristics of cargo items.

The aircraft database contains detailed aircraft characteristics including dimensions, constraint regions, and special requirements for each aircraft.

The dynamic loading requirements database contains the user's dynamically updated preferences regarding loading strategies.

The loading module contains procedural knowledge used to generate valid aircraft loads. It is a collection of representations, each delineating sequential actions associated with a particular loading strategy. These actions are represented by Prolog procedures that can specify conditional, recursive, or iterative actions associated with the loading strategies employed. AALPS uses this knowledge to determine individual aircraft load configurations as well as total mission aircraft requirements.

**System development**

To understand both problem process and content when developing AALPS, SRI worked closely with Military Airlift Command (MAC) loadmasters at Fort Bragg. These users became involved in the development process itself. Initial contact with experts and users, beginning during the last quarter of 1979, has continued throughout the system's evolution.

To sustain expert/user involvement at a level sufficient to achieve project goals, AALPS developers generated a system prototype in about two months. Experts and users tested the system, providing almost daily feedback. As a result, system developers could better understand the overall problem, and could improve and enhance existing versions. The rapid prototyping approach became a standard cycle for AALPS' development, ensuring its evolution according to user requirements.

During the development cycle, AALPS developers met with sponsors and users to provide in-process reviews approximately every four to six months. These reviews gave developers the opportunity to discuss achievements and advances made during the previous development period, and gave users the chance to express their views. Beginning in 1980 and for about two years, AALPS served as a Corps planning tool for military exercises, providing the opportunity for additional evaluation. The initial version of AALPS, written in Pascal, was executed on a DEC20.

In 1984, the system was redeveloped in Prolog because it could provide the same underlying representation for both declarative database knowledge and procedural loading knowledge. We felt that Prolog would result in a more modular system, permitting a straightforward modification of program heuristics—thereby accommodating various loading strategies for test and use. That the AALPS Prolog version meets these objectives proves the correctness of our choice.

We developed the current version of AALPS in Quintus Prolog on Sun workstations, each having approximately 2M bytes of memory and access to approximately 50M bytes of disk space from a central file server. We wrote the databases and loading module, as well as portions of the user interface, in Prolog. The typeload editor was written in C and accessed from within Prolog. The runtime environment is a compiled version for both the Sun workstation and a
VAX 11/780. Both the Sun and the VAX run 4.2 BSD Unix.

Throughout AALPS development and knowledge acquisition, the typeload editor served as a debugging and development tool—enabling developers to graphically view the consequences of adding new procedural knowledge or changing aircraft database specifics. By watching the program work as it loads, developers and expert consultants can observe loading order. Graphic aircraft database representation helps us to check for database inconsistencies and to evaluate where changes or corrections are needed.

Problem representation

The related bin-packing problem is known to be NP-hard in the strong sense, suggesting it unlikely that a polynomial solution exists. However, as has been pointed out in other NP problems—the traveling salesman problem, for example—if the requirement of finding the best solution is relaxed to finding a good solution, then heuristic techniques can be applied successfully. This motivated us to adopt a heuristic approach modeling the procedural knowledge commonly used by loadmasters.

To accomplish this, we have adopted a production system formalism consisting of a database of facts, a set of operators, and a control strategy that iteratively transforms the initial state of the cargo into a final arrangement. In this way, we can represent an action sequence as a chain of operators. We find the appropriate chain of operators by searching through a space of possible arrangements. Adopting this sort of formalism lends itself to a modular problem-solving approach. AALPS expresses a set of actions as a Prolog goal in which the antecedents of the goal statement represent the sequence of actions. Tests are associated with the application of an operator—such as two- and three-dimensional spatial checks, as well as aircraft-, equipment-, and mission-specific constraint checks. The type of operator used depends on the particular loading strategy; some of the more important operators are described below.

The make-pile operator creates piles of items—conceptual collections of items with common characteristics. For example, a pile of similar-weight items might be constructed, or a pile might specify that its items be loaded in a certain order. As described in the next section, piles are useful in the reasoning process.

Slide, swap, and reorder operators rearrange items in the aircraft. For example, the slide operator—usually invoked if a piece’s placement violates some constraint—slides items to new locations. If necessary, it can slide other pieces in the general neighborhood or even slide the entire load.

The pick-item operator chooses the most promising item for placement from the piles of available items; that is, the one most likely to satisfy constraints under consideration.

The position-item operator chooses a location on the aircraft for an item. Balancing longitudinally (from front to back) is implicitly considered when positioning items in the aircraft; one strategy frequently employed by loadmasters in the field and adopted by the system here is to load heavier items in the center of the aircraft, because the center can bear more stress, and to position lighter items toward the ends. After invoking position-item, the system conducts various tests. For example, if the plane is full, the system will proceed to balance the plane; if an item’s location violates constraints, the system will pick another item or slide the item to a different position.

The spatial representation we chose was Cartesian coordinate, rather than topological (such as quad trees). We evaluated and rejected a quad-tree approach because it (1) complicated the maintenance of variable granularities in the representation, and (2) made it difficult to represent complex topological boundaries that resulted when spacing between items was varied.

The AALPS 3D spatial representation presents all items as blocks—cubes at various levels of resolution; that is, as a collection of rectangular prisms with their edges parallel to the axis of a reference coordinate frame. For example, while it may be appropriate to consider a truck as a single block at one level, at another level of resolution it may be appropriate to consider the truck as a cube with smaller cubes underneath representing the wheels. And at an even more detailed level, it may be appropriate to consider a truck as three blocks—the hood, the driver’s cabin, and the cargo bed.

We encoded the system in horn-clause logic using the Prolog programming language. Prolog represents database facts as unit clauses. Along with a large assortment of facts regarding equipment and aircraft characteristics (such as length, width, height, and weight) is a series of facts representing constraint regions described earlier. AALPS represents these constraint regions with “checkerboards.” The system divides aircraft interiors into regions of uniform constraint values. If an item rests on more than one
weight checkerboard, for example, that item can weigh no more than allowed by the weight checkerboard with the lowest limit.

**The reasoning approach**

Loading cargo resembles the assembly of jigsaw puzzles. When loading aircraft with AALPS, however, the final arrangement can only be judged in terms of how well one has modeled a loadmaster's procedural knowledge. The system must reason about how items are loaded, and not just what the final arrangement looks like—an important consideration for tactical loading methods.

The search space is the set of all possible arrangements of items to be loaded. At any step during problem-solving, an item can be located somewhere on a plane or in a set of piles constructed according to predefined rules. Piles effectively partition a load into manageable groups of items, precluding a search through an entire list of items at each load-planning step. For example, the system can place cross-loaded items in a separate pile containing items that must be loaded on different planes. Items of similar size and weight can be kept in a pile; if a particular arrangement of items from that pile results in a good fit on one side of the plane, the same arrangement can be duplicated fairly easily on the other side. A similar strategy is often used when solving jigsaw puzzles to avoid sorting through all the pieces—a good jigsaw heuristic might involve separating blue pieces (perhaps representing the sky) from border pieces, and so on.

In general, we have adopted a hill-climbing approach with a limited number of backtracking points in which heuristics direct the search in the most profitable direction. We believe this approach approximates typical loadmaster techniques and also significantly reduces system search. In effect, this typifies expert system methodology by supplanting excessive search with more and more domain knowledge.

Domain knowledge, in this case, includes standard loading methodologies that were developed as a result of many meetings with experienced Air Force loadmasters. When differences of opinion were observed, this knowledge was coded into the dynamic loading rules database whenever possible. This loading rules database allowed the expert user to modify those rules to suit personal requirements. For example, many of the loadmasters interviewed insisted that loading oversized cargo first was the best way to structure the loading process and we encoded this knowledge in the system's control structure. In other cases, we noted that the order in which experts selected items to load on the aircraft varied with the loadmaster interviewed, and this difference of opinion resulted in an entry to the Prolog loading rules database that dictates cargo selection choices.

The majority of loading strategies share the following common order of necessary actions:

1. **Sorting the cargo:** Prior to loading, cargo is sorted into appropriate piles. Initially, users input loading parameters and list the cargo to be loaded. Using this information, the system sorts equipment into piles, enabling effective cargo selection when filling each aircraft.
2. **Filling the aircraft:** Based upon several criteria, the system selects items from the cargo list—considering cross-loading, aircraft balance, aircraft cargo area, and proper use of cargo weight capacity. The system checks for a three-dimensional fit by reasoning at the appropriate level of resolution. A single block representation of an item might suffice—or a more com-
plex one might be appropriate—depending on the shape of available space. As each item is positioned in the plane, the system also checks constraints. In air/land missions, we load from aircraft center out toward the ends. In airdrop and LAPES missions, we load from nose to tail. Our strategy in all cases seeks a good first try—one requiring the fewest alterations to achieve a valid load.

3 Aircraft balance: Once it fills the aircraft, the system checks the load’s center of balance against acceptable limits for the aircraft. If the load is out of balance, the system balances it by applying appropriate operators to the configuration. It first attempts to slide the load, then tries to rearrange items, and finally considers swapping items in the load for others on the cargo list.

4 Validating constraints: Since balancing the aircraft may entail repositioning items, the system conducts a final check of constraints. This includes checking individual items in the load as well as groups of items. Once the load has been validated, the system rechecks center of balance and (if necessary) repeats Steps 3 and 4 until all constraints are satisfied.

5 Generating typeloads: To increase efficiency, once the system generates a valid load, it replicates as many copies of that load as possible (depending on the number of items remaining to be loaded).

The system contains about 300 rules so far, and roughly 4M bytes of declarative knowledge in its database. This includes the aircraft database, the equipment database (consisting of approximately 1500 items), and miscellaneous other databases. Because of its tightly coupled development and user environments, and the capabilities it provided for aircraft load planning, AALPS demonstrated marked success with the XVIII Airborne Corps. Sound development methodology combined with the system’s successful user support caused MAC to request early initiation of tests and evaluations.

Representatives of MAC, the Air Force, and the Army viewed a live AALPS demonstration early in 1982. One result of this demonstration was an agreement that, within certain constraints, the AALPS manifest could be used for actual aircraft load plans. A statement from MAC HQ presented guidelines for using the AALPS manifest and auxiliary documentation. This event, representing a major milestone in the program’s development, was the first major hurdle in transferring AALPS technology to the Army.

Following MAC approval, the Army officially used AALPS for planning and execution (AALPS produced actual manifests), and deployed it in support of the Corps during numerous field exercises. In addition, AALPS supported the deployment of the XVIII Airborne to Granada. The success of AALPS, coupled with user enthusiasm, caused the US Army deputy chief of staff for logistics to recommend that AALPS be enhanced and adopted as the Army’s single aircraft load planning and execution system, and that it be included in the Joint Transportation Coordinator-Automated Command and Control Information Coordinator program—a recommendation endorsed by the US Army chief of staff.

AALPS represents a significant improvement over traditional, manual methods. Because of this, and because of the Army’s strong commitment, AALPS should soon become the Army’s standard air cargo planning system.
Acknowledgments

We would like to acknowledge the early work of Greg Stachnick and Richard Grise, who were responsible for initial system versions. We would also like to thank the rest of the AALPS team for their dedicated contributions: Reggie Hutcherson, Eric Negaard, Bill Park, and Julie Wong. And finally, we wish to acknowledge the useful comments that Mike Frankel, Len Karpf, Bill Baeck, and Jeff Peck provided as we prepared this article.

AALPS research was supported by the Defense Advanced Research Projects Agency and the Department of the Army under contract MDA903-84-C-0261.

References


Debra Anderson is a software engineer at SRI International. She has been the AALPS project leader since 1985, as well as a principal software developer responsible for the aircraft loading algorithms. She received her BA in computer and information science from the University of California, Santa Cruz in 1984.

Charles Ortiz is program manager of the AI applications program at SRI International's Information Sciences and Technology Center. His professional experience at SRI has been in the areas of artificial intelligence, natural language processing, cognitive science, and logic programming. Since 1984, he has been involved in developing AI tools and applications. He received his BS in physics from MIT in 1982, and his MS in computer science from Columbia University in 1984.

The authors can be reached at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.