Continuous Management of Airlift and Tanker Resources: A Constraint-Based Approach

Stephen F. Smith[1] Marcel A. Becker[2] Laurence A. Kramer[1]

[1] Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
sfs,lkramer@cs.cmu.edu
(412) 268-8811

[2] Kestrel Institute
3262 Hillview Avenue
Palo Alto CA 94304
becker@kestrel.edu
(650) 493-6871

Abstract

Efficient allocation of aircraft and aircrews to transportation missions is an important priority at the USAF Air Mobility Command (AMC), where airlift demand must increasingly be met with less capacity and at lower cost. In addition to presenting a formidable optimization problem, the AMC resource management problem is complicated by the fact that it is situated in a continuously executing environment. Mission requests are received (and must be acted upon) incrementally, and, once allocation decisions have been communicated to the executing agents, subsequent opportunities for optimizing resource usage must be balanced against the cost of solution change. In this paper, we describe the technical approach taken to this problem in the AMC Barrel Allocator, a scheduling tool developed to address this problem and provide support for day-to-day allocation and management of AMC resources. The system utilizes incremental and configurable constraint-based search procedures to provide a range of automated and semi-automated scheduling capabilities. Most basically, the system provides an efficient solution to the fleet scheduling problem. More importantly to continuous operations, it also provides techniques for selectively re-optimizing to accommodate higher priority missions while minimizing disruption to most previously scheduled missions, and for selectively "merging" previously planned missions to minimize non-productive flying time. In situations where all mission requirements cannot be met, the system can generate and compare alternative constraint relaxation options. The Barrel Allocator technology is currently transitioning into operational use within AMC's Tanker/Airlift Control Center (TACC). A version of the Barrel Allocator supporting airlift allocation was first incorporated as an experimental module of the AMC's Consolidated Air Mobility Planning System (CAMPS) in September 2000. In May 2003, a new tanker allocation module is scheduled for initial operational release to users as part of CAMPS Release 5.4.

1 Introduction

Efficient utilization of transportation resources is central to the effectiveness of operations at the USAF Air Mobility Command (AMC). In normal day-to-day operations, airlift demand must increasingly be met with less capacity and at lower cost. In crisis situations, rapid, large-scale deployment is crucial to overall military success. The allocation of aircraft and aircrews to airlift and tanker missions is a challenging problem. Several thousand missions are typically flown worldwide on a weekly basis, involving several hundreds of aircraft and comparable numbers of air crews. Individual missions impose a myriad of temporal constraints on their execution, and these constraints must be reconciled with resource availability and usage constraints (e.g., crew duty day restrictions and scheduled return dates, aircraft speed, range, and capacity) to find feasible assignments. Generally there are many more mission requests than can be accommodated by available assets in any given time frame, and hence an ability to optimize resource usage contributes directly to reduced reliance on higher-cost, commercial transportation assets.

Like many practical planning and scheduling problems, resource management at AMC is further complicated by the fact that it is situated in a continuous planning and execution environment. New mission requests of varying priority enter the system every day, and the current schedule is constantly evolving. Since taskings are incrementally communicated to the executing air wings, new requirements must be integrated into the current schedule without wholesale disruption to previous assignments. Likewise, the dynamics of execution regularly force changes to planned activities. Aircraft break down, airports become unavailable due to weather, missions become delayed due to diplomatic clearance problems, etc., and all such events can warrant reassessment of previous allocation decisions. In such execution-driven rescheduling contexts, it is similarly important to remain sensitive to solution stability concerns.

A third distinguishing aspect of the AMC allocation problem is the need for flexible accommodation of and integration with human decision-making. Given the scale and complexity of the AMC environment, there will always be user knowledge that is outside of system models and hence it must always be possible for the user to over-ride, constrain, bias, or otherwise direct system problem solving processes. While problem scale necessitates automation, effective problem solving also requires flexible user intervention.

These problem requirements are at direct odds with the design of most current transportation planning and scheduling tools. Current tools tend to be organized as black-box, batch-oriented solution generators which, when invoked, re-solve the problem from scratch. This presents three fundamental problems. First, there is no memory of solutions from one run to the next, and small changes in inputs can lead to large changes in outputs. Hence, they are difficult to control and use in circumstances where localized change is necessary or advantageous. Second, the computational cost of batch-oriented solution procedures makes it difficult for planning to keep pace with execution in dynamic (or higher tempo) circumstances, forcing the traditional disconnect between these two processes. Finally, in user-driven (and mixed-initiative) problem solving contexts, the "specify and solve" pattern of user-interaction promoted by batch-oriented solvers is inherently inefficient.

In this paper, we describe an alternative approach implemented in the AMC Allocator, a tool for day-to-day

allocation of aircraft and aircrews to airlift and tanker missions. In contrast to the above mentioned tools, the AMC Allocator is designed specifically for continuous operations, and provides a range of automated and semi-automated capabilities for allocating resources to missions against the backdrop of a pre-existing airlift schedule. Underlying the AMC Allocator is a novel constraint-based search approach to transportation scheduling. This approach is *incremental* by nature, and this characteristic is key to the Allocator's effectiveness as a continuous planning tool. The incrementality of the approach provides a direct basis for localizing and controlling solution change. It also enables real-time response to user rescheduling directives. The AMC Allocator has been positively evaluated by AMC personnel and is currently transitioning into operational use within the Tanker Airlift Control Center (TACC) at AMC. It was first embedded as an experimental component of AMC's Consolidated Air Mobility Planning System (CAMPS) in 2000, and an enhanced tanker allocation module is scheduled for operational release in May 2003.

The remainder of the paper is organized as follows. In section 2 we summarize the allocation problem faced by AMC. In section 3 we contrast identified problem requirements with traditional solution approaches, and consider the advantages of incremental, constraint-based search models as a basis for a more suitable approach. The solution framework developed within the AMC Allocator is then described in Section 4. We first review basic representational assumptions and solution generation procedures, along with the functional capabilities they give rise to. We then discuss some layered solution optimization capabilities. In Section 5, we summarize the history of the AMC Allocator Project and indicate the current status of the work. Finally, in Section 6, we indicate the current directions of our research and development efforts.

2 The AMC Barrel Master Allocation Problem

The overall mission planning, scheduling and execution process in the TACC at AMC is depicted in Figure 1. Customer requirements flow into several distinct planning offices which respond by generating *missions*. A mission typically involves the movement of cargo and/or personnel, and each planning office generates air missions of a specific type. The channel planning office, for example, is responsible for planning channel missions, which correspond closely to the types of routes flown by commercial airlines; these missions are established and revised periodically, and then flown repeatedly at regular (e.g., daily or weekly) intervals. Special Assignment Airlift Missions (SAAMs) planned by the SAAM planning office, alternatively, correspond more to chartered air flights. In this case, custom itineraries are generated to satisfy the customer's movement requirements, and the customer's requirements constitute the sole focus of the mission. The missions associated with Presidential travel are SAAM missions. Other offices are concerned with generating missions that address other types of requirements, such as contingency operations, exercises, and training. Different types of missions create different types of resource requirements, but all planned missions specify an itinerary, a priority, a particular type of aircraft (technically referred to as the Model Development Series or MDS type) to be used, a preferred USAF unit (or *wing*) assignment and a time period, represented as a pair of dates, in which the mission should be executed.

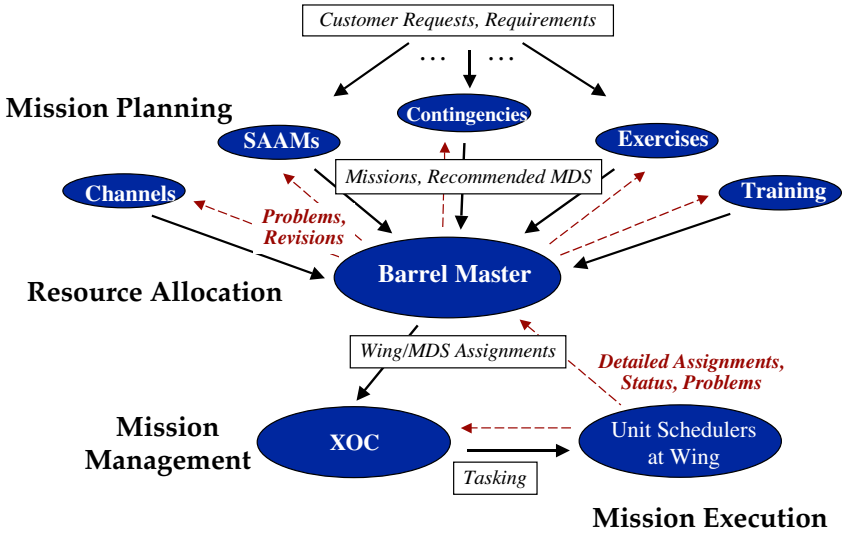

Figure 1: The AMC planning and execution environment

All generated missions flow into the "Barrel Master" office (or Barrel for short) for allocation of necessary resources. The Barrel is the office within the TACC that has total visibility of AMC assets. Based on mission specific constraints and current resource availability, the Barrel determines which missions will be supportable and for those that are, which air wing(s) will be tasked to fly them. Within the Barrel Master office, the resource allocation and management problem is distributed among multiple Barrel Masters. Each Barrel manages the aircraft and crews of a particular set of air wings, determined by aircraft type and geographic location. An air *wing* consists of a set of aircraft (and corresponding crews) of the same type stationed at a particular USAF base. For example, McGuire Air Force base has a wing of C141s and a wing of KC135s. Currently, the "West Strat" Barrel is responsible for all west coast C141, C5 and C17 wings, the "East Strat" Barrel is responsible for all east coast C141, C5 and C17 wings, the "Tanker Barrel" is responsible for all tanker wings (i.e., KC135s and KC10s) and so on.

At present, all Barrels track available aircraft and aircrew capacity at various wings, and taskings of missions to specific wings are made on this notional basis. The individual air wing has responsibility for determining the actual aircraft tail and air crew(s) that will fly a given assigned mission.[1] The resource assignment for any planned mission is managed by the Barrel until the 24 hours before execution point, at which time the mission is "pushed" to the execution office (XOC) for plan review and execution management. As problems arise during the execution of missions, this information flows back to the Barrel, and may necessitate either

[1] Plans call for this assumption to change and for the Barrel to begin to track and allocate actual tails, as AMC's "state of the world" initiative increases the visibility of execution data to planning processes.

reassignment of specific assets or revision to previously planned missions.

In allocating resources to a given mission, the Barrel must confirm that all constraints associated with its planned itinerary can be satisfied. A mission's itinerary is the sequence of stops or airports the aircraft should visit during the execution of the mission. We refer to the flight between two successive stops as a *mission leg* (or *leg* for short). Each leg has an origin airport, the Point of Embarkation (POE), and a destination airport, the Point of Debarkation (POD). Each leg is followed (or preceded) by a certain ground time. During the period the aircraft is on the ground, a number of activities, or *ground events*, can occur: for example, loading and offloading of cargo, refueling, crew rest, crew change. The time period specified in the mission request should be at least as large as the time required by the aircraft to fly between all intermediate stops in the itinerary plus the required ground time at each airport. To feasibly support the mission, a sufficient number of aircraft and aircrews of the specified MDS type must be available during this entire period. The earliest date the mission can start is called the *Available to Load Date* (ALD) and the latest date the mission should finish is called the *Latest Arrival Date* (LAD). The length of this interval should be at least as large as the total duration of the mission.

Aircraft availability and aircrew availability are defined for each wing on a daily basis. Each wing has a total number of aircraft of a particular type and a corresponding total number of aircrews. Considering that some planes are undergoing maintenance and the wing also has some need for training and local missions, the wing will make a subset of its total aircraft and crews available for tasking to AMC missions. Each day, each wing will provide a certain number of *contract* aircraft and crews that can be allocated by the Barrel. The remaining wing assets, designated as *fenced* aircraft and aircrews, are reserved for local wing use and are beyond the jurisdiction of the Barrel.

The allocation problem just outlined is similar to the classically defined *Fleet Assignment Problem:* Given a schedule of flights defining the departure and arrival times for each flight leg, the *Fleet Assignment Problem* is the problem of deciding which flight equipment, or fleet, should be assigned to each flight segment [Clarke *et al.*, 1996, Rushmeier and Knotogiorgis, 1997]. For commercial airlines, the standard objective is to maximize revenues minus operating costs. The Barrel Master, alternatively, generally tries to maximize the total sum of priorities: s/he will try to assign the maximum number of high priority missions, and will generally only consider assigning lower priority missions after all higher priority ones have been assigned.

Most traditional solutions to the fleet assignment problem assign fleets to individual flight segments. The Barrel Master, alternatively, is concerned with assigning fleets to a sequence of segments or *strings.* A *string* is a sequence of connected flight segments that begins and ends at possibly different maintenance stations [Barnhart *et al.*, 1998]. By default, an AMC mission itinerary is planned as a string that starts and ends at the same location (i.e., a round trip). Strings that start and end at the same station are usually referred as *aircraft rotations.* If possible, the Barrel would consider, and sometimes even prefer, using the same rotation for more than one mission. The difficulty in combining missions is in identifying and sorting out the opportunities for potential combinations among thousands of missions in the database.

The Barrel Master responsible for refueling assets, the Tanker Barrel, has a somewhat different allocation

problem. In this case, the starting point is an *air refueling event*, which identifies one or more receiver missions, a cumulative fuel requirement, a refueling location (or track), and a rendezvous time. Considering the availability of tanker assets at various wings within range of the target refueling constraint, together with the fuel carrying capacities and burn rates of available aircraft types, one or more tanker missions are generated, sourced to a specific wing (or wings), and linked to the originating air refueling event.

In current practice, the allocation process carried out within the Barrel Master office is a mostly manual process. In her/his daily activities, each Barrel Master utlizies an electronic *commitment matrix*, which tracks available aircraft and aircrew capacity of different wings over time and records those missions already allocated. As new missions are received from various planning offices, the Barrel consults this matrix and tries to allocate resources which satisfy mission requirements. If all requirements can be satisfied, the wing assignment is made and this commitment is communicated back to the mission planner. In those cases where there are insufficient resources available to support a particular set of missions, the Barrel will consider more disruptive allocation alternatives. For example, s/he may consider using resources already allocated to lower priority missions, s/he may consider using resources provided by a wing other than the planner's preference, s/he may consider delaying the mission, and so on. Once one or more acceptable options are found, the Barrel communicates these possibilities back to the relevant planner and a solution that would best satisfy all sides is negotiated. A mission may also be determined to be unsupportable, in which case this information is communicated back to the planner. In general, resource assignment is performed one mission at a time with little automation. Capabilities for generating more sophisticated allocation alternatives (e.g., involving mission combination) are quite limited, and generally such optimizations are not considered. In crisis situations where time pressure is greater, the lack of automated decision aids has increasing ramifications; high priority missions get accomplished, but at the expense of inordinate numbers of routine missions and at very high cost.

3 Incremental Constraint-based Search Models

The continuous, dynamic nature of the Barrel Master's allocation problem restricts the appropriateness of various automated planning and scheduling techniques. For example, classical fleet assignment solution procedures present several practical difficulties. First, they operate as black-box solution generators, and, as such, provide no ability to control solution change over time. Second, by virtue of their "re-solve from scratch" design, it can be difficult for these techniques to keep pace with execution events as tempo increases. In commercial airline settings, these shortcomings are not that important, since the set of routes to be flown tends to be rather stable (e.g., changing only month to month), and scheduling is more of a periodic activity aimed at establishing assignments for the next period. However, in the AMC environment, short notice (SAAM) missions mix with longer term (Channel) mission sets to create dynamically changing demand patterns over time. This requires more attention to advance commitment, and since new orders must be "re-cut" each time a mission is retasked there is pragmatic utility to minimizing disruption to scheduled missions when responding to new mission requests. Moreover, the higher time pressure associated with crisis situations puts increased emphasis

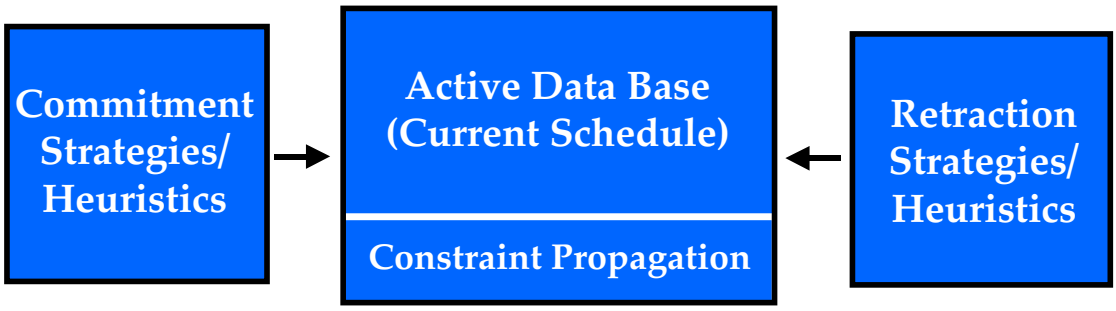

Figure 2: Constraint-Based Scheduling Models

on computationally efficient scheduling and allocation processes. In these contexts, automated techniques that integrate flexibly and gracefully with user decision processes is an important additional necessity.

Constraint-based search models provide a general approach to scheduling that is well matched to the requirements of the Barrel Allocator's allocation problem. Constraint-based scheduling models combine principles of constraint programming and heuristic search in the formulation and solution of scheduling problems (e.g.,[Smith *et al.*, 1996, Cheng and Smith, 1997, Baptiste *et al.*, 2001]). Figure 2 shows the basic framework, which consists of three main components. In the center is an *active data base*, which contains a representation of the current solution. As scheduling decisions are posted to or retracted from this data base, a set of *constraint propagation* routines are triggered. Propagation computes the consequences of any change for related scheduling decisions, possibly winnowing (or enlarging) the set of feasible values for other decision variables or detecting a constraint conflict (which signifies an infeasible state). The other two principal components of a constraint-based scheduling model are commitment and retraction strategies/heuristics, which respectively guide the search process in moving forward (e.g., allocating resources or assigning start and end times to an as yet unassigned mission) or moving backward (e.g., unassigning a previously assigned mission) in the underlying search space. By configuring commitment and retraction heuristics with different search control mechanisms and search space assumptions, it is possible to define a range of both constructive (i.e., global search) and repair-based (i.e., local search) scheduling methods.

Constraint-based scheduling models are attractive in the current context for several basic reasons:

- *Incrementality* - By definition, constraint-based scheduling models are incremental decision-making procedures. As such, they provide a direct basis for managing solution change over time, and for minimizing undue disruption to the current schedule as new demands (missions) are accommodated.
- *Generality* - Constraint-based models have been shown to operate efficiently under quite general representational assumptions. Simple Temporal Problem (STP) constraint networks [Dechter *et al.*, 1991], for example, allow expression of a diverse range of relative and metric temporal constraints, while providing

efficient incremental propagation algorithms. Similarly rich representations of resource constraints are also possible.

- *Configurability* - Constraint-based search procedures are also quite compositional by nature and these models tend to be easily reconfigured to accommodate new constraints and objectives [Smith *et al.*, 1996]. In the case of the AMC Allocator described below, this property is used to provide selective constraint relaxation capabilities in situations where all constraints cannot be simultaneously satisfied.

4 The AMC Allocator

As just suggested, an incremental, constraint-based scheduling model provides the core basis for the functionality provided by the AMC Allocator, a software tool designed to address the AMC Barrel Master allocation problem. Most basically, this model is used to define an efficient solution to a dynamic, string-based variant of the fleet assignment problem. More importantly from the standpoint of continuous operations, it is also used to define techniques for selectively re-optimizing to accommodate higher priority missions while minimizing disruption to most previously scheduled missions, and for selectively "merging" previously planned missions to minimize non-productive flying time. Finally, it is used to generate and compare alternative constraint relaxation options.

The AMC Allocator has been developed using the Ozone scheduling framework [Smith *et al.*, 1996, Becker, 1998], a shell for constructing constraint-based scheduling applications. Ozone consists of three major components: (1) a library of modeling primitives for constructing scheduling domain models, (2) an underlying constraint-based search architecture (an elaborated version of the basic model in Figure 2), and (3) a library of scheduling and rescheduling methods (encoding various commitment and retraction strategies/heuristics). Through instantiation of an *abstract domain model* with modeling primitives that match the characteristics of a given application domain, and selection of one or more basic scheduling methods, it is possible to quickly obtain an executable model, and allow development effort to quickly focus on the customizations of the model and the scheduling methods necessary to obtain high performance. The Ozone application framework consolidates the results of application building experiences in a wide range of complex scheduling domains.

The AMC Allocator itself currently provides two core sets of functionality to the AMC Barrel Master: *resource allocation* and *mission combination*. Separate modules are provided for airlift allocation and tanker allocation respectively, owing to differences in the nature of these two allocation problems as currently structured within the Barrel Master office. In all cases, functionality may be utilized in a more or less automated fashion, ranging from a fully manual mode where the system does little more than decision bookkeeping, to a semi-automatic mode, where the system generates alternative options and previews their impact, to a completely automatic mode, where the system determines the best decisions based on user-specified preferences. The system may also be invoked in an extended optimize mode, in which case the underlying incremental search process that is performed is broadened. All of these capabilities are provided within a graphical, spreadsheet-like

user interface, designed to emphasize the real-time responsiveness of the system's scheduling and allocation procedures.

In the following subsections, we describe the representations and search procedures underlying the AMC Allocator in more detail. The configurability of these search procedures is considered in further depth in [Becker and Smith, 2000]. Other mixed-initiative aspects of the approach are emphasized in [Kramer and Smith, 2002].

4.1 Representational Assumptions

Within the AMC Allocator, missions are represented as hierarchical task (or activity) networks. A mission expands into a network of more detailed flight segments and ground activities. Flight segments correspond to individual legs of the mission's itinerary; ground activities model such activities as crew rest periods and aircraft preparation tasks. Each constituent activity has a *start time* and an *end time*, and is linked via precedence constraints to other mission activities. The duration of an activity is a function of its type. For example, the duration of a flight activity depends on resource speed and the distance traveled from its *origin* to its *destination*.[2] For many ground events, durations are specified as fixed constants. The duration of an aggregate activity (e.g., a mission) is the duration of its sub-network.

A mission specifies requirements for resource capacity for its entire duration. In the case of airlift missions, some number of aircraft and aircrews of a particular type are specified. In the case of air refueling (or tanker) missions, there is often flexibility with respect to type of refueling aircraft, and the required numbers of aircraft and aircrews are derived from fuel offload requirements and fuel carrying capacity limits. A mission also has associated temporal restrictions on when it can execute. In the case of an airlift mission, execution is constrained by its "Available to Load date" (ALD) and its "Latest Arrival Date" (LAD). Tanker missions, alternatively, must synchronize to arrive at the refueling point at the designated "Air Refueling Control Time" (ARCT).

Aircraft and aircrews are modeled at an aggregate wing level. A wing designates a co-located set of aircraft and aircrews of a particular type. It specifies a particular geographic *location*, a pool (or fleet) of aircraft and a pool of aircrews. A given pool of aircraft or aircrews has a *total capacity*, with each unit of capacity representing one aircraft or aircrew. The total capacity of a resource pool is partitioned into two subsets: *contract* capacity, representing the capacity that has been budgeted to AMC and is available for allocation, and *fenced* capacity, representing the aircraft that have been held back for the wing's local use. The contract and fenced capacity of a wing can vary over time, and resource availability will become further restricted as capacity is assigned to specific missions over time. The available capacity of a resource pool is thus represented as a sequence of *capacity intervals*, each recording the total, contract, fenced and available capacity over the interval spanned by its start and end time. As missions are assigned to wings (or de-assigned) during the allocation process, the affected portion (along the time line) of their available capacity representations is updated to reflect the

[2] Currently, distances are computed based on great circle route.

reserving (or freeing up) of required aircraft and aircrew capacity. Thus resource capacity is represented and allocated over continuous time, at some level of temporal granularity.

The output of the AMC Allocator (i.e., the current schedule) is a set of assignments of the form *(Mission, Wing, start-time, end-time)*, indicating that *Wing* will perform *Mission* over the interval from *start-time* to *end-time*. To be feasible, such an assignment must satisfy the following constraints:

- **Wing capacity constraints** - The cumulative resource requirements of all missions assigned to *Wing* over the interval from *start-time* to *end-time* must not exceed the number of contract aircraft or aircrews available at *Wing* for any t, *start-time* $\leq t \leq$ *end-time*.
- **Resource ground time constraints** - The assigned interval from *start-time* to *end-time* must account for aircraft onload, offload and minimum time-on-ground requirements. Each of these constraints is specified as a function of aircraft type.
- **Flight duration constraints** - The assigned interval from *start-time* to *end-time* must account for cumulative flight time requirements, including necessary positioning and depositioning flight segments.
- **Aircraft range constraints** - Any given flight segment inserted into *Mission's* itinerary as a consequence of assigning *Wing* must cover a distance less than the flying range of the assigned aircraft type. In general, the creation of positioning flights, depositioning flights and/or bridging legs (from offload of one mission to onload of the next) may be implied by a given wing assignment.
- **Crew duty day constraints** - Depending on positioning/depositioning considerations and the crew type required by *Mission* - basic or augmented - it may be necessary to insert crew rest periods at appropriate intermediate points of the mission to enforce crew duty day restrictions. The assigned interval from *start-time* to *end-time* must account for this additional constraint as well.
- **Mission timing constraints** - Finally, *start-time* and *end-time* must be consistent with all timing constraints on the execution of *Mission*. In the case of an airlift mission, the start of the first cargo carrying flight segment must be $\geq$ *Mission's* ALD, and the end of the last cargo carrying leg must be $\leq$ *Mission's* LAD. For tanker missions, the end time of the positioning flight must $=$ *Mission's* ARCT.

Given the oversubscribed nature of the Barrel Master allocation problem, it is not always possible to satisfy mission time constraints with existing available lift capacity, and those missions that cannot be feasibly scheduled at any point are designated as unassignable. In such situations, the user may choose to analyze and compare options that involve relaxation of various constraints. Currently, the following relaxed formulations may be solved to generate such options:

- **Priority-based pre-emption** - Every mission has a pre-defined priority that establishes its intrinsic importance. Under a priority-based pre-emption formulation, an unassignable mission may obtain its required resource capacity by pre-empting (or bumping) one or more previously assigned, lower-priority

missions. If this occurs, the set of pre-empted missions is then rescheduled in succession and may, in turn, find alternative resource assignments at the expense of still lower priority missions. At quiescence, any remaining unassigned missions become unsupportable and are added to the current set of unassignable missions. When operating under this formulation, additional constraints can be added by the user to exert greater control of the extent of solution change. A *mission locking* mechanism allows any specific mission assignment to be designated as unalterable. A *freeze interval*, specifying a period of time in advance of execution within which pre-emption is not allowed, can also be imposed.

- **Resource Over-allocation** - Under this formulation, the aircraft and aircrew contract capacity constraints are considered relaxable. Since the numbers of contract aircraft and aircrews are typically smaller than the total number of assets possessed by the wing, the user may choose to go over the published contract levels of a given wing. This happens with a fair amount of frequency. It generally reflects private knowledge that the Barrel Master may have about wing assets or agreement on the part of the wing to use fenced aircraft.

- **Mission Delay** - Under this formulation, the mission's LAD constraint (in the case of airlift) or ARCT constraint (in the case of refueling) is considered relaxable. The user may consider the option of delaying the current mission until necessary resources are available. If delay seems like a potentially viable alternative then this information can be suggested to the mission planner.

- **Alternative MDS** - Under this formulation, the requirement that the mission use the airframe (or MDS) type designed by the mission planner is relaxed, and alternative aircraft types can be considered as a means of accommodating the mission. Due to differences in carrying capacities of different aircraft, the numbers of aircraft and crews required to support a mission may vary across aircraft type as well. As with mission delay, potentially viable options can be communicated back to the mission planner as suggestions.

- **Composite mission constraint relaxations** - Formulations that permit simultaneous relaxation along multiple dimensions (e.g., Mission Delay and Priority-based Pre-Emption, Mission Delay and Over-Allocation) are also possible.

- **Mission Combination** - By default, missions are flown as round trips from a particular home base. If the origin and/or destination do not coincide with this base, then positioning and/or de-positioning flight segments are added into the mission itinerary. In mission combination mode, the requirement that each mission be flown as a round trip is relaxed, and opportunities to exploit non-productive flying time by "recycling" an aircraft directly from the destination (offload) of one mission to the origin (onload) of the next are sought. Mission combination can provide a direct option for supporting an otherwise unassignable mission. It can also provide a basis for compressing the resource requirements of a set of previously assigned missions and reclaiming resource capacity for other use.

Below, we outline solution procedures for these various problem formulations.

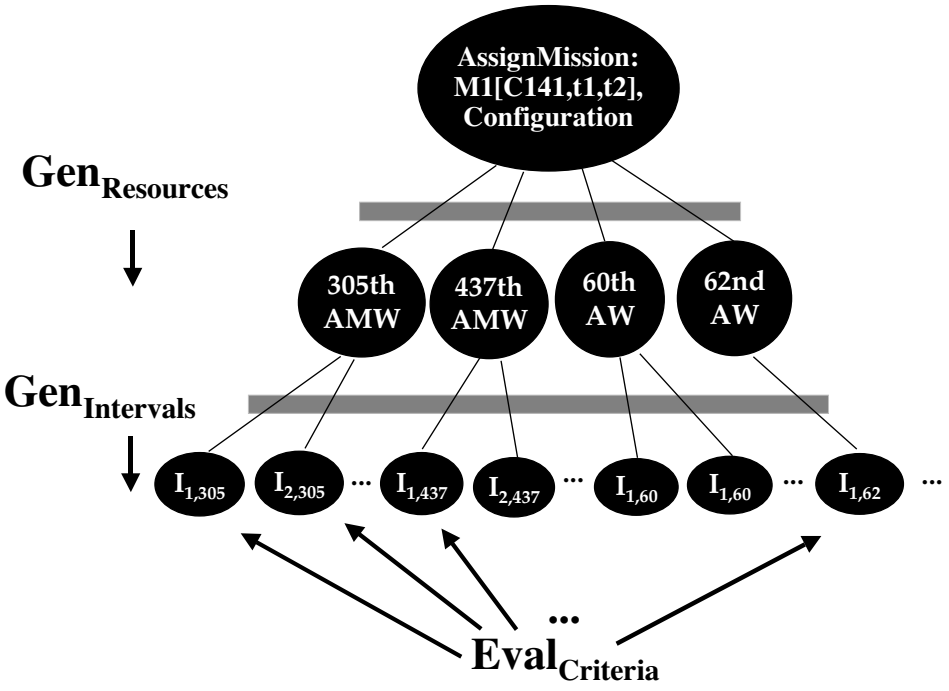

Figure 3: Basic search procedure for resource allocation. This example assumes that mission $M1$ requires $C141$ aircraft with earliest pickup at $t1$ and latest delivery by $t2$), and that $C141$s can be provided by either the 305th Air Mobility Wing (AMW), the 437th AMW, the 60th Air Wing (AW) or the 62nd AW.

4.2 Allocation of Airlift Assets

The central function supported by the Allocator is that of assigning planned missions to wings over time. In typical mode of operation, there is an existing set of assignments (i.e., the current schedule) in place, and the problem is one of determining assignments for sets of newly planned missions in the presence of this backdrop. Figure 3 graphically depicts the basic search procedure used to make the wing/time assignment for a single mission.

The search procedure proceeds in 3 basic steps:

1. a set of candidate resources (wings) is generated,
2. for each candidate wing, a set of possible allocation intervals is generated, and
3. each <wing, allocation interval> pair is evaluated and the highest ranked candidate is selected.

As implied by Figure 3, a full instantiation of the `AssignMission` search procedure is obtained by specifying three components: a search operator for generating candidate resources (referred to generically as

$Gen_{Resources}$), a search operator for generating candidate allocation intervals (generically called $Gen_{Intervals}$), and an evaluation metric ($Eval_{Criterion}$) for ranking alternatives. By parameterizing the procedure to operate with different sets of operators and evaluation criteria, resource assignments can be generated and evaluated under a range of different constraint relaxation assumptions.

In the most basic case, `AssignMission` is configured to search only for feasible assignments, i.e., <`wing, allocation interval`> pairs that are consistent with the time and resource requirements specified by the mission and are also compatible with the assignments of previously scheduled missions. This *feasible* configuration of `AssignMission` is obtained by incorporation of the triple $< Gen_{RequestedRes}, Gen_{FeasibleInts}, Eval_{MinFlyingTime} >$. Here, $Gen_{RequestedRes}$ generates candidate wings consistent with aircraft type requested by the mission. Likewise, $Gen_{FeasibleInts}$ scans a candidate wing's aircraft and aircrew capacity profiles for allocation intervals (1) with sufficient amounts of available capacity to support mission requirements, (2) with a duration greater than or equal to the time required to accomplish the mission (a function of mission itinerary, aircraft speed, wing's home base location, crew rest requirements, and other constraints on duration identified in Section 4.1), and (3) with start and end times that satisfies the mission's earliest on-load time and latest off-load time constraints. Candidates are differentiated on the basis of total flying time and the candidate assignment that minimizes this metric is selected.

By selectively substituting different search operators and/or evaluation criteria, `AssignMission` can alternatively be used to find assignments under various relaxed problem assumptions identified in Section 4.1. For some types of constraints, relaxation simply implies the consideration of a different discrete set of options. For example, substitution of $Gen_{AlternativeRes}$ for $Gen_{RequestedRes}$ results in generation of assignments that consider types of aircraft other than the type requested by the mission planner. For other classes of constraints, however, relaxation is more continuous in nature, and in substituting a search operator that assumes constraints can be relaxed, the search must also be biased to promote their satisfaction to the extent possible.

By varying the operator used to generate allocation intervals and the evaluation metric used to prioritize candidate solutions, a number of useful `AssignMission` configurations are defined:

- **Delay** - Incorporation of the triple $<Gen_{RequestedRes}, Gen_{DelayInts}, Eval_{MinTardiness}>$ yields an assignment procedure which assumes that mission deadlines can be relaxed if necessary. $Gen_{DelayInts}$ uses the same mechanism used by $Gen_{FeasibleInts}$ but considers a larger portion of the candidate wing's aircraft and crew capacity profiles, and $Eval_{MinTardiness}$ ensures that the mission deadline will be relaxed to the minimum extent possible

  In this configuration and others, advantage can be taken of the relationship between search operator and evaluation criterion to effectively constrain the number of candidate solutions generated. For example, by scanning forward in time through the capacity profiles of required resources, the first interval with available capacity found will be the one that minimizes delay for that resource. If this approach is taken only one interval need be generated for each candidate resource. In other configurations (e.g., the *preemption* case below), where there is no such dominance condition for constrained solution generation,

more ad-hoc heuristic cutoffs can be used.

- **Over-allocation** - The triple $<Gen_{RequestedRes}, Gen_{OverInts}, Eval_{MinOverUsage}>$ defines an assignment procedure where capacity constraints are relaxable. $Gen_{OverInts}$ scans the capacity profile of a candidate wing, but generates allocation intervals that extend above the wing's "contracted" level (i.e., dipping into its locally reserved or "fenced" pool of aircraft capacity). $Eval_{MinOverUsage}$ promotes selection of the generated allocation interval that minimizes the level of over-allocation.

  In this case, maximal intervals at different levels of over-allocation can be efficiently generated via linear scans of the capacity profiles of the two required resources, and subsequently pruned to minimize the temporal extent of over-allocation.

- **Priority-based Pre-emption** - A configuration which assumes that some number of previously made assignments can be relaxed (or disrupted) is defined by the triple $<Gen_{RequestedRes}, Gen_{BumpInts}, Eval_{MinDisruption}>$. This configuration implements a form of pre-emption, based on mission priority. In scanning a candidate wing's capacity profile, $Gen_{BumpInts}$ considers capacity currently allocated to lower priority missions as available for assignment, and generates allocation intervals based on this assumption. $Eval_{MinDisruption}$ promotes allocation intervals that disrupt the fewest missions and those with the lowest priority. This minimizes the cascading effect (since any mission that is pre-empted by a higher priority mission is recursively re-scheduled using the same procedure).

  Given the combinatorial number of allocation intervals that can be generated via a complete capacity profile scanning procedure ($O(c^f)$ where c is the capacity of the resource and f is the duration of capacity profile fragments), our current implementation utilizes a linear, heuristic sampling strategy compatible with $Eval_{MinDisruption}$. Briefly, allocation intervals are generated by single forward scan through the required resources' capacity profiles over the time interval where capacity is required. At each time point encountered during the scan, the set of pre-emptable missions is computed. If non-empty, the current allocation interval is extended by (1) computing the subset of pre-emptable missions of lowest priority and (2) selecting the mission in this subset that maximally extends the current interval. If there are no pre-emptable missions, the current allocation interval is ended (and retained if of sufficient duration), and the scan continues at the start point of the next pre-emptable mission.[3]

- **Composite Relaxations** - Components of the above base configurations can also be composed to define configurations of `AssignMission` where multiple constraints are simultaneously relaxed.

`AssignMission` (in any of the configurations described above) can be applied to any selected set of missions via the `AssignMissions` procedure given in Figure 4. Within this Constraint Satisfaction Problem Solving (CSP) style search procedure, the *sort-order* parameter determines the overall variable-ordering

[3] An alternative heuristic strategy, which attempts to find pre-emptable missions that best match target "resource area" requirements, is described in [Zhou and Smith, 2002]. Though somewhat more costly computationally, this approach has been found to be considerably less disruptive than the above strategy, and we plan to incorporate it as an option in a future Allocator release.

AssignMissions($UnassignedMissions, configuration$, *sort-order*)
  While $UnassignedMissions \neq \emptyset$ **Do**
    $M \leftarrow$ SelectMission($UnassignedMissions$, *sort-order*)
    $UnassignedMissions \leftarrow UnassignedMissions - \{M\}$
    If `AssignMission(`$M, configuration$`)`
      Then $Status_M \leftarrow$ `assigned`
      Else $Status_M \leftarrow$ `unassignable`
  EndWhile
End

Figure 4: Overall Mission Scheduling Procedure

strategy (i.e., the order in which missions are selected for assignment). In the current implementation, mission priority (first) and latest delivery date (second) are used as a heuristic basis for prioirtizing unassigned missions. Input missions are sorted on this basis (within `SelectMission`) and `AssignMission` is then sequentially applied to each to allocate required resources (using whichever of the above configurations has been designated).

In cases where a given mission has no feasible assignments (i.e., `AssignMission(M, feasible)` is applied and returns no solution), an exploration of possible relaxation options can be conducted through repeated application of `AssignMission` in different configurations. In the current implementation, this procedure is used in what-if mode to generate alternative options, and the user performs the evaluation and selection. Figure 5 shows a sample output as displayed within the Allocator's "Compare Options" interface.

4.3 Allocation of Air Refeuling Missions

The process of allocating wings to refueling or "tanker" missions is similar to that used for assignment of airlift missions, but requires an important extension. Tanker missions are generated and sourced to satisfy an *air refueling (AR) request*, which minimally calls for a specified amount of fuel to be delivered to a particular rendezvous point at a particular time. The number of tanker missions required to satisfy a given AR request will be a function of the fuel required and the fuel capacity of aircraft allocated, and hence may not be specified in advance. The AR request may also specify a particular aircraft type and possibly a preferred wing assignment, which is referred to as a specific tanker request.

Given these distinctions, the tanker allocation process must incorporate an additional mission planning function. The extended `AssignRequest` procedure is given in Figure 6, In this procedure a mission generation and assignment loop is iterated until either the fuel requirement has been satisfied or it is determined that the

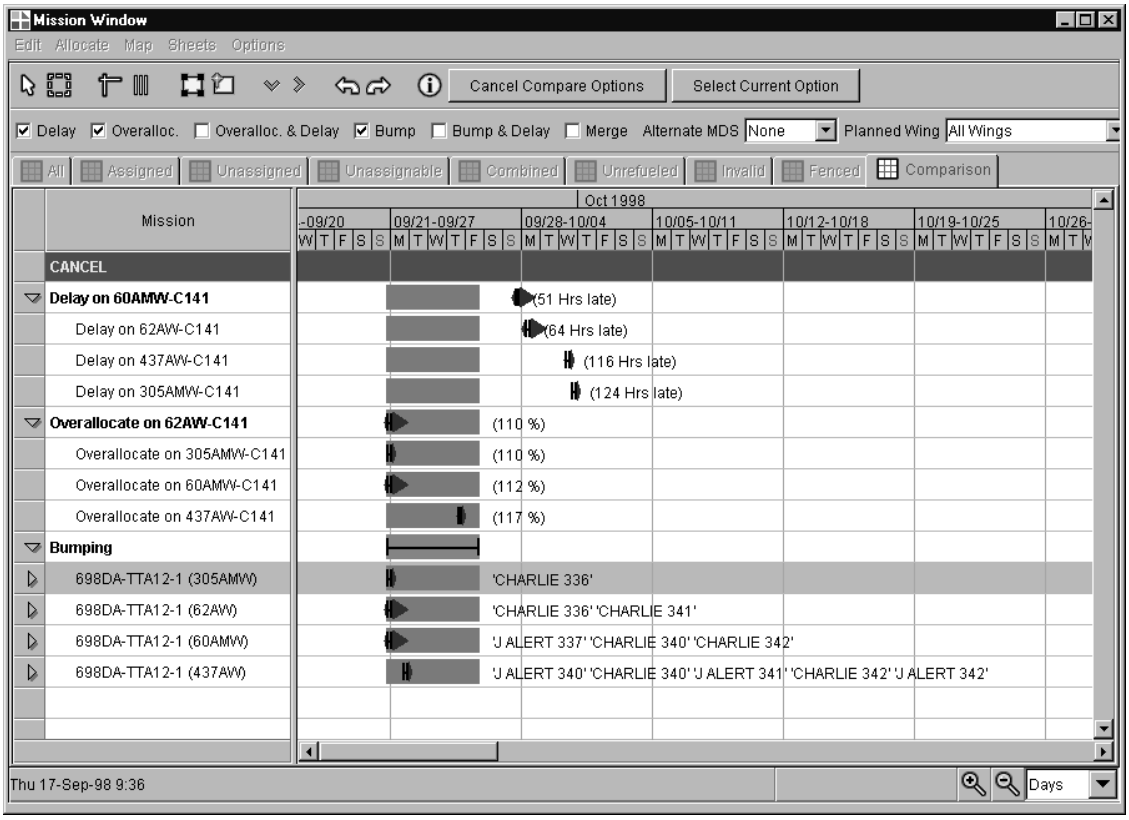

Figure 5: Generating Options for an Unassignable Mission

AssignRequest($AR, configuration$)
  While $RequiredFuel_{AR} > 0$ **Do**
    $M \leftarrow$ GenerateNewTankerMission(AR)
    If `AssignMission(`$M, configuration$`)`
      Then $RequiredFuel_{AR} \leftarrow RequiredFuel_{AR} - FuelCapacity_M$
      Else $Status_{AR} \leftarrow$ `unassignable;` **Return**
  EndWhile
  $Status_{AR} \leftarrow$ `assigned`
End

Figure 6: Air Refueling AssignRequest Procedure

AR request is unsupportable. As indicated, the full range of AssignMission configurations available for airlift allocation are equally applicable in the air refueling allocation context.

4.4 Mission Combination

A second core function provided by the Barrel Allocator is mission combination. This capability is defined by coupling the basic `AssignMission` procedure with a procedure for generating possible mission pairings.

More specifically, the process of merging a given mission M with another existing mission proceeds in three basic steps:

1. Generate a set of possible composite missions - In this step, all possible composite missions involving M are generated. For any pair of missions (M_1,M_2), there are two potential pairings: one in which mission M_2 flies after the end of mission M_1's last leg; and one in which mission M_1 flies after mission M_2. A possible combination is one which (1) satisfies the conjunction of constraints established by spatial and temporal restrictions, and (2) establishes user-imposed limits on acceptable candidates. The user may specify three additional constraints on acceptance:

   - a minimal percentage reduction in total duration of the combined mission,
   - a maximum layover time between missions, and
   - a maximum distance between the destination of the first mission and the origin of the second.

   The output of this step can be seen as the set of all notional composite missions $\overline{M}$ resulting from concatenation of possible combination pairs (M_1,M_2).

CombineMission($M, MaxLayover, MaxDistance, ReqReduction$)

$PossibleCombinations \leftarrow \emptyset$

For each mission pair $(M_1, M_2) : (M_1 \vee M_2 = M) \wedge (M_1 \neq M_2)$ **Do**

$\overline{M} \leftarrow$ GenerateCompositeMission(M_1, M_2) ; $\overline{M}$ = M_1+$M_{1,2}$+M_2

If ConstraintsSatisfied($\overline{M}, MaxLayover, MaxDistance, ReqReduction$)

Then $PossibleCombinations \leftarrow PossibleCombinations \cup \{\overline{M}\}$

EndFor

$Candidates \leftarrow \emptyset$

For each $\overline{M} \in PossibleCombinations$ **Do**

If `AssignMission(`$\overline{M}$`,feasible)` **T**hen $Candidates \leftarrow Candidates \cup \{\overline{M}\}$

EndFor

SelectCandidate($Candidates, Eval_{MaxTripRed}$)

End

Figure 7: Mission Combination Procedure

2. Generate set of feasible composite missions - In this step `AssignMission` is applied to each candidate $\overline{M}$ identified in step 1 to determine the set of composite missions that are resource feasible (i.e., there is available resource capacity to support the composite mission over its entire duration). Although currently only *feasible* allocations are considered, any of the relaxed `AssignMission` configurations could also be used.

3. Selection of best candidate - Once all feasible combinations have been determined, the candidate providing the largest overall reduction in airplane flying time is selected.

Figure 7 summarizes the mission combination procedure. Note that although `CombineMission` performs only pairwise merging of missions, the combination of larger numbers of missions can be accomplished by recursively applying the algorithm to composite missions.

4.5 Incremental Optimization

The above described procedures are designed to flexibly support a continuous, interactive resource management process. New missions are integrated incrementally as problem constraints permit, and in circumstances where missions cannot be feasibly accommodated, a range of constraint relaxation and solution change options can be explored to find acceptable compromises and increase the number of supportable missions. In operation, the system provides an efficient, mixed-initiative decision-making environment – a 2 week interval of missions

extracted from the current AMC data base (approximately 1000 missions, 5000 flights) is scheduled from scratch in less than 5 seconds on a Pentium 4 1GHz machine. More typical incremental planning and scheduling actions are executed in real-time.

This efficiency is gained principally through reliance on search heuristics, and within the core Allocator procedures, automated solution generation is strongly biased by mission priority. This coincides well with AMC business practice, but can also sometimes result in missed opportunities to support additional lower priority missions.

To compensate for such circumstances, we have been exploring the development and use of incremental optimization procedures, which can be applied at extended computational cost when decision time constraints permit. One simple example provided within the current Allocator release is a "resource usage compression" capability, which applies the basic mission combination procedure discussed above to the entire set of missions allocated to a wing (or set of wings) over some interval, to identify opportunities for more efficient recycling of assets and to reclaim resource capacity for additional tasking.

Another incremental optimization procedure we have recently developed attempts to incorporate additional missions by temporarily de-emphasizing mission priority as a pre-emption criterion and exploring other feasible <resource, allocation interval> assignments for selected higher priority missions [Kramer and Smith, 2003]. One consequence of the priority-based allocation bias of the core `AssignMissions` procedure is that some (higher priority) missions with greater scheduling flexibility may be assigned before other (lower priority) missions with less scheduling flexibility; and this possibility can lead to sub-optimal greedy commitments that leave lower priority missions unassignable. The `MissionSwap` procedure (see Figure 8) is designed to provide a framework for locally rearranging assignments to take better advantage of the collective scheduling flexibility associated with a given set of competing missions. It proceeds by retracting some number of currently assigned missions, which frees up capacity sufficient to enable insertion of the (previously unassignable) input mission. `MissionSwap` is then recursively called for each retracted missions that cannot be immediately assigned elsewhere (by a subsequent call to `AssignMission`), with the additional constraint that once a mission has been reassigned via a swap move, it is "protected" from further retraction.[4] In our current implementation, the results of MissionSwap are incorporated conditionally; If the procedure bottoms out with feasible reassignments for all retracted missions, then the results are accepted as a new improved solution. Alternatively, if all retracted missions cannot be feasibly reassigned, then the solution is restored to its prior state and the original input mission is returned to "unassignable" status".

Central to the effectiveness of this swapping procedure is the heuristic employed to select which mission or missions to retract in a given conflict situation. Conceptually, one would like to retact the mission (or missions) that possess the greatest potential for reassignment. One simple estimate of this potential is the scheduling flexibility provided by a mission's feasible execution window. More precisely, we define the flexibility of a

[4] It should also be noted that since this and other incremental optimization procedures utilize the same underlying search infra-structure, additional constraints can always be introduced to prohibit the system from changing specific allocation decisions if greater control over solution change is required.

MissionSwap($M, ProtectedMissions$)

$ConflictSet \leftarrow$ ComputeCapacityConflicts (M)

$Retracted \leftarrow$ RetractMissions ($ConflictSet, ProtectedMissions$)

if $Retracted = \emptyset$ **then** Return($\emptyset$) ; no possibility to assign M

$ProtectedMissions \leftarrow ProtectedMissions \cup \{M\}$

`AssignMission(`M`, feasible)`

`AssignMissions(`$Retracted$`, feasible, least-flexible-first)`

For each ($i \in Retracted \wedge status_i =$ `unassignable`) **do**

$ProtectedMissions \leftarrow$ MissionSwap($i, Protected$)

if $ProtectedMissions = \emptyset$ **then** Return($\emptyset$) ; unable to reassign i

EndFor

Return($ProtectedMissions$) ; success

End

Figure 8: Mission Swap Procedure

given mission m as

$$Flex_m = \frac{\sum_{r \in R_m} \textit{alloc-dur}_{r,m}}{(lft_m - est_m) \times |R_m|}$$

where R_m is the set of alternative resources (i.e., wings) that could support m, *alloc-dur*$_{r,m}$ is the duration that resource r would require to perform m, est_m is the earliest start time of m, and lft_m is the latest finish time of m. This measure of scheduling flexibility gives rise to the following retraction heuristic:

$$MaxFlex = i \in C : Flex_i \leq Flex_j \forall j \neq i$$

where C is a set of conflicting missions in the $ConflictSet_u$ for some unassignable task u. This $MaxFlex$ heuristic drives the `RetractMissions` step of `MissionSwap`.

Preliminary experimental analysis of the performance of the `MissionSwap` procedure on representative AMC mission data at different levels of resource contention has indicated an ability to substantially improve the airlift schedules initially produced by `AssignMissions` in certain circumstances. As well, the $MaxFlex$ heuristic has been shown to consistently outperform several, more complex "contention-based" retraction heuristics at a fraction of the computational cost. The reader is referred to [Kramer and Smith, 2003] for a complete description of this incremental optimization procedure and for full details of these initial performance results. Our current work is investigating the performance/cost tradeoffs associated with variants of `MissionSwap` that carry out additional search when initial retraction choices don't pan out. We anticipate full incorporation of this capability in a future Allocator release.

A final longer-term approach to incremental optimization that we have been exploring involves the use of randomization to boost the performance of search heuristics [Cesta *et al.*, 2002, Cicirello and Smith, 2002]. This work assumes an iterative sampling approach, where solutions (schedules) are generated repeatedly through non-deterministic application of base search heuristics. The key idea is to bias the degree of randomness to the level of discrimination provided by the heuristic in different decision contexts; when the heuristic clearly favors one choice over others, choose more deterministically, and vice versa when the heuristic is less informing choose more randomly. These techniques have yielded strong performance on classical scheduling problems but have not yet been extended to deal with the full range of constraints in the Barrel Master allocation problem.

5 Project History and Status

The AMC Allocator was developed initially under sponsorship of the Joint Department of Defense Advanced Research Projects Agency/USAF Research Laboratory (DARPA/AFRL) Planning Initiative Program. Following early success within this core technology development program in the area of constraint-based planning and scheduling, we were put in contact with AMC in hopes of identifying potential avenues for future technology transition. The AMC Barrel Master allocation problem was selected as an initial application focus, and several successively more sophisticated Allocator prototypes were developed over the following two years. Following positive evaluation by AMC personnel in 1999, a decision was made to transition the Allocator technology into operations within the TACC. In April 2000, project sponsorship shifted from DARPA/AFRL to AMC, with research and development work continuing under subcontract to Northrop Grumman Information Technology (NGIT), the prime contractor of AMC's operational Consolidated Mobility Planning System (CAMPS). This collaboration with NGIT is ongoing.

A version of the Allocator supporting airlift allocation was first incorporated as a module of the AMC's Consolidated Air Mobility Planning System (CAMPS) for user testing and experimental evaluation in September 2000. In this preliminary insertion, where the goal was to gain experience and build user acceptance, provisions were made to shadow the operational CAMPS business process rather than directly manipulate the data in the CAMPS data base. In retrospect, this decision turned out to be ill-advised, since users essentially had to do "double work" to make use of the new technology. Nonetheless, considerable insight was gained regarding how to better insert the Allocator technology into current CAMPS business processes. Following this feedback, and a programmatic decision to refocus on the Tanker Barrel's problem and processes, an enhanced AR (Air Refueling) Allocator module as been developed and integrated into CAMPS. In May 2003, this version of the Allocator is scheduled for initial operational release as part of CAMPS Release 5.4. At this point, Tanker Barrels will commence training for actual operational use.

6 Summary and Current Directions

In this paper we have described the incremental, constraint-based scheduling model that underlies the design of the AMC Allocator. The model is particularly well suited to the continuous nature of the AMC Barrel Master allocation problem, where newly planned missions must be integrated into an evolving global schedule on an ongoing basis. In this environment, it is important to manage solution change, as it can be unproductive and costly to continually redirect various executing agents. The incremental scheduling and allocation procedures provided by the Allocator directly address this issue. When possible, they allow new missions to be incorporated without change to previous allocation decisions. In situations where non-disruptive allocation is not possible, the procedures support generation of multiple change options and promote selective, controlled manipulation of previous allocation decisions. These capabilities are provided in a real-time, interactive decision-support framework.

Our current research aims at broadening the scope and applicability of the Allocator functionality. One major thrust aims at integration with "state of the world" information updates. Although current application of the technology within the Barrel Master office aims primarily at supporting the AMC mission planning process in advance of execution, the incremental scheduling and allocation capabilities provided by the Allocator are equally applicable in the context of execution management, where unexpected events (e.g., aircraft failures, base closures) often necessitate reassessment of current plans. Furthermore, there is increasing momentum within AMC and the TACC toward tighter integration and information flow between planning and execution offices, which can be expected to provide the basis for more execution-driven planning and allocation processes. Our work here is focusing on (1) techniques for reconciling actual execution status with expectations contained in the current schedule and recognizing the need for change, (2) tools for visualizing current replanning problems and opportunities, and (3) strategies and heuristics for responding to unexpected execution events.

A second thrust of our current research is on expanding the core search procedures to incorporate a larger and more diverse set of mission planning and scheduling constraints. In some cases, these model extensions correspond directly to the additional types of constraints that come into play in closing the loop with execution. Others relate to a desire to explore the applicability our current incremental constraint-based search approach to upstream mission planning problems.

7 Acknowledgements

Many individuals have contributed to the development and deployment of the AMC Allocator software system. The authors would like to thank Susan Buchman, Mark Burstein, Charlie Collins, David Crimm, Chuck Dearth, Brian Gloyer, David Hildum, Ora Lassila, Dirk Lemmermann, Gary Pelton, Lindy Resnor, Mark Shieh, Seppo Torma, Rod Thornton, Crystal Whittenburg, and Joe Zhou. The research reported in this article was sponsored in part by the Department of Defense Advanced Research Projects Agency and the US Air Force Research

Laboratory under contracts F30602-97-2-0227 and F30602-96-D-0058, by the USAF Air Mobility Command under subcontract 10382000 to Northrop-Grumman Information Technology, and by the Robotics Institute at Carnegie Mellon University.

References

[Baptiste *et al.*, 2001] P. Baptiste, C. LePape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishing, Boston MA, 2001.

[Barnhart *et al.*, 1998] C. Barnhart, N.L. Boland, L.W. Clarke, E.L. Johnson, G.L. Nemhauser, and R.G. Shenoi. Flight string models for aircraft fleeting and routing. *Transportation Science*, 32(3):208–220, August 1998.

[Becker and Smith, 2000] M.A Becker and S.F Smith. Mixed-initiative resource management: The amc barrel allocator. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, pages 32–41, Breckenridge CO, April 2000. The AAAI Press.

[Becker, 1998] M.A. Becker. *Reconfigurable Architectures for Mixed-Initiative Planning and Scheduling*. PhD thesis, Graduate School of Industrial Administration and The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 1998.

[Cesta *et al.*, 2002] A. Cesta, A. Oddi, and S.F Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136, 2002.

[Cheng and Smith, 1997] C. Cheng and S.F. Smith. Applying constraint satisfaction techniques to job shop scheduling. *Annals of Operations Research, Special Issue on Scheduling: Theory and Practice*, 70:327–357, 1997.

[Cicirello and Smith, 2002] V. Cicirello and S.F Smith. Amplification of search performance through randomization of heuristics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 124–137, Ithica NY, September 2002. Springer Verlag.

[Clarke *et al.*, 1996] L.W. Clarke, E.L. Hane C.A., Johnson, and G.L. Nemhauser. Maintenance and crew considerations in fleet assignment. *Transportation Science*, 30:249–260, 1996.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1):61–96, May 1991.

[Kramer and Smith, 2002] L. Kramer and S.F Smith. Optimizing for change: Mixed-initiative resource management with the amc barrel allocator. In *Proceedings of the 3rd International Workshop on Planning and Scheduling for Space*, Houston,TE, October 2002.

[Kramer and Smith, 2003] L. Kramer and S.F Smith. Maximizing flexibility: A retraction heuristic for oversubscribed scheduling problems. In *Proceedings 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August 2003. Springer Verlag.

[Rushmeier and Knotogiorgis, 1997] R.A. Rushmeier and S.A. Knotogiorgis. Advances in the optimization of airline fleet assignment. *Transportation Science*, 31(2):159–169, May 1997.

[Smith *et al.*, 1996] S.F. Smith, O. Lassila, and M.A. Becker. Configurable, mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, Menlo Park, 1996.

[Zhou and Smith, 2002] Q. Zhou and S.F Smith. A priority-based pre-emption algorithm for incremental scheduling with cumulative resources. Technical Report CMU-RI-TR-02-11, Robotics Institute, Carnegie Mellon University, June 2002.