A Multiagent Society for Military Transportation Scheduling

David Montana, Jose Herrero, Gordon Vidaver, and Garrett Bidwell
BBN Technologies
10 Moulton Street, Cambridge, MA 02138
{dmontana,jherrero,gbidwell,gvidaver}@bbn.com

Abstract

We are in the process of building a proof-of-concept automated system for scheduling all the transportation for the United States military down to a low level of detail. This is a huge problem currently handled by many hundreds of people across a large number and variety of organizations. Our approach is to use a multiagent society, with each agent performing a particular role for a particular organization. Use of a common multiagent infrastructure allows easy communication between agents, both within the transportation society and with external agents generating transportation requirements. We have demonstrated the feasibility of this approach on several large-scale deployment scenarios.

1 Introduction

An important open issue in scheduling is how to handle problems that are so large and complex that they cannot be solved without some decomposition into subproblems. Of particular interest are those cases for which there are significant dependencies and dynamic interactions between the subproblems. The military transportation problem, described in Section 1.1, is such a problem.

We use multiple scheduling agents interconnected to form a multiagent society in order to decompose this scheduling problem into many interacting subproblems. A similar approach has been previously employed for scheduling of manufacturing supply chains (Barbuceanu & Fox, 1996). Section 1.2 introduces some key issues for multiagent systems.

While the focus of the paper is on the interaction between the different agents, the quality of the schedules produced is ultimately limited by the performance of the individual agents. In Section 1.3 we discuss our approach to implementation of the individual agents, with an emphasis on how we have used genetic algorithms for our more sophisticated agents.

1.1 Military Transportation Scheduling

Scheduling all the transportation for the United States military is an extremely complex problem requiring a huge amount of time, resources and people. For example, building a partial transportation schedule for

Operation Desert Shield / Desert Storm required more than a month. The complexity of the problem stems from the following reasons:

Large Number of Tasks and Resources: In order to support its transportation needs, the United States military has hundreds of planes, dozens of ships, thousands of rail cars and trucks, a variety of seaports and airports, tens of thousands of containers, and thousands of people maintaining and operating these assets. During the seven months of deployment for Operation Desert Shield / Desert Storm, the transportation system moved over half a million passengers, over half a million short tons of air cargo, and over three million short tons of sea cargo not including petroleum, oil and lubricants (which added another six million short tons) (Matthews & Holt, 1996). In peacetime, there is much less to transport, but there is a whole set of different tasks and requirements involving training and readiness. Scheduling all these resources in order to handle all these tasks down to the level of detail of who is flying which plane and what item goes on which truck is a problem of huge scale.

Large Variety of Scheduling Subproblems: The overall scheduling problem can be divided into its air, sea, and land components, each of which can be further subdivided into smaller scheduling problems. The air component requires scheduling of aircrews, aircraft, maintenance crews and equipment, air spaces, dropzones, runways, etc. The sea component requires scheduling of ships, space within the ships, docks, resources of the ports for loading and unloading, etc. The land component requires scheduling of trucks/convoys, road access, trains and commercial trucks, etc. Compounding the variety of scheduling problems is the fact that most of these subproblems are replicated many times but with variations in the business rules for each instance. For example, there are approximately 100 different air squadrons each scheduling their own crew members, but each have slightly different training requirements that their crew members must satisfy.

Need for Coordination: Greatly adding to the complexity of the overall scheduling problem is the fact that there is a large amount of coordination required between the various pieces. For example, scheduling a land move from a fort to a seaport requires coordination with:

- the truck packing scheduler at the fort
- the truck unpacking and ship loading schedulers at the seaport
- the ship scheduler (to ensure that the right ship is at the seaport)
- other land move schedulers (to ensure that they are not combining to overload the commercial roads with military traffic)

Need for Dynamic Rescheduling: As with most scheduling problems, particularly ones of this size, things rarely go according to plan. For example, traffic can delay movements, people can get sick, vehicles can get broken, etc. When the inevitable deviation from the schedule occurs, the schedule must be redone to adjust for these deviations. The process of monitoring actual performance against the ideal schedule and making adjustments is an inherent part of the scheduling process.

The transportation needs of the U.S. military are currently coordinated by the United States Transportation Command (USTC). The air, sea and land components are coordinated by the Air Mobility Command (AMC), Military Sealift Command (MSC), and Military Traffic Management Command (MTMC) respectively. Under these high-level organizations are approximately

- 30 air wings scheduling planes, maintenance crews, airport operations, etc.
- 100 air squadrons scheduling aircrews
- the forts and depots each doing their own land move scheduling

- the seaports each scheduling their own operation

(There are few enough ships that scheduling them requires no subordinate organizations.)

1.2 Multiagent Systems

A multiagent system is a type of distributed system. (An introduction to multiagent systems is (Weiss, 1999).) There are a variety of opinions on what distinguishes an agent-based system from an ordinary distributed system, such as a set of personal computer clients linked with data, file and print servers. For our purposes we identify two distinguishing characteristics. First, multiagent systems require a greater degree of uniformity among its different components (i.e, agents) in terms of both behavior and communication. Second, agents, unlike ordinary modules in distributed systems, continuously attempt to achieve certain goals without the need for outside invocation.

The field of multiagent systems has such a large and varied history that we cannot attempt to summarize the entire field. Instead, we will highlight a few key issues in the field that directly apply to our work and briefly discuss how we have handled these issues.

Off-The-Shelf vs. Custom Infrastructure: There are a variety of general-purpose multiagent architectures and software packages that are available. Probably the most commonly used and time-tested of these is the Knowledge Query and Manipulation Language (KQML) for communication together with the Knowledge Interchange Format (KIF) for data representation (Genesereth & Ketchpel, 1994), sometimes known jointly as the Agent Communications Language. However, there are many others such as MACE (Gasser, Braganza & Herman, 1988). As detailed in Section 2.1, we have chosen to develop our own infrastructure rather than use any of these. The reasons are that the off-the-shelf packages do not have all the features we require and are not tested to scale to anywhere near the size of the system we anticipate.

Message Passing vs. Shared Dataspace: These are two competing views on how agents communicate with each other in order to coordinate their actions and share data. In the message passing approach, agents communicate and share data by sending messages back and forth. Each agent has its own internal data, state and goals and can only learn of another agent's data to the extent that they communicate directly with each other. In the case of agents cooperating to solve a big problem (which is the case we are considering), the only way to see the full solution is to piece together the data distributed among the different agents. KQML implements a message passing scheme (Genesereth & Ketchpel, 1994).

In the blackboard system approach, agents utilize a common shared dataspace (Corkill, Gallagher & Murray, 1986). Agents post into that dataspace any data/knowledge that might be of interest to other agents or the external world. Agents can also read what other agents have written. The problem with the blackboard system approach is that it has scalability issues. It is infeasible for thousands of agents to be writing to and reading from the same dataspace simultaneously. Even beyond the most immediate issue of how to synchronize access to the shared data by many distributed agents, there is the issue of information overload, with agents having easy access to more data than they may be able to handle.

While message passing avoids the currently intractable problem of efficiently synchronizing access to shared data across many agents, there are some times when agents need a shared dataspace. For example, consider one agent that makes a schedule and another agent that monitors the schedule for deviations. It is best if they both work off the same copy of the schedule. While the data sharing in our multiagent infrastructure is primarily based on message passing, there is support for a small number of agents to have a shared data space, hence making it a type of hybrid.

Shared Ontologies: An ontology is a set of classes, relations, functions, etc. representing knowledge of a particular domain. Having a shared ontology that all the agents utilize in their inter-agent communication is critical to successful communication because a shared ontology provides the common format in which to express data and knowledge (Falasconi, Lanzola & Stefanelli 1996). Ontologies are by their nature specific to a particular domain. There are tools and extensible baseline ontologies, such as KIF (Genesereth & Ketchpel, 1994) and Ontolingua (Gruber, 1992), that make the task of constructing an ontology simpler. However, we have started from scratch in developing an ontology for the military logistics domain. (As explained below, our transportation society is just part of a larger logistics society.) The two big issues we have had to face are completeness, i.e. ensuring that all concepts are represented, and extensibility, the ability to add new concepts without causing major upheaval for the system.

Agent Specifications - External View vs. Internal Structure: Another issue in multiagent systems is how to define the specifications for what constitutes an agent. Some approaches, such as the belief-desire-intention model (Georgeff et al., 1998), specify how an agent should actually function internally. Other approaches only define a set of interfaces that the agents must implement, what might be known in software engineering as an Application Program Interface. The advantage of the latter approach is that it allows maximum flexibility in the development of agents, including the ability to wrap existing applications as agents (Genesereth, Singh & Syed, 1994). It is for this reason that our architecture uses this approach.

Static vs. Dynamic Configuration: Some multiagent systems have a fixed set of communication interconnections between agents, while others allow the interconnections to form and break as the system runs. An example of the latter is contract nets, where agents interconnect dynamically based on services needed and services provided (Davis & Smith, 1983). We have not yet needed dynamic reconfiguration, since the interrelations between the military organizations we have modeled so far are fairly static. Hence, we have used a static configuration based on its greater simplicity and efficiency (not requiring the overhead of continually having the agents find and select other agents). We anticipate needing a more dynamic topology in the future.

System Stability: The dynamics of multiagent systems, particularly those that react to changes in the external world, are difficult to predict and control. Even when the individual agents themselves are stable and predictable, the interactions between agents can easily lead to system-level instability and chaos (Hogg & Huberman, 1991). One set of experiments indicates that limiting the availability of information to the small number of agents to which it most directly applies can lead to system stability (Sen, Roychowdhury & and Arora,1996). We have done this anyway for issues of scalability (see above). So far, we have been able to make our society stable by having the agents attempt to minimize their changes to the schedule. As we discuss in Sections 3 and 4, this is not likely to be sufficient once we start driving the system closer to its capacity and with more changes to the external world.

Collective Intelligence vs. Intelligent Agents: There are two different bodies of research on multiagent systems. The distributed artificial intelligence approach examines societies where the agents themselves are “intelligent”, and the “intelligence” of the group is largely a sum of the “intelligence” of the parts. The artificial life approach examines societies consisting of very simple agents that collectively display a complexity not evident in any of the parts, a concept known as “collective intelligence”. Examples of the artificial life approach are the flocking behavior of Boids (Reynolds, 1987), the traffic signal control of

Montana and Czerwinski (1996), and the approach to scheduling outlined in (Sen & Durfee, 1998). Our approach to transportation scheduling, as well as all the other work referenced above, is of the distributed artificial intelligence variety. We only use agents to decompose the problem enough to be solved by other types of algorithms within the agents. The reasons are simplicity (of algorithms and software) and effectiveness; once we have a reasonably sized problem assigning relatively homogeneous tasks to relatively homogeneous resources, there are other highly effective techniques for scheduling, as we now discuss.

1.3 Scheduling Algorithms for the Individual Agents

Even after using a multiagent scheme to decompose the very large scheduling problem into smaller atomic problems, these smaller problems still often present great difficulties in terms of finding a legal and high-quality schedule. Some of the reasons for the difficulties are (Montana, 1998):

- **Large and complex search spaces:** The size of the search space scales superexponentially with the number of tasks and resources, of which there are still dozens or hundreds even in the atomic problems. The fitness landscape is both discrete and generally rugged.
- **Dynamically changing problems:** Any schedule of future events makes assumptions about how these events will unfold. For most real-world scheduling problems, the schedule rarely gets executed exactly as planned because of some unanticipated deviations. A scheduler needs to adapt its schedules in real time to take these deviations into account.
- **A variety of constraints:** There are two types of constraints in scheduling problems. Hard constraints are those which must be satisfied for the schedule to be considered legal, while soft constraints are essentially preferences. Different problems have different constraints of different forms, often not linear or even continuous. A general approach to scheduling must be flexible enough to adapt to these different constraints without sacrificing the efficiency and effectiveness of its search.

Traditional operations research (OR) techniques, such as branch-and-bound and integer programming, do not provide the scalability or flexibility required by our atomic scheduling problems. We have instead used two other types of scheduling algorithms for the individual agents. For those agents where either the scheduling logic is inherently simple or we have not had time yet to implement a complex scheduling algorithm, we have used simple problem-specific heuristics, often rule-based.

When implementing more complex scheduling algorithms for the agents, we have used genetic algorithms. While other techniques for optimized scheduling are available (e.g., simulated annealing, tabu search, and ant colony optimization), the experience that we and others have had indicates that genetic algorithms are a generally effective approach to solving difficult scheduling problems. Genetic algorithms have been applied to such problems in a wide variety of domains including

- jobshop/flowshop scheduling (Bierwirth et al., 1995)
- urban transit systems (Deb & Chakroborty, 1998)
- supply chain management (Hart, Ross & Nelson, 1998)
- exam timetabling (Burke, Newall & Weare, 1998)
- scheduling computing tasks (Gonzalez & Wainwright, 1994)
- scheduling laboratory equipment (Syswerda, 1991)
- crew scheduling (Levine, 1996)
- maintenance/rehabilitation scheduling (Halhal et al., 1997)
- talent/project scheduling (Nordstrom and Tufekci, 1994)

The reason for genetic algorithms' success at a wide and ever growing range of scheduling problems is a combination of power and flexibility. The power derives from the empirically proven ability of evolutionary algorithms to efficiently find globally competetive optima in large and complex search spaces. The flexibility is based on the ability to tailor the representation, operators, initialization method, etc. to fit the problem/domain.

We have developed a general approach to scheduling using genetic algorithms that allows us to relatively easily and quickly produce effective scheduling algorithms for different domains (Montana et al., 1998). We now discuss the major components of this approach:

- **Domain-Specific Representation and Operators:** Genetic algorithms require a representation, i.e. some method of encoding potential solutions to a problem in a data structure known as a "chromosome", and genetic operators that can create a new "child" chromosome by transformation of one or two "parent" chromosomes (with the operators known as "mutation" and "crossover" respectively). Which representation and genetic operators are best depends greatly on the constraints of the problem to be solved (Davis, 1991), and we select a representation and operators well matched to the problem.
- **Multiobjective Evaluation Function:** There generally are multiple evaluation criteria. We therefore utilize an evaluation function that is a linear combination of the different individual criteria, with the weights on the criteria being adjustable to allow different tradeoffs. (A variety of other ways of doing multiobjective optimization using evolutionary algorithms are discussed in (Coello Coello, 1999).) The criteria can be of any form and are often nonlinear and discontinuous.
- **Heuristic Initialization:** Grefenstette (1987) and Davis (1991) have shown that using domain knowledge to initialize the population with better-than-random individuals can potentially greatly increase the performance of the genetic algorithm as compared to initialization with random individuals. Burke, Newall and Weare (1998) have shown that heuristic initialization is most successful when the initialization procedure makes the initial population maximally diverse. We often get large performance gains by utilizing heuristic initialization that maintains diversity.
- **Reconciliation and Reseeding for Dynamic Rescheduling:** We have the genetic algorithm continually executing and creating new schedules. During any single run of the genetic algorithm, the problem remains fixed. At the end of the run, the best schedule is quickly reconciled, using a heuristic algorithm, with any changes to the problem/data reported from the real world before the schedule is published. The next run uses this published schedule as one of the members of its initial population but with the rest generated using the standard initialization procedure. A big benefit to this process is that periodically diversity is reintroduced into the population. This appoach to dynamic rescheduling is similar to that of (Bierwirth et al., 1995) and (Gonzalez & Wainwright, 1994).

2 The Multiagent Society

In this section we describe our multiagent society for transportation scheduling. In Section 2.1 we discuss the multiagent infrastructure that we have utilized. In Section 2.2 we outline the structure of the society, describing the various agents and how they interconnect. In Section 2.3 we illustrate how a single injected task propagates through the society.

Table 1: An example of a task object

verb:	transport
direct object:	Some Cargo
from:	Fort Stewart
to:	Ramstein AFB
ready to leave date:	04/15/99
earliest arrival date:	05/07/99
best arrival date:	05/09/99
latest arrival date:	05/11/99

2.1 Multiagent Infrastructure

The custom multiagent architecture on which we have built our society of transportation agents was developed by another group at BBN as part of DARPA's Advanced Logistics Program (ALP). It was designed to allow many different groups to develop agents solving different pieces of the military logistics problem in an attempt to automate the entire process of logistics planning and scheduling. (In fact, as we discuss in Section 2.3, our transportation society is part of a larger logistics society.) We now describe certain of its key components.

Agents and Clusters: An agent is simply a module that implements the agent application program interface (API). This API provides standardized mechanisms for an agent to interact with the rest of the society, including

- expansion (and aggregation) of tasks into subtasks
- assignment of tasks to assets
- assignment of tasks to other agents
- reception of tasks from other agents
- sharing of data with other agents

While the syntax of the agent API is beyond the scope of this paper, we will describe each of these mechanisms for agent interaction in the remainder of this section.

A cluster consists of a group of one or more agents along with a shared dataspace of tasks, assets and data. Each agent must belong to one, and only one, cluster. Agents do not interact directly with other agents but rather with the cluster infrastructure. Agents read tasks, assets and other data from the cluster and write updates to this data back to the cluster. Mechanisms within a cluster control access to the shared dataspace to ensure the integrity of the data.

Tasks: Tasks are what the agents need to schedule. A task consists of a verb, a direct object, and an indefinite number of prepositional phrases, with each phrase consisting of a preposition and an indirect object. Table 1 shows an example of a task object. The verb represents the action that is being requested. The direct object is the asset or set of assets (represented via the Logical Data Model) on which the action will be executed. The remaining clauses represent the constraints (both hard and soft) on how the task should be performed.

Tasks can be handled by agents in one of three ways:

- Allocation: An agent can simply schedule one or more assets to the task at particular times. (If the time is not yet fully constrained, the allocation selects a time within the constraints.) For example, the task to provide a plane for an airlift mission could be allocated to a plane. The plane would then be unavailable for other purposes during the duration of the mission.
- Expansion: An agent can decompose a task into subtasks. For example, the task to schedule an airlift mission could be expanded into a task to provide a plane for the mission and a task to provide a crew for the mission.
- Aggregation: An agent can group a set of tasks into a single aggregate task. For example, three different tasks to sealift equipment from three different brigades, all of which could fit in a single ship, could be aggregated into a single sealift mission.

Expansion and aggregation transform tasks into "children" tasks. The transformation process ends when a task is allocated. If the society is properly structured, all threads of this transformation process end with an allocation.

While most tasks are a child to some parent task, there have to be some tasks that are at the root of a task hierarchy. Such tasks are created based on inputs from human users, other systems, or external databases.

Inter-Agent Communications: As we discussed in Section 1.2, our multiagent infrastructure uses a hybrid of shared dataspace and message passing communications. Inter-agent communication via a shared dataspace is restricted to agents within the same cluster. All the tasks, assets and data that an agent creates or modifies within a cluster are accessible by all other agents within that cluster, and hence communicated to the agents monitoring or utilizing this data. As an example of shared dataspace communication, one agent can read the state of the available assets and write this data into the cluster's shared dataspace (after translating into Logical Data Model objects), and another agent can read this data for use in deciding which assets to schedule to which tasks.

Communication by message passing occurs between clusters. An agent in one cluster assigns a task to another cluster (and not a particular agent in that cluster), using a mechanism similar to that for allocating a task to an asset. Assigning a task to a cluster sends the task to that cluster. The receiving cluster puts the task in its shared dataspace and passes the task on to the agent that registered to handle tasks of this type (with an error occuring if there is no such agent). An agent can choose to assign a task only to those clusters that have registered as providing service to the agent's cluster. Figure 1 illustrates this process. Note that any task that is not allocated, expanded or aggregated should be assigned to another cluster.

There is a secondary, but important, type of communication associated with assignment of a task to another cluster. The receiving cluster passes back to the assigning cluster an allocation result that tells whether it was capable of handling this task, and if so a summary of how it was handled. The assigning cluster has the potential to rescind the task at any time based either on bad allocation results or future dynamic changes to the data. The rescinding mechanism is particularly important to allow dynamic updating of the schedule, because decisions will change as the problem and the environment change.

Logical Data Model (LDM): The LDM is a shared ontology (i.e., set of objects, relationships, methods, etc.) and shared knowledge base. While agents can store data in whatever form they want internally, any data they write back to the cluster for sharing with other agents must be LDM objects. This ensures that the agents share a common representation of data syntax and semantics.

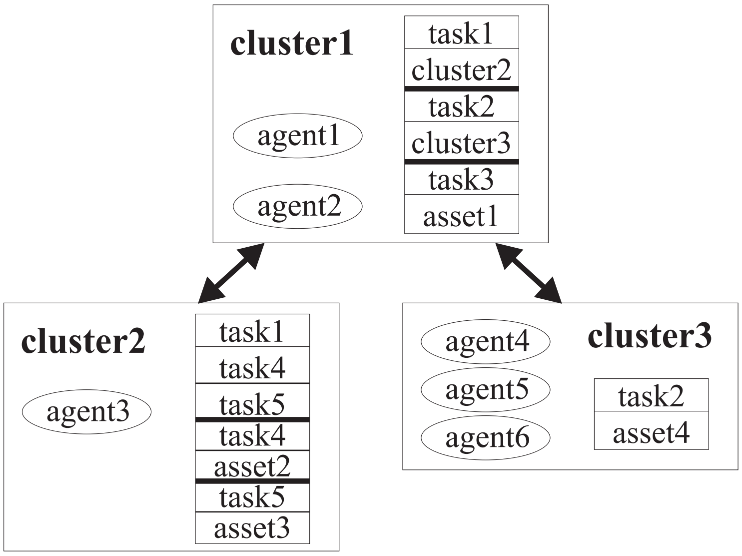

Figure 1: This figure illustrates with a very simple example the operation of the multiagent infrastructure. Cluster2 and Cluster3 have reported for service to Cluster1. Cluster1 has assigned one of its internal assets, asset1, to task3 and has assigned task1 and task2 to cluster2 and cluster3 respectively. Cluster2 has expanded task1 to create two new tasks, task4 and task5, and has assigned task4 to asset2 and task5 to asset3. Cluster3 has assigned task2 to asset4.

The LDM we utilize is geared specifically towards the logistics domain, containing objects such as containers and planes and data such as the cargo capacity of a C-141 cargo plane and the cruise speed of a particular type of ship. There are a lot of domain-specific objects and knowledge that need to go in the LDM, with the amount increasing as we expand the functionality of the society. Creation and maintenance of the LDM is a large and interesting task, but the techniques used to make the task manageable are beyond the scope of this paper.

2.2 Structure of the Multiagent Society

Our transportation society is currently composed of fifteen different clusters representing a cross-section of the different organizations that plan and execute military deployments. Figure 2 shows the hierarchical arrangement of the transportation society. The fifteen different clusters are instances of nine different types of clusters. In the future we hope to expand the current set of clusters representing a cross-section of the full set of transportation scheduling organizations into a full-scale transportation society. Figure 3 shows the approximate size and structure of such a full-scale society. We now discuss briefly the functionality of each of the nine cluster varieties of the current society plus those missing from the current society.

Global Mode: The Global Mode cluster decides if a item will be transported by air or by sea or neither (i.e., requires only a ground move). It is a function performed in real life at the headquarters of the U.S. Transportation Command. The heuristics used by the scheduling agent are currently very simple. If the distances are short enough and both end points are inside the continental United States, then the item moves only by land. Otherwise, the cluster must decide whether to send the item by sea or by air (in

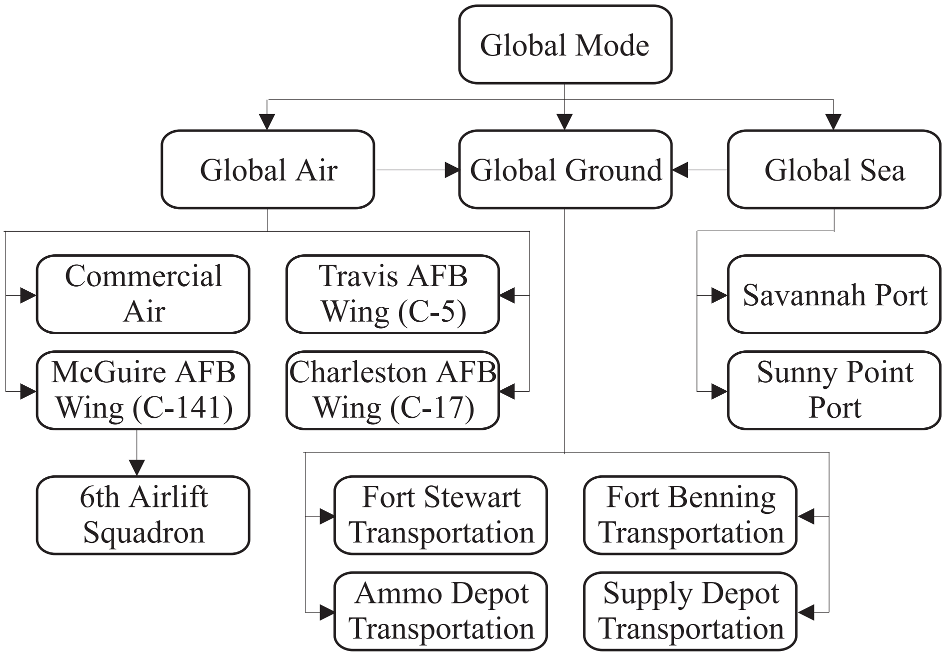

Figure 2: Clusters of the current transportation scheduling society

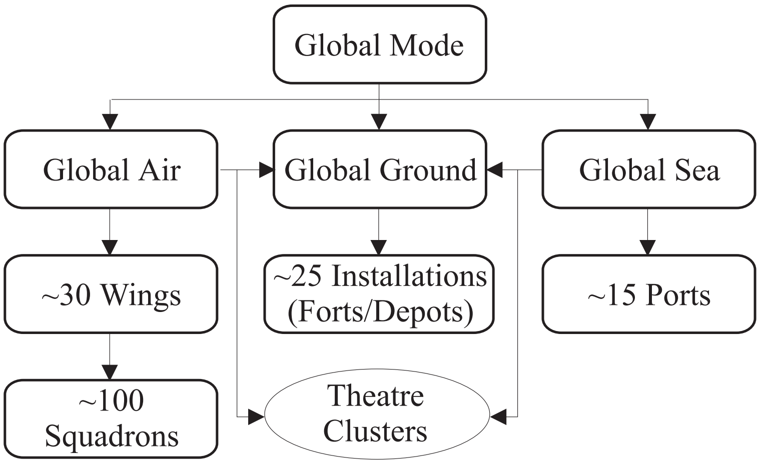

Figure 3: Size and structure of a full-scale transportation scheduling society

addition to the land moves required at either end). All people travel by air. All cargo items are assumed to go by sea as the default. The item travels by air only if it fits on a plane and the time constraints are such that it cannot go by sea. In the future, the scheduler agent will have to take a variety of other factors, including the shipping cost, into account when deciding the mode of transportation of a requirement.

Global Air: The Global Air cluster creates airlift missions to handle the requirements it receives to transport equipment and personnel by air. It is a function performed in real life by the Air Mobility Command. (Note that the Air Mobility Command also handles all requests for air refueling, and so must this cluster eventually.) When forming missions, it has a number of constraints to consider, including:

- There are a limited number of plane of each type (we currently consider C-141's, C-17's, and C-5's), and the planes are stationed at different air bases.
- Each plane has limited cargo capacity with the values of the limits depending on the type of plane.
- Certain items can only travel on certain types of aircraft (e.g., tanks require a C-5) or are preferrably assigned to certain types of aircraft (e.g., passengers on commercial aircraft).
- Airports, both refueling points and final destinations, have a maximum number of aircraft allowed on the ground at once.
- Items traveling on the same mission should have the same pickup and dropoff locations.
- The movements have timing requirements.

We have been able to use a very simple heuristic algorithm to build missions that satisfy the constraints, because we have worked so far with a relatively small number of requirements. When handling all the air transportation requirements of the full U.S. military, this is a difficult problem requiring a sophisticated algorithm. In the future, we plan to wrap the agent API around a system currently in development in order to incorporate its sophisticated scheduling algorithm into this cluster.

Global Sea: The Global Sea cluster creates sealift missions to handle the requirements it receives to transport items by sea. It is a function performed in real life by the Military Sealift Command. When forming missions, it has a number of constraints to consider, including:

- There are a limited number of ships of each type.
- Each ship has limited cargo capacity.
- Certain items can only travel on certain types of ships (e.g., ammunition on special ammunition ships) and must be loaded at certain ports (e.g., ammunition, and only ammunition, at Sunny Point).
- Items traveling on the same mission should have the same pickup and dropoff locations.
- The movements have timing requirements.

We have been able to use a simple heuristic so far because of the current small size of the problem. We are working on developing a genetic algorithm approach.

Global Ground: The Global Ground cluster coordinates the land moves in the continental United States. This is a function performed in real life by the Military Traffic Management Command. The primary function of this cluster is to distribute a set of routes to be used with each land move. Having the route selection done at this cluster rather than the lower-level (installation transportation) clusters that handle the details of the land move provides a mechanism to keep multiple land moves from interfering with each other. Route optimization of a number of varieties have been extensively studied, including some that use genetic algorithms (Gabbert et al., 1991; Thangiah, 1995). Our problem differs from standard route optimization problems in that we are looking for a set of routes, rather than a single route, between each source-destination pair. The main objectives are to

- minimize the travel time of the routes
- maximize the joint capacity/bandwidth of each set of routes given tight limits on the amount of military traffic on different roads (note that the limit on traffic for the majority of roads is actually zero, hence allowing us to eliminate most roads from consideration)

We perform the route optimization using the genetic algorithm described in (Montana et al., 1999). [The current trend for land moves within the U.S. is strongly towards using commercial carriers rather than military transportation, and hence this cluster may eventually become more of an electronic commerce broker. However, its basic functionality of finding sets of routes will still be required for the clusters that plan ground movements in theatre (see below), where there are not trusted and reliable commercial carriers.]

Airlift Wing: In the current society, the airlift wing clusters (representing the wings at McGuire, Charleston and Travis Air Force Bases) provide the planes for the airlift missions and request aircrews from their subordinate airlift squadrons. This is a simple scheduling problem handled by a simple heuristic scheduling algorithm. The real challenge lies in the future as we implement the full scheduling functionality of a wing, including scheduling of airspace, runways, loading equipment and personnel, and (most challenging) maintenance time for planes and maintenance resources (personnel and equipment). To do so, we will need more agents and possibly more clusters. As we expand the number of clusters, we will also have to handle the case when a wing has multiple subordinate squadrons with the same aircraft type and hence needs to balance the aircrew requests among these subordinates.

Commercial Air: The Commercial Air cluster schedules planes from commercial airlines to transport passengers and small amounts of cargo. The current simplistic version assumes (incorrectly) that a fixed number of planes are fully available to the military. We will eventually replace this with an "agentized" (i.e., with a wrapper implementing the agent API) version of the "virtual airlines" system currently under development by another contractor. This system automatically searches a commercial airline's database and charters planes not already scheduled for commercial use.

Airlift Squadron: The squadron clusters allocate an aircrew to each assigned mission. This is a hard scheduling problem because of the large number of contraints, both hard and soft, that must be satisfied. These constraints vary from squadron to squadron based on factors such as the type of aircraft, whether the squadron is active duty or guard/reserve, and the discretion of the local commanding officer. Some of the hard constraints for military crew scheduling are

- Each mission requires a certain number of crew members of each type (pilot, co-pilot, flight engineer, navigator, loadmaster, boom operator), known as the crew complement.
- For a particular position on a particular mission, there is a well-defined set of qualifications required of a crew member filling that position.
- Each crew member requires a certain amount of crew rest time before and after each mission.
- There are certain limits on the maximum number of hours that a crewmember can spend in the air during any 30-day or 90-day period.
- There are a variety of training requirements that each crewmember must continually satisfy to stay up-to-date with his/her training.

Some of the soft constraints for military crew scheduling are

- Provide crews for as many missions as possible, particularly the high-priority missions.
- Minimize the "ripple effects" in the schedule when new missions arrive, missions get delayed or exteded, or crewmembers get sick.
- Ensure that crewmembers all fly on the right number and variety of missions to keep current.
- Spread the load evenly among the crewmembers.

Commercial airrew scheduling is a problem that has been solved by a variety of methods including genetic algorithms (Levine, 1996) and more traditional operations research methods (Stojkovic & Soumis, 1998; Day & Ryan, 1997). We are building a military aircrew scheduler for the U.S. Air Force that will soon be operational. We turned it into an agent by wrapping it with a layer implementing the agent API.

Port Operations: The port operations clusters (Savannah Port and Sunny Point Port) each have three main scheduling agents. One agent develops stow plans to tell where all the items are to be stowed in the ship. A second agent creates loading schedules that tell in what order the items are to be loaded on the ship. A third agent plans the details of how to load the items on the ship. These agents utilize complex modeling of physical structures. This cluster and its agents was developed by another contractor but interacts easily with our clusters and agents via the agent API and the LDM.

Installation Transportation: The installation transportation clusters (Fort Stewart, Fort Benning, Ammo Depot and Supply Depot) schedule the land moves from the forts and supply depots (i.e., installations), where the items originate, to the airports and seaports. (There are other clusters at forts and depots not dealing with transportation; for example, at depots there are clusters to handle inventories of supplies. However, these are separate clusters outside of the transportation society.) This is handled in real life by the Installation Transportation Officer (ITO) at each installation. This cluster is responsible for a variety of aspects of the land move including

- whether to send items by military truck, commercial truck, or train
- how to allocate the items to particular trucks
- how to group the military trucks into convoys
- what time each convoy should leave the installation
- what route from the set sent by Global Ground to assign to each convoy

The objectives are to

- minimize the amount of time that items sit at the port waiting to be loading (i.e., minimize staging)
- minimize the disturbance to civilian traffic on the roads by not sending too many vehicles on a given road within a given time span

We use the genetic algorithm described in (Montana et al., 1999) to perform land move convoy scheduling. [As for the Global Ground cluster, the installation transportation clusters may eventually focus more on contracting with commercial carriers rather than planning their own moves. The basic functionality of convoy building and scheduling will still be needed in theatre.]

Theatre Clusters: The full society contains a major component not in the current society, the clusters for scheduling transportation in the theatre of operations. The current society only schedules the movement to an airport or seaport, but there remains the final leg of the journey, generally by ground but also possibly containing a short air or sea movement, to the final destination. We are close to completion of initial versions of the theatre clusters to add to our transportation society.

2.3 Operation of the Multiagent Society

We now illustrate the operation of our transportation scheduling society as a whole by describing how a sample task propagates through the society. This sample task is to transport all the equipment belonging to a brigade from its home base at Fort Stewart, Georgia to a theatre location in Al Haba, Saudi Arabia. The current date is May 11, 1999, and the task requests that the items arrive somewhere between three and seven days in the future, optimally on the fifth day. Figures 4-10 depict the progressions of operations performed by the different clusters. The incoming arrows on the left of each figure and the outgoing arrows on the right show the flow of tasks between clusters. Note the increasing number of tasks and the increasing level of detail as the tasks flow through the society. Furthermore, observe how similar the structure (syntax) of the operations are despite the very different content (semantics); this regularity of structure is key to allowing relatively easy integration of these agents into a single society.

Global Mode (Figure 4): The point of entry into the society for the task, as for all tasks generated by external agents, is the Global Mode cluster. (While the primary root tasks in the transportation society

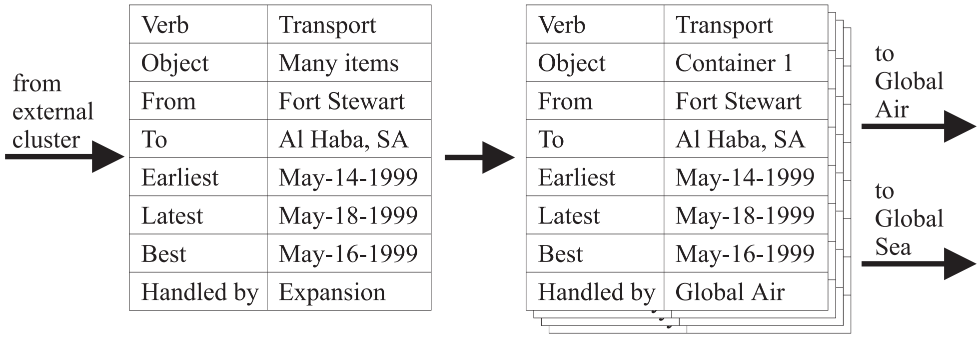

Figure 4: Transformation of the sample task by the Global Mode cluster

are externally generated transportation requirements, there also potentially are internally generated root tasks for secondary activities such as training and maintenance.) The first operation that occurs is that an agent expands the single task to move many items into many subtasks to move single items. Another agent decides for each of these subtasks whether the item should be moved by sea or by air, assigning the task to the Global Sea cluster or Global Air cluster accordingly. Because the items need to arrive in at most seven days, the Global Mode cluster will choose to airlift as many items as possible, with only those that cannot be transported by air traveling by sea instead. Once the Global Sea confirms that the sealifted items will be late, this information is relayed back to the external requesting cluster.

Global Sea (Figure 5): Each task that arrives at the Global Sea cluster from the Global Mode cluster is eventually expanded into three subtasks: one for the land move from Fort Stewart to the port, a second for the loading of the ship at the port, and a third for the sealift portion. (We will soon add an additional subtask for the unloading and in-theatre transport, which will be assigned to the theatre clusters.) Another agent within the Global Sea then aggregates the sealift subtasks, and whatever other such subtasks derived from other top-level tasks it can use to fill a ship, into a single sealift mission task. As part of forming the mission, it selects at which ports to embark and debark (in the case of our example, Savannah and Ad Damman), which ship to use, and which dates to start and end the trip. The port and date information are included in the prepositional phrases of the new aggregated task, while the ship information is included by allocating the aggregated task to the ship. After the sealift mission task is completely specified, the port, date, and ship can be included in the port loading task as prepositional phrases. The port loading tasks are then sent to the appropriate port operations cluster, in this case the Savannah Port cluster. Once the port cluster has scheduled the loading tasks (the details of which are not included in this section because another contractor built this cluster), part of the port cluster's response to the Global Sea cluster is the time (and not just the date) that each loading task will begin. The loading time and port are filled into each land move task before assigning it to the Global Ground cluster.

Global Air (Figure 6): Each task that arrives at the Global Air cluster from the Global Mode cluster is eventually expanded into two subtasks: one for the land move from Fort Stewart to the airport and a second for the airlift portion. (Soon there will be a third subtask for the in-theatre transport. Other potential subtasks, such as runway selection and loading of the plane, are currently omitted because they are relatively simple to perform manually and are hence not a high priority.) Another agent within the Global Air then aggregates the airlift subtasks into a smaller number of airlift mission tasks. Each airlift mission task contains the airports for pickup and dropoff (in the case of the example, Hunter AFB and Riyadh Airport) the date it should be scheduled and the type of plane for which it is targeted. (Recall that

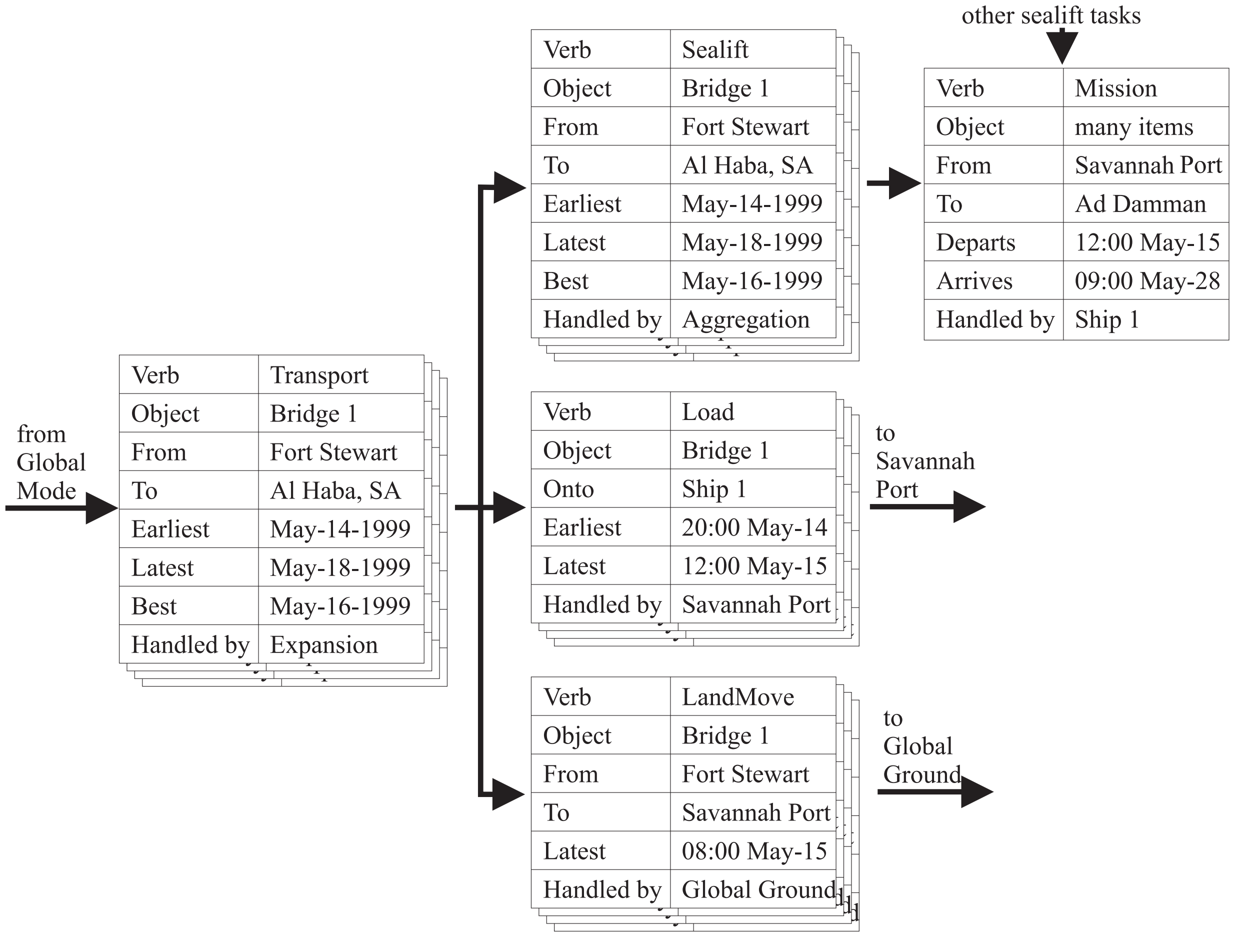

Figure 5: Transformation of the sample task's descendants by the Global Sea cluster

each aircraft type has different capabilities, and therefore missions must be geared towards a particular type of plane.) The airlift missions are then assigned to the airlift wing (McGuire, Charleston, or Travis) that has the required aircraft type. Once the airlift wing cluster has scheduled the time of a mission, it relays this time to the Global Air cluster. The Global Air can the specify the required arrival time for each land move subtask and assign these tasks to the Global Ground cluster.

Global Ground (Figure 7): The Global Ground cluster expands each task it receives into a single subtask. This subtask will be the same as the original task except there will be an extra prepositional phrase specifying the set of routes from which to choose when scheduling the task's move. (The expansion into a single subtask is necessary because a cluster cannot modify a task sent to it by another cluster.) The subtask is then assigned to the appropriate for or depot transportation cluster, in this case the Fort Stewart cluster.

Fort Stewart Transportation (Figure 8): The Fort Stewart cluster, like all installation transportation clusters, receives tasks to transport items to ports. After scheduling the full land move, it allocates to each task the assets it has assigned to it. For items traveling by commercial truck, this is just the trucking company. For items traveling by rail, the assets are the railcar and the train it is part of. For items

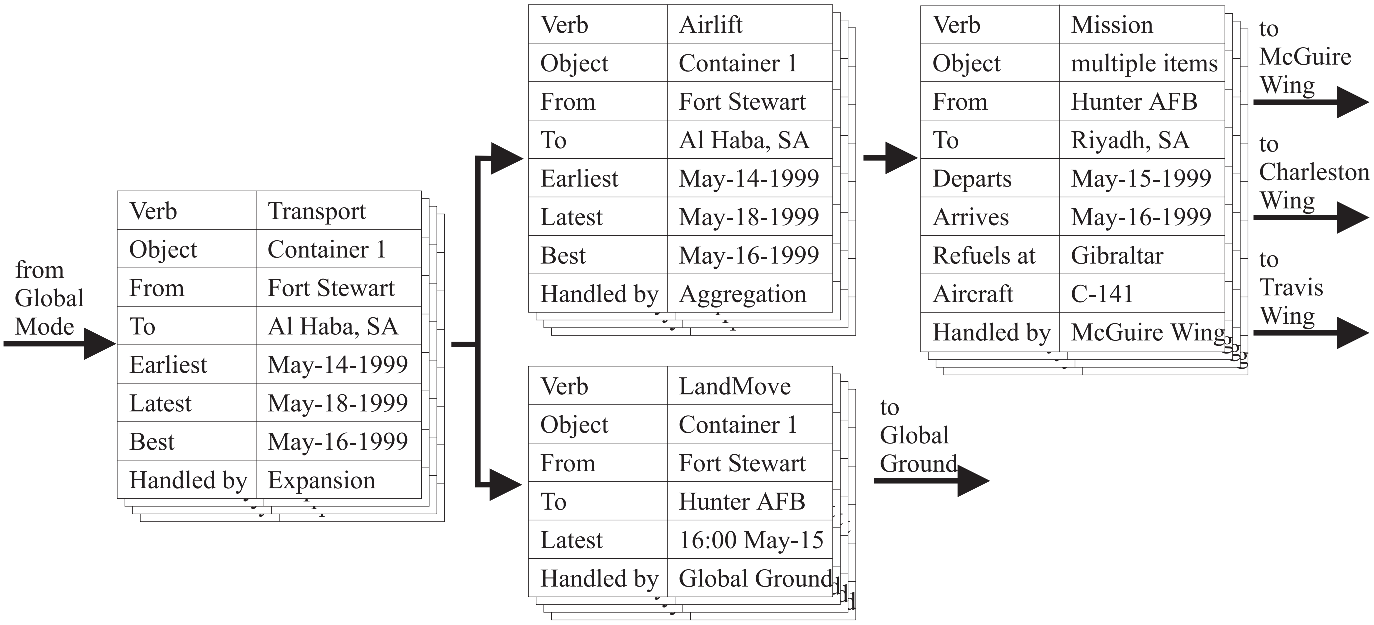

Figure 6: Transformation of the sample task's descendants by the Global Air cluster

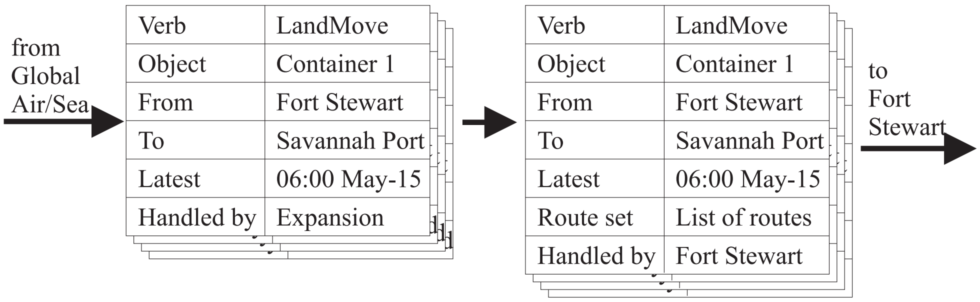

Figure 7: Transformation of the sample task's descendants by the Global Ground cluster

traveling by military truck, the assets are the truck, the convoy it is part of, and the route the convoy takes.

McGuire AFB Wing (Figure 9): We show only the McGuire AFB Wing cluster because the Charleston and Travis clusters, which also receive tasks from the Global Air cluster, work essentially the same. The McGuire AFB Wing cluster, like all airlift wing clusters, receives tasks to handle all aspects of air missions. An agent expands each such task into two subtasks corresponding to the two parts of the air missions the cluster currently handles, one subtask to provide the plane and one subtask to provide the crew. Another agent allocates a plane to each task requesting a plane, while a third agent assigns each crew task to one of the wing's subordinate clusters.

6th Airlift Squadron (Figure 10): The 6th Airlift Squadron cluster, like all airlift squadron clusters, receives tasks to provide crews for air missions. After (re-)building the full schedule, it handles each task

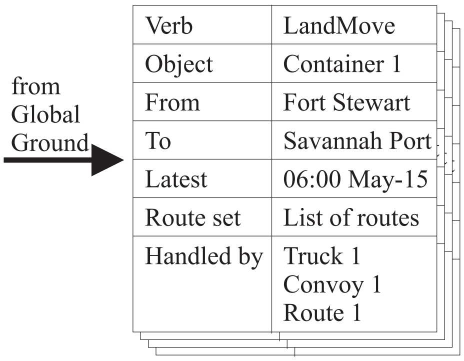

Figure 8: Transformation of the sample task's descendants by the Fort Stewart Transportation cluster

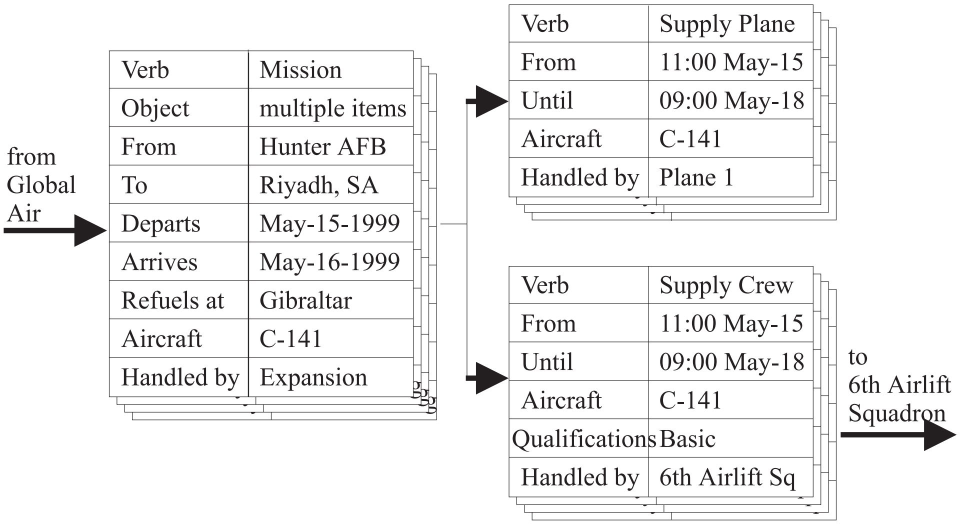

Figure 9: Transformation of the sample task's descendants by the McGuire AFB Wing cluster

by allocating to it the multiple crewmembers scheduled to fly the mission.

3 Results

Our transportation society is part of a larger military logistics society that was demonstrated in January 1999. The goal of the demonstration was to show the ability to automatically plan and schedule the deployment of the 3^{rd} Infantry Division and an Air Expeditionary Force to a location in the Middle East. Our transportation society received requests to transport not only the personnel and equipment of these units but also the personnel and equipment of the required support units and the spare parts and ammunition needed for supplies for these units.

The demonstration was successful both for our transportation society and the larger logistics society. During the demonstration, the transportation society received an estimated 600 tasks for transporting from McGuire AFB Wing

Verb	Supply Crew
From	11:00 May-15
Until	09:00 May-18
Aircraft	C-141
Qualifications	Basic
Handled by	Smith Jones Garcia

Figure 10: Transformation of the sample task's descendants by the 6th Airlift Squadron cluster

thousands of different items. As far as the resources available to perform the transport tasks, each of the wings had an average of 30 planes available at initialization time, the GlobalSea had 6 ships available, the installations had each about 600 trucks and railcars available, and the air squadron had about 200 crew members. The planes were read from a live military database, while the remaining data were real but not live, captured via snapshots of military databases. The transportation society succeeded in building a full transportation schedule down to a low level of detail, modulo those pieces that have not yet been included in the society (such as maintenance scheduling).

The full logistics society was composed of 76 clusters and hundreds of agents representing different organizations in the military logistics domain. The full society ran over a wide-area network at different locations in the country. The transportation society all ran at the same physical location but read data from databases at different locations and communicated with non-transportation agents at different locations. Our clusters, which are all those in the transportation society except the port operations cluster, shared two Windows NT Pentium 450 MHz machines, each with 256 Megabytes of RAM.

Since transportation requirements were being generated by clusters outside of the transportation society that used a variety of different algorithms, not all the transportation requirements arrived at the Global Mode cluster (the portal to the transportation society) at the same time. Instead there was a steady stream of requirements arriving a the Global Mode for about 20 minutes. The kinds of items that had to be scheduled varied from tanks to different kinds of ammunition, to pallets packed with loose equipment, to military personel. The transportation society converged on a deployment schedule in about half an hour after the arrival of the first transportation requirement.

This result of requiring only a half an hour to generate a schedule is the key one. It is true that the problem we solved is only a fraction of the size of the full-scale deployment problem that required a month for hundreds of people to solve. However, the inherent parallelism of the multiagent approach (dividing the problem into manageable subproblems) implies that, assuming that our approach continues to scale as well as it has to this point, we can achieve a multiple-orders-of-magnitude speedup for the full problem. Such a system would provide the means to rationally rebuild schedules as objectives change or things do not proceed according to plan, something that the current manual approach does not provide.

As to the quality of the schedules, we could not compare our results with manually generated schedules, because no humans had never solved the same problem as our system did, and for similar problems that humans have solved the results have not been saved. The best that we could do was to have human experts review our schedules and provide their opinions. All pieces of the schedule were considered realistic to first order. Closer inspection revealed, as expected, suboptimalities in certain portions of the schedule generated by the "simple" agents. However, the more sophisticated genetic-algorithm-based agents produced high-quality schedules. Based on these results, the Air Force decided to fund the development of an operational aircrew scheduler based on our squadron aircrew scheduling agent.

One important result was the demonstration of the relative ease with which standalone systems not originally designed to work together could be made into agents and made to cooperate. It has traditionally been very difficult to get different systems to work cooperatively towards a larger goal. When such systems have been integrated, the resulting aggregate system has usually been hard to understand and maintain, sometimes known by the pejorative term "stovepipe systems". We have shown that the right infrastructure provides the capability to build elegant and easily understood large-scale scheduling systems from smaller systems.

A final noteworthy result is that our multiagent society has remained stable, even as the external agents making transportation requests have rescinded and changed their original requests. We have accomplished this largely by making the agents (particularly the ones at the upper levels of the hierarchy) minimize the changes to the schedule, sometimes at the expense of optimality. Because we have not been operating the system near its capacity, the lowering of schedule quality due to this emphasis on stability is very small. However, as we start to operate the system closer to capacity, we expect this to become an important issue (as we discuss in the next section).

4 Future Work and Issues

While we have made good progress towards demonstrating full automation of the transportation scheduling for the U.S. military, there are some large problems remaining to be solved. We now discuss some of these remaining issues.

Improving Functionality of Individual Agents - There are some functions required to perform military transportation scheduling that we have ignored in the current society. Included in these are scheduling of vehicle maintenance, scheduling of maintenance crews, and usage of commercial carriers rather than military transportation. Concerning the latter, we are currently working on including a "virtual airlines" agent, which will be the first agent in our society that uses a commercial carrier. It will automatically schedule the excess capacity of a commerical U.S. airline for military transportation, primarily passengers.

Other than the squadron aircrew scheduler, all of the agents have only been tested for correctness at a high level and under non-stressful conditions. If we ever were to field such our transportation society, these agents would have to be improved to the point where they are shown to consistently provide performance comparable to their human counterparts under a wide range of conditions.

Replication of Similar but not Identical Agents - For a society handling the full problem and not just a slice of the problem (as our current society does), each agent at the lower levels of the hierarchy will be one of a large set of agents that perform essentially the same function for a different collection of assets. For example, there will be approximately 100 different aircrew schedulers corresponding to all the different air mobility squadrons. The problem is that these agents will not be exactly identical because the different organizations they represent each have a slightly different set of constraints (i.e., business rules).

For instance, the different forts and depots each have different agreements with the local governments on rules governing road usage, and the different air mobility squadrons each can set their own additional training requirements beyond the minimum specified for all squadrons.

We, the developers, cannot be responsible for configuring all the different instances of our aircrew scheduler or land move scheduler. The approach we are currently pursuing is to provide customizable business rules with a graphical interface for creating and editing the business rules. The different organizations will then be able to customize the agent that represents them in the society.

Managing Instabilities - Our decision so far to keep the society stable by minimizing schedule changes is one that will cost us an increasing price in terms of schedule quality as we feed the system more tasks and hence operate closer to capacity. When the system is near capacity, it will be unlikely to find some simple way of handling new tasks based on a large number of idle resources. The problem is that it is generally the case that nonlinear dynamic systems tend to have greater instabilities as they approach capacity (Bak, 1996). Therefore, we will need to develop techniques that allow us to keep the system instabilities under control without sacrificing excessively the schedule quality. [Note that the tendency to opt for stability at the expense of optimality and the difficulty of controlling the "chaos" that can result from continually pursuing optimality are problems not only for societies of electronic agents but also, according to some observers (Peters, 1987), for organizations of human agents.]

5 Conclusion

We have made significant progress towards automating the complete U.S. military transportation scheduling problem. We have done this using a multiagent system to decompose the full problem, which is far too large to solve all at once, into separate but interacting subproblems. More important than the fact that these subproblems can be distributed across multiple computers is the fact that these subproblems are of the size that can be solved by existing scheduling algorithms in relatively short times. A highly structured multiagent infrastructure has allowed us to integrate into a coordinated society agents of all types, including standalone systems "wrapped" to act like agents and agents built by other organizations. Our results indicate that it is possible to achieve a multiple-orders-of-magnitude speedup over the approach currently employed, which uses the largely manual labor of hundreds of people. We are continuing to work on scaling up our system and improving the individual agents to the point where this possibility becomes a reality. In the meantime, our society provides a model for how to construct very-large-scale scheduling systems using a multiagent approach.

Acknowledgements

We would like to thank Todd Carrico, Stu Draper, Ed Kera, Steve Milligan, Marshall Brinn, and Sean Moore for their suggestions and guidance. We would like to thank Oliver Oberdorf, Susan Fairchild, Lora Goldston, Nick Pioch, Beth DePass, Marshall Brinn and Andy Shultz for their work on the development of the transportation society. We would like to thank Steve Milligan, Jeff Berliner, Mike Thome, Rich Lazarus, Dan Cerys, and Nate Combs among others, for their work on design and development of the ALP architecture.

References

Bak, P. (1996). *How Nature Works: The Science of Self-Organized Criticality.* New York: Springer-Verlag.

Barbuceanu, M. & Fox, M. (1996). The Design of a Coordination Language for Multi-Agent Systems. In M. Woolridge, J. Muller, M. Tambe (Eds.), *Intelligent Agents II: Agent Theories, Architectures and Languages.* New York: Springer-Verlag.

Bierwirth, C., Kropfer, H., Mattfeld, D. & Rixen, I. (1995). Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment. *IEEE Conf. on Evolutionary Computation,* 439–443.

Burke, E., Newall, J. & Weare, R. (1998). Initialisation Strategies and Diversity in Evolutionary Timetabling. *Evolutionary Computation,* 6(1), 81–102.

Coello Coello C. (1999). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3), 269–308.

Corkill, D., Gallagher, K. & Murray, K. (1986). GBB: A Generic Blackboard Development System. *Proceedings of the Fifth National Conference on Artificial Intelligence,.*

Davis, L. (1991). *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold.

Davis, R. & Smith, R. (1983). Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence,* 20(1), 63–109.

Day, P. & Ryan, D. (1997). Flight Attendant Rostering for Short-Haul Airline Operations. *Operations Reserach,* 45(5), 649–661.

Deb, K. & Chakroborty, P. (1998). Time Scheduling of Transit Systems with Transfer Considerations Using Genetic Algorithms. *Evolutionary Computation,* 6(1), 1–24.

Falasconi, S., Lanzola, G. & Stefanelli, M. (1996). Using Ontologies in Multi-Agent Systems. *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop.*

Gabbert, P. et al. (1991). A System for Learning Routes and Schedules with Genetic Algorithms. In R. Belew & L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms,* 430–436. San Mateo, CA: Morgan Kaufmann.

Gasser, L., Braganza, C.& Herman, N. (1988). Implementing Distributed AI Systems Using MACE. *Readings in Distributed Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann.

Genesereth, M., Singh N. & Syed, M. (1994). A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation. *CIKM Workshop on Intelligent Information Agents.*

Genesereth, M. & Ketchpel, S. (1994). Software Agents. *Communications of the ACM,* 37(7).

Georgeff, M. et al. (1998). The Belief-Desire-Intention Model of Agency. *Intelligent Agents V: Proceedings of the Fifth International Workshop on Agent Theories, Architectures and Languages,* 1–10.

Goldberg, D. (1988). *Genetic Algorithms in Search, Optimization and Machine Learning.* Redwood City, CA: Addison-Wesley.

Gonzalez, C., & Wainwright, R. (1994). Dynamic Scheduling of Computer Tasks Using Genetic Algorithms. *Proceedings of the First International Conference on Evolutionary Computation*, 829–833.

Grefenstette, J. (1987). Incorporating Problem Specific Knowledge in Genetic Algorithms. In L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing,* 42–60. Los Altos, CA: Morgan Kaufmann.

Gruber, T. (1992). Ontolingua: A Mechanism to Support Portable Ontologies. *Stanford University, Knowledge Systems Laboratory, Technical Report KSL-91-66.*

Halhal, D., Walters, G. A., Ouazar, D. & Savic, D. A. (1997). Water Network Rehabilitation with Structure Messy Genetic Algorithm. *Journal of Water Resources Planning and Management,* 123(3), pp. 137–146.

Hart,E., Ross, P. & Nelson, J. (1998). Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder. *Evolutionary Computation,* 6(1), 61–80.

Hogg, T. & Huberman, B. (1991). Controlling Chaos in Distributed Systems. *IEEE Transactions on Systems, Man, and Cybernetics,* 21(6).

Levine, D. (1996). Application of a hybrid genetic algorithm to airline crew scheduling. *Computers & Operations Research,* 23(6), 547–558.

Matthews, J. & Holt, C. (1996). *So Many, So Much, So Far, So Fast.* Washington, D.C.: United States Transportation Command and Office of the Chairman of the Joint Chiefs of Staff.

Montana, D. & Czerwinski, S. (1996). Evolving Control Laws for a Network of Traffic Signals. *Genetic Programming 1996: Proceedings of the First Annual Conference*, 333–338. Cambridge, MA: MIT Press.

Montana, D. (1998). Introduction to the Special Issue: Evolutionary Algorithms for Scheduling. *Evolutionary Computation,* 6(1), v–ix.

Montana, D., Brinn, M., Moore, S., & Bidwell, G. (1998). Genetic Algorithms for Complex, Real-Time Scheduling. *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics,* 2213–2218.

Montana, D., Bidwell, G., Vidaver, G. & Herrero, J. (1999). Scheduling and Route Selection for Military Land Moves Using Genetic Algorithms. *1999 Congress on Evolutionary Computation,* 1118–1123.

Nordstrom, A.-L. & Tufekci, S. (1994). A Genetic Algorithm for the Talent Scheduling Problem. *Computers & Operations Research,* 21(8), pp. 927–940.

Peters, T. (1987). *Thriving on Chaos.* New York: Harper Collins.

Reynolds, C. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics,* 21(4), 25–34.

Sen, S. Roychowdhury, S. & and Arora, N. (1996). Effects of local information on group behavior. *Proceedings of the Second International Conference on Multiagent Systems,* 315–321. Menlo Park, CA: AAAI Press.

Sen S. & Durfee, E. (1998). A Formal Study of Distributed Meeting Scheduling. *Group Decision and Negotiation,* 7, 265–289.

Stojkovic, M. & Soumis, F. (1998). The Operational Airline Crew Scheduling Problem. *Transportation Science,* 32(3), 232–245.

Syswerda, G. (1991). Schedule Optimization Using Genetic Algorithms. In L. Davis (Ed.), *Handbook of Genetic Algorithms,* 332–349. New York: Van Nostrand Reinhold.

Thangiah, S. (1995). An Adaptive Clustering Method using a Geometric Shape for Vehicle Routing Problems with Time Windows. In L. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms,* 536–543. San Francisco: Morgan Kaufmann.

Weiss, G. (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* Cambridge, MA: MIT Press.

Whitley, D., Starkweather, T. & Fuquay, D. (1989). Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In J. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms,* 133–140. San Mateo, CA: Morgan Kaufmann.