ABSTRACT

MELENDEZ, BARBRA SUE. On Scheduling Delivery in a Military Deployment Scenario. (Under the direction of Thom J. Hodgson and Kristin Thoney.)

The ability to rapidly and accurately perform sensitivity analysis in military deployment planning is a vital tool for force deployment planners. The Deployment Scheduling Analysis Tool (DSAT), a new software tool, provides this ability. DSAT builds the deployment scenario through a graphic user interface, invokes an adaptation of the Virtual Factory to schedule the movement and delivery of the equipment and provides meaningful output in the form of reports and graphics. The Virtual Factory is a job shop scheduling procedure developed at North Carolina State University which is proven to rapidly provide near-optimal solutions to large problems.

This research focuses on evaluating both the accuracy and effectiveness of DSAT. An existing tool, the Deployment Analysis Network Tool Enhanced (DANTE), is proven to minimize the time required to deliver the equipment (C_{max}). Since DANTE is a relaxation of the original problem, it establishes a lower bound for C_{max}. An extension of DANTE, COMFLOW, includes due date information and establishes a lower bound on the maximum lateness of the equipment, L_{max}. DSAT's schedule, in terms of C_{max} and L_{max} are compared to their lower bounds. Finally, DSAT's schedule, in terms of transportation asset utilization, is compared to accepted asset utilization planning factors. This evaluation indicates that DSAT provides near optimal schedules for air deployments and good schedules for deployments including rail and sea movement.

On Scheduling Delivery in a Military Deployment Scenario

by

BARBRA S. MELENDEZ

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Graduate Program in
Operations Research

Raleigh

2002

Approved By:

Thom J. Hodgson Co-Chair of Committee	Kristin A. Thoney Co-Chair of Committee
Yahya Fathi	Russell E. King

Dedication

This work is dedicated to my mother, Judith Suzanne Myers, who always encouraged me to be all I could be.

Biography

Lieutenant Colonel Barbra Melendez was born on 24 October 1959 in Peoria, Illinois to Fred Henneike and Judy Myers. Her family moved several times until settling in Decatur, Georgia. Barbra enlisted in the United States Women's Army Corps on September 5, 1978.

Barbra spent approximately two years as an enlisted soldier prior to being accepted as a cadet in the United States Military Academy. She graduated in 1984 with a Bachelor of Science degree in Applied Mathematics. Barbra spent the next eighteen years in various locations and positions of greater responsibility in service to the United States.

In addition to the Bachelor of Science degree from West Point, Barbra holds a Master of Science in Operations Research from the Georgia Institute of Technology. She is married to Chief Warrant Officer Three (Retired) Rangel Melendez. They have two children, Robert Sean age 12 and Mark Christopher age 9.

Acknowledgements

I owe a great debt of gratitude to many people for their support of my work and efforts. First and foremost, I must thank my husband. Without his full support, I would never have tried to steer my career down this particular path.

I must thank each of my advisors for their support and efforts. Dr. Hodgson provided valuable insights into the complex nature of the subject matter. His tireless support and honest enthusiasm for this subject continued to motivate me throughout my work. Dr. Thoney provided essential support in several areas. She provided a much needed critique of my written work which improved the quality of my dissertation. She also continued to work on her specialized application of the Virtual Factory adding new capabilities and helping me to implement needed code to perform my analysis for this research. Drs. King and Fathi not only provided me valuable feedback concerning my writing, but also provided expertise and knowledge in numerous subject areas. Their productive and informative classroom lectures continued to hold my interest and pique my curiosity throughout my stay at NC State.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AB	Air Base
ADANS	Air Mobility Command Deployment Analysis System
AMC	Air Mobility Command, a subordinate organization of USTRANSCOM
ASB	Army Science Board
Closure Time	Amount of time needed to deliver last item in the deployment package.
C_{max}	Maximum Completion Time
CONUS	Continental United States
CRAF	Civil Reserve Aircraft Fleet
DAMMS	Department of Army Movements Management System
DANTE	Deployment Analysis Network Tool Enhanced
Deployment Horizon	Length of time required to deliver the complete deployment package to the final destination
Deployment Network	Collection of origin sites, final destination site, ports, and transportation assets operating between ports which support a specific deployment operation
Deployment Package	List of unit equipment and personnel selected for deployment
DSAT	Deployment Sensitivity Analysis Tool
ELIST	Enhanced Logistics Information Support Tool
Hazmat	Hazardous material

ISB	Intermediate Staging Base
JFAST	Joint Flow and Analysis System for Transportation
LB	Lower bound
L_{max}	Maximum Lateness
MASS	Mobilization Movement Control Automated Support System
MOG	Maximum on ground capacity
Movement Requirement	List of equipment, with defining characteristics, due dates and destinations, designated for deployment
MSC	Military Sealift Command, a subordinate organization of USTRANSCOM
MTMCTEA	Transportation Engineering Agency of the Military Traffic Management Command, a subordinate organization of USTRANSCOM
NP	Non-polynomial
PORTSIM	Port Simulation System
PREPO	Pre-positioned equipment
QCOA	Quick Course-of-Action Evaluation Tool Kit
SQL	Structured Query Language
STON	Short ton (equivalent to 2000 pounds)
TARGET	Transportation Analysis Reports Generator
Theater	The area to where military forces deploy to execute their missions. For example, in Operation Desert Storm, the Middle East was the theater of operations.
TPFDD	Time Phased Force Deployment Data
USTRANSCOM	United States Transportation Command

VB	Visual Basic

Chapter 1

Introduction

1.1 Overview

The military must often move a large number of personnel and equipment from one location (or many) to another, always as rapidly as possible. In other words, the military must effectively plan, schedule and execute force deployment operations. These force deployments are characterized by the following main components:

- The military units, consisting of personnel and equipment, that are deploying, their current location, availability dates and their final destination;
- The selection of transportation ports (air, rail and sea) through which the units will travel;
- The selection of available transportation assets (planes, trains, and ships) on which the units will move.

Each component is a dynamic element in the overall force deployment problem. Unit availability dates may change as a result of unforeseen maintenance setbacks; maintenance issues as well as many others also affect port capabilities and available transportation assets. Each component also impacts the other components. The final destination affects port selection; choices are reduced to ports in or near the target area.

Port selection influences the selection of transportation assets; all ports can not support all assets. Airports with short runways or dirt runways can not support the larger aircraft. Seaports with shallow depth cannot support the larger ships with deeper draft.

Effectively scheduling the deployment of forces is an extremely complex problem. Given that the above components are dynamic, the force deployment scheduling problem must be repeatedly solved based on the current situation. Furthermore, many military deployments are short-notice contingency and peacekeeping operations (Humanitarian Relief, etc) which require rapid planning and scheduling. Force deployment planners simply do not have the luxury of unlimited time to determine the best means of deploying a given force to meet the demands of an operational plan.

In addition to providing rapid analysis of ongoing force deployment operations, the force deployment planners must also provide accurate and time-sensitive analysis in support of military studies on future force design. The Army Chief of Staff, General Shinseki, plans to transform the Army into a lighter force capable of deploying one division any where in the world in five days, and five divisions in 30 days. The Airforce maintains that with its current structure and cargo aircraft, these goals are impossible to meet. Their inventory of aircraft must change to support the changing structure of the military forces. While the actual product development timeline might be quite long (up to at least 10 years), money must be committed now to see the results 10 years from now. Significant amounts of money are committed in the early stages of product development. Force Deployment Sensitivity analysis is a key tool in the design of future equipment and cargo carriers.

Current Military Deployment Planning models are not adequate tools to provide timely and accurate sensitivity analysis for force deployment planners. The current models are cumbersome and difficult to work with, requiring detailed time-phased force deployment data (TPFDD) to run. TPFDD is very time consuming to generate; it involves both operational and logistical planners working in an interactive process. Both types of planners require extensive training to effectively generate the required data. Force deployment planners need a fast, accurate and deployable tool to effectively plan and provide analysis for a military deployment problem.

1.2 The Military Deployment Problem

The military performs many different missions. These missions range from humanitarian relief through peacekeeping to armed conflict. Different missions require different types of units. Medical, engineering and water purification units are needed for humanitarian relief efforts. Military police and infantry are used in peacekeeping missions. Finally, depending on the size and scale of the armed conflict, either a partial or total mobilization of forces is required.

Each military unit has a unique collection of equipment that enables them to conduct their unit mission in any type of environment; each piece of unit equipment has specific characteristics such as weight, length, width and height. The collection of units selected for a specific deployment has become known as the “deployment package”. The units within the deployment package are each given a specific required delivery date (due date) and final destination. Defining these additional aspects of the unit equipment transforms the deployment package into a movement requirement. The movement requirement determines several aspects of the military deployment problem. This requirement identifies the number of pieces of equipment to be moved, the type of equipment, the origin (current location) and destination of the equipment as well as when it must be delivered.

Once the movement requirement has been determined, the deployment planner must now focus on the port facilities that will be utilized during the deployment process. First, the need for intermediate ports (transshipment nodes) must be established; a plane can only fly so far before it must land. In addition, if a particular route employs different modes of transportation, i.e. rail and shipping, the modal change occurs at a transshipment node. The origin ports, final destination ports, and any selected transshipment sites become the nodes in the deployment network with an associated (geographical) distance between them. Each port has its own characteristics associated with it. Sea ports are characterized by the number of berths, the material handling equipment available to move, lift and load containers onto the ship, the stevedore crews available, the depth of the harbor area, etc. Similar details, as well as the number of runways available, the runway length, surface material and the MOG characterize airports. MOG refers to the Maximum-on-Ground capacity of an airport - the maximum number of aircraft allowed in the operational area of the port at any one time. The number of switching tracks available, the number of loading ramps, and the size of the rail system characterize rail ports (depots). All of these characteristics impact the port cargo throughput capacity, the amount of cargo that can be processed through the port complex and loaded on to an awaiting transportation asset in a given time period.

Once all the ports in the network are finalized, the arcs must be “capacitated” by assigning the transportation assets to them. All transportation assets (ships, planes and trains) are limited resources that must be judiciously allocated to each ongoing mission. Each transportation asset is also characterized by several factors: weight capacity, volume capacity, area capacity, door size, speed of travel, amount of time required to load (equipment onto said asset), unload, and perform maintenance, etc. The allocation of transportation assets to each arc in the network will have a significant impact on the total amount of time required for the last unit to arrive, unload and clear the final destination port. This total amount of time is called the “deployment horizon” or deployment closure time.

The deployment planner allocates the transportation assets to the routes or the network arcs with the goal of minimizing the deployment closure time. The asset – route allocation usually changes over time. As the majority of the movement requirement moves towards the final destination, so do the transportation assets. This type of allocation, a dynamic asset – route allocation, establishes a dynamic deployment network. An asset – route allocation that remains fixed over time, a static allocation, establishes a static deployment network. The asset – route allocation, dynamic or static, completes the definition of the deployment network: the origin nodes, transshipment nodes, and the final destination nodes, and the capacities of each arc linking the nodes. Once the deployment network has been established, the next phase in the deployment problem is to determine the schedule of equipment onto the transportation assets that will minimize the maximum lateness of any piece of equipment.

Thus, the deployment problem is really a series of two interrelated combinatorial problems. First, the deployment planner must determine the optimal allocation of available transportation assets to each leg of a deployment network (defined as the set of origin, intermediate, and final destination nodes) which will minimize the deployment closure time. Next, given the established deployment network, the planner must determine the optimal assignment of equipment to routes that will minimize the maximum lateness of any piece of equipment.

1.3 Research Objectives

There are 4 distinct objectives of this research. The first objective is met in the last major section of Chapter 3. In this chapter, a key deployment analysis tool will be discussed, the Deployment Analysis Network Tool Enhanced (DANTE). DANTE models the military deployment problem as a multiperiod minimum cost network flow problem. DANTE aggregates all movement requirements into a total weight requirement (disregarding equipment volume and area specifications and transport asset volume and area limitations), it also aggregates port facilities and transportation assets to represent combined capabilities according to a fixed network structure. DANTE establishes a lower bound for the deployment horizon, $LB(C_{\max})$, for a fixed transportation network and specified movement requirement. This chapter's last section presents a proof that DANTE's minimum cost objective function does minimize the deployment closure time.

This proof establishes DANTE as an effective tool for determining the lower bound for the deployment closure time, LB(C_{max}), or the deployment completion time. DANTE forms the basis for a new lower bound tool developed later in this research.

The second objective is met in the last section of Chapter 4. In this chapter, the DANTE network flow model is evolved into a multicommodity, multiperiod network flow model. The total movement requirement is broken down into multiple commodity packages. This simple modification provides the deployment planner with the capability to track the movement of individual commodities through the deployment network; it also allows the planner to determine when the individual commodities arrive at the final destination. This chapter's last section details an iterative method to establish the lower bound for the maximum lateness, LB(L_{max}), of the deployment package.

The third objective is met in the last major section of Chapter 5. In this chapter, a new deployment planning tool, the Deployment Scheduling Analysis Tool (DSAT), is introduced. This tool is designed to create deployment schedules that meet delivery date requirements of a specified deployment package. Key heuristics and procedures of this tool are presented in detail. Modifications are made to DSAT's scheduling routine to improve performance. The last section describes an experiment conducted to determine the DSAT scheduling routine that yields the best results. The tool's improvements are defined in terms of the reduction of deployment closure time and maximum lateness and the increase in computer run time required to achieve these results.

The fourth objective is met in Chapter 6. This chapter details an evaluation of the effectiveness of DSAT's generated deployment schedules. The DSAT deployment schedules are first analyzed to determine how effectively the transportation assets are utilized throughout the deployment. Effective asset utilization is based on how well an asset's multiple capacities (weight, volume or area) are filled during each mission the asset executes. DSAT’s asset utilization is compared to accepted asset utilization planning factors. Next, DSAT’s schedule, in terms of deployment closure time (C_{max}) and maximum lateness of the deployment package (L_{max}) is compared to the lower bounds for both C_{max} and L_{max}.

Chapter 2

Literature Review

2.1 Military Deployment Tools

There are several deployment tools currently in use today by deployment planners of all branches (Air Force, Army, Navy, etc). Most of these tools are service specific in orientation and application. Some model all aspects of the deployment process.

2.1.1 Service Specific Tools

The various branches of the Armed Services do have good models and tools for planning and analyzing specific aspects of the overall deployment problem. These tools are all powerful aids but are limited to a small portion of the overall deployment problem.

The Air Force schedules both peacetime and wartime airlift operations. Their various tools include the Air Mobility Command Deployment Analysis System (ADANS) [25] which enables the Air Force to perform airlift scheduling operations on various scopes from large scale planning to detailed monitoring and execution of a current mission. Rappoport, et al [28], developed a new approach to assigning and scheduling aircraft. While most analyses addressed aircraft assignment as a plane minimization problem because of the limitations on available aircraft, they addressed the problem as an aircraft load maximization problem. They developed a heuristic for prioritizing the requirements for loading; their goal seeks to utilize each aircraft more fully. Solanki, et al [32], developed an execution planning algorithm that could be used daily to modify the existing airlift operations plan. An algorithm based on the insertion heuristic was selected for implementation due to its computational feasibility and its capability to absorb complex constraints arising in the execution planning problem. The insertion heuristic has the additional desirable feature of keeping intact as much of the existing schedule as possible. Rathi, et al [29], presents mathematical models for analysis of the airlift deployment problem, which involves the allocation of a limited number of aircraft towards the shipment of cargo and personnel between many origins and destinations within prescribed time windows. These formulations attempt to optimally allocate strategic airlift resources towards the shipment demand (cargo and personnel) such that the maximum amount of demand is delivered on time using preferred aircraft types. A much smaller statement of the problem is achieved in a third formulation by assuming that lateness is distributed evenly over all routes. All formulations were modeled by linear programming formulations and offered a tradeoff between the degree of control to be exercised by the planner and the speed of computation.

The military also has several decision support tools at its disposal. The Port Simulation (PORTSIM) [24] analyzes input data for real-world ports and determines the port's throughput capability and any limiting constraints (space available, material handling equipment, etc). The Transportation Analysis Reports Generator (TARGET) [23] provides detailed equipment listings for Army units and estimates the transportation asset requirements to move various groupings of units. The Department of Army Movements Management System (DAMMS) [26] provides the military with worldwide automated cargo tracking capability. The Mobilization Movement Control Automated Support System (MASS) [26] provides planning and control systems for the highway convoy of military units to their deployment ports of embarkation.

2.1.2 Multi-Mode Tools

There are currently very few software packages in use to assist in planning a military deployment utilizing all available resources. One of these, the Joint Flow and Analysis System for Transportation (JFAST) [24] was developed by the Oak Ridge National Laboratories to support the United States Transportation Command (USTRANSCOM). JFAST is a simulation program that provides detailed estimates for strategic force movement by air and sealift assets. Deployment planners use JFAST to estimate unit closure dates (time for entire unit to reach final destination). Unfortunately, JFAST requires a very rigid input structure, which requires a great deal of time to prepare. This input requirement makes JFAST both a very difficult and an extremely time consuming program to operate. Because of these operational characteristics, JFAST is not a good tool for sensitivity analysis.

The Deployment Analysis Network Tool Enhanced (DANTE) [17] provides a sensitivity analysis tool for studying large-scale deployment scenarios. DANTE models the “flow” of troops and equipment from Air- and Sea-Ports of Embarkation (APOE, SPOE), through air and sea movement corridors to Air- and Sea-Ports of Debarkation

(APOD, SPOD), then through forward transportation corridors (air, road, rail or inland waterways) to the final assembly area. The process is modeled as a time-phased network flow problem with the objective of minimizing the deployment closure time. The model is optimized using an Out-of-Kilter algorithm. The Transportation Engineering Agency of the Military Traffic Management Command, MTMC_TEA, validated DANTE against the Joint Flow and Analysis System for Transportation (JFAST). A number of varying scenarios were modeled using both tools. The results from both analyses were consistent. However, DANTE provides analysis for an aggregate tonnage requirement; it does not deal with equipment/asset compatibility issues. Individual port capacities are also aggregated into a general geographical representative port. DANTE reduces all deployment scenarios to a general network structure utilizing notional representative port facilities and assets. DANTE lacks the detail necessary to create a unit deployment schedule.

In addition, DANTE does not take into account the possibility of reallocating transportation assets to routes over time. To alleviate this shortfall, Trainor [34] developed the Dynamic Transportation Asset Allocation Model. This model incorporates asset constraints into a linear programming formulation of a deployment scenario. These additional constraints involve integer variables. Since the resulting integer program is extremely difficult to quickly solve, the linear programming relaxation is solved. A post-processing algorithm is employed to obtain an integer solution. This dynamic allocation of assets to routes builds a dynamic deployment network designed to minimize the deployment closure time.

Currently there exists a shortfall of programs that allow users to rapidly perform accurate deployment planning and sensitivity analysis and retain enough problem definition to develop an effective unit deployment schedule.

2.2 Network Flow Problems

A large body of previous work exists in this area. This major area has been divided into the smaller areas of multiperiod network flow problems, service network design problems, and multicommodity network flow problems. Each specific area has it own unique characteristics and solution methods.

2.2.1 Multiperiod Network Flow Problems

Aronson, et al [1] developed a forward network simplex algorithm for solving multiperiod network flow problems. The method exploits the natural decomposition of multiperiod network problems by limiting its pivoting activity. A forward algorithm is an approach to solving dynamic problems by solving successively longer finite subproblems, terminating when a stopping rule can be invoked or a decision horizon found.

Aronson, J.E., [2] developed a specialized branch-and-bound algorithm for the multiperiod assignment problem. The multiperiod assignment problem, a specialization of the three dimensional assignment problem, is an optimization model that describes the situation of assigning people to activities or jobs over time. He considered the most general case, which has both a cost of assigning a person to an activity in each time period, and a cost of transferring the person from one activity in each period to another activity in the next period. He formulated the multiperiod assignment problem as a multicommodity network flow problem. The special structure of this problem enabled him to (relatively) quickly find a good feasible solution using the shortest path heuristic algorithm. A new branch and bound algorithm for solving this problem was developed. The computational tests of the algorithm were performed on moderately sized problems involving up to seven personnel and activities, and up to ten time periods.

2.2.2 Service Network Design Problems

Kim, et al [20] developed and solved a large-scale service network design problem involving package delivery under various constraints (time windows, limited ground and air transport fleet, etc). They strengthened the linear programming relaxation by adding valid inequalities. By exploiting the problem structure using a specialized network representation and applying a series of problem reduction methods, they achieved dramatic decreases in problem size without compromising the optimality of the model. One of their reduction techniques involved creating "super commodities". A commodity was defined by origin and destination information for each package. All packages with the same origin (or destination) were aggregated into a super commodity. They concluded that although state-of-the-art integer programming methods can work well for relatively small, uncongested service network design problems, they must be used in concert with heuristics to be effective for large-scale, congested problems encountered in practice. One data set representing a complete operation of a real world express package delivery company resulted in a problem with 151,002 constraints, 1 billion package flow variables and 0.3 billion asset and route variables. After applying their network reductions, the number of forcing constraints was reduced to 15,355.

Fisher, et al [14] considered the problem in which a fleet of vehicles must be scheduled to pickup and deliver a set of orders in truckload quantities. They developed an algorithm based on a network flow relaxation, which imposes necessary conditions on the flow of empty vehicles. The network flow model provides a lower bound and a nearly feasible solution that can be made feasible with some simple heuristics. On both real world scenarios and random problems, the algorithm consistently produces solutions within 1% of optimality after seconds of computer runtime. The algorithm constructs the routes for the vehicles. The problems based on real data varied from 74 to 87 orders (number of truckloads to be delivered) using from 33 to 53 (100 available) vehicles. The random problems varied from 20 to 100 orders and 15 to 73 vehicles.

Crainic, et al [12] examined the freight transportation problem which occurs when the same authority controls and plans both the supply of transportation services (modes, routes, frequencies of the services, and governing policies for terminals) and the routing of freight. They developed a general modeling framework, based on a network optimization model, which could be used to enhance the tactical and strategic planning process for such a system. The purpose of the model was not to obtain a detailed representation of operations, but to generate "best" operating strategies to reduce costs and delays and to improve the quality of service. The problem is solved by means of an algorithm based on decomposition and column generation principles.

2.2.3 Multicommodity Network Flow Problems

Crainic, et al [11] developed a tabu search procedure for multicommodity location/allocation problems. This problem typically arises in the shipping industry where container depots have to be selected, the assignment of customers to depots has to be established for each type of container, and the interdepot container traffic has to be planned to account for differences in supplies and demands in various zones of the geographical territory serviced by the shipping company. It is modeled as a mixed integer program, combining zero-one location variables and a multicommodity network flow structure. They use a relaxed formulation of the original problem by fixing the location variables, yielding an uncapacitated multicommodity network flow problem. They then restricted their search procedure to the space of the location variables.

Iakovou, et al [19] consider the strategic level routing problem of hazardous materials in marine waters over a multicommodity network with multiple origins-destinations using a network flow model. A commodity is defined as an origin-destination pair of nodes. They worked on selecting the best route for each hazardous material and commodity couple. Their objective was to generate the best global strategies to balance the tradeoffs between the transport costs and the expected total risk costs, while enforcing an equitable distribution of risk. They developed an efficient two-phase solution approach. The first phase obtains good lower bound and upper bounds for the problem. Phase two concentrates on closing the gap between the two bounds.

Lamar, et al [21] worked with uncapacitated multicommodity flows on a fixed charge network. The uncapacitated, multicommodity network design problem is modeled with aggregate and disaggregate forcing constraints. (Forcing constraints ensure logical relationships between the fixed charge-related and the flow-related decision variables.) Origin and destination nodes define their commodities. A new lower bound for this problem – referred to as the capacity improvement (CI) bound – is presented. A key feature of the CI lower bound is that it is based on the LP relaxation of the aggregate version of the problem. They also demonstrated through a numerical example that the CI lower bound can converge to the optimal objective function value of the IP formulation.

Assad, A.A, [3] wrote a comprehensive survey of the literature dealing with the multicommodity flow problem. He presented various models for the multicommodity problems, including capacitated and uncapacitated linear multicommodity problems and nonlinear multicommodity problems. He also presented solution techniques (and source references) for both linear and nonlinear flow problems. The former included decomposition, partitioning, compact inverse methods, and primal-dual algorithms. A variety of feasible directions are presented for the latter.

Crainic, et al [13] presented a branch-and-bound algorithm for the multicommodity location-allocation problem with balancing requirements. The formulation displays a network flow structure that is most favorable to efficient algorithmic developments due to the presence of the balancing requirements. In particular, tight bounds may be efficiently computed by using a reformulation of the weak relaxation of the problem as a minimum cost multicommodity flow problem. The relaxations are actually uncapacitated multicommodity flow problems, which may be separated into single commodity flow problems. They also developed and analyzed various branching criteria and showed that the branching rules, which form the most efficient branch-and-bound procedure for the present problem, are quite different from those used to solve the classical location problems. The experimentation was conducted on both randomly generated problems and on a large-scale application.

2.3 Scheduling

This subject area has a wealth of written material. It is divided into four sub sections: single machine scheduling, parallel machine scheduling, job shop scheduling and Virtual Factory scheduling.

2.3.1 Single Machine Scheduling

Baker, et al. [5] dealt with sequencing jobs on a single machine with no preemption to minimize maximum tardiness. The static problem assumes all jobs are available at the same time while the dynamic problem assumes jobs become available over time. They present the optimal solution for the static problem; Jackson's Theorem states that processing jobs in nondecreasing order of their due dates yields an optimal sequence of jobs. They also state that a modification of Jackson's theorem can be applied to the dynamic problem if preemption is allowed. The adaptation is as follows: 1. At each job completion, the job with minimum due date among the available jobs is scheduled. 2. At each ready time, the due date of the now available job is compared to the due date of the job currently under process. The current job is preempted if the new due date is earlier than the current job's due date. They note that the solution in this dynamic preemptive situation is easy to construct because the mechanism is a dispatching procedure, which means that scheduling decisions are made at chronologically ordered points in time. They developed a branch and bound technique for the dynamic non-preemptive situation. They used four different dispatching procedures to construct an initial solution using the adapted Jackson's Theorem. They presented a short comparison of the different dispatching procedures on the absolute deviation of the initial solution from the optimal solution. Although the paper addresses a specific scheduling criterion, their general purpose is to illustrate how basic properties of the static problem can effectively be utilized in solving the dynamic problem. Their computational results show that problems with up to 30 jobs can be solved quite rapidly, most of the time. They proposed relying on their heuristic methods for larger problems where the branch and bound algorithm does not converge to an optimal solution quickly enough.

Pinedo [27] authored a book, which summarized scheduling theories and algorithms for diverse situations including deterministic and stochastic models, and single and multiple machine models. His section concerning the minimization of maximum lateness on a single machine, deterministic model showcases several typical scenarios. Two scenarios of special concern are well known single machine, no job precedence scheduling problems. The first problem assumes that all jobs are available at the same time. The second assumes that each job has a defined release date. He further states that the optimal job sequence for the first problem is the sequence of jobs scheduled according to the Earliest Due Date (EDD) rule. For the second problem, he presents a proof that it is strongly NP-hard. This latter problem has received a lot of attention because it appears frequently as a subproblem in flow shop and job shop problems. This attention has resulted in numerous "relatively effective" branch-and-bound techniques for determining an optimal sequence of jobs.

Bratley, et al [7], dealt with sequencing with earliest start and due dates with application to computing bounds for the general job shop problem to minimize the maximum makespan. They worked with a single machine scheduling problem that results from the relaxation of some constraints as follows: Suppose that a partial schedule already exists where the earliest start time of the remaining operations are affected by the earliest finish time of the scheduled jobs. A strong lower bound may be obtained by solving the one machine problem for each machine that must process the remaining jobs. The one machine problems consists of sequencing the remaining jobs, each with release time, $r_i \geq 0$, processing time, p_i, on that machine, and additional processing time, q_i, on the other machines. The q_i times do not interfere with each other. The sequence of jobs must minimize the completion time for all jobs, including the q_i's. They developed an algorithm that obtains optimal and near optimal solutions for a relaxed problem and used this method to obtain lower bounds for the general problem. This technique requires knowledge of job routing prior to scheduling.

McMahon et al. [22] dealt with scheduling on a single machine with earliest start times and due date constraints to minimize maximum lateness. They developed an algorithm for sequencing the jobs. Their algorithm could be classified as branch and bound, however, it provides a complete solution with each node of the enumeration tree. They start with a good initial solution based on a constructive heuristic proposed by

Schrage, and referenced in Bratley, et al [7]. Schrage's heuristic is basically to schedule all available jobs in earliest due date (EDD) order at time index *t*, update *t* to the end of the processing time, and then repeat until all jobs have been scheduled. This results in a non-preemptive schedule of jobs.

2.3.2 Parallel Machine Scheduling

Bratley, et al. [6] discussed the scheduling of jobs with release times and due dates on identical machines. They minimized the total time to complete all jobs without preemption. They presented a multi-stage solution algorithm that is based on an implicit enumeration procedure and also uses a labeling type algorithm, which solves the problem when preemption is allowed. Their problem sizes remain "small", both in the number of jobs and machines. They stated that when the number of jobs and machines is large, the size of the enumeration tree is very large. The solution may not be obtained in a reasonable amount of time on a contemporary computer. They worked with a maximum number of 25 jobs.

Horn [18] considered situations in which jobs require only one operation on a single machine, or on one of a set of identical machines. Penalty-free interruption is allowed. Some simple algorithms are given for finding optimum schedules to minimize maximum lateness and total delay, for the single-machine case, and maximum lateness for a restricted multi-machine case. The general multi-machine case has unique availability and due dates for each job. A simple flow problem formulation permits minimizing maximum lateness for the more general multi-machine case. The solution to the flow problem establishes a feasible schedule for the jobs. This flow problem was repeatedly solved in an iterative process developed to establish the maximum lateness in an optimal schedule.

Baker, et al [4], considered the problem of scheduling *n* jobs to minimize the total earliness and tardiness penalty. They reviewed the literature on this topic, providing a framework to show how results have been generalized. They started with a basic model containing symmetric penalties, one machine and a common due date. Then they progressed to more complicated situations by adding parallel machines, complex penalty functions and distinct due dates. In their conclusion, they mentioned that the problems could be described by two classes. One class involves a common due date for all jobs. The other class permits distinct due dates. For the second class, their research indicates that in general, it appears that only branch and bound techniques have been effective at solving these problems.

2.3.3 Job Shop Scheduling

Cho, et al [9] studied the problem of obtaining feasible preemptive schedules for independent jobs with associated release times and due dates in open, flow and job shops. They showed that for the case of the flow shop (and hence the job shop) determining whether or not all jobs can be completed by their due dates is NP-Hard.

Schutten [30] defined the term “practical job shops” as job shops with practical features such as transportation times, simultaneous resource requirements, setup times, and many minor but important other characteristics. He further stated that the principle of decomposing a classical job shop problem into a series of single-machine problems could be easily applied to such practical job shops. He presented several extensions of the Shifting Bottleneck procedure to accommodate issues such as due dates and release times of jobs. The Shifting Bottleneck procedure presupposes known routings prior to scheduling the jobs.

Brucker, et al [8], developed a new lower bound for the job-shop scheduling problem. The lower bound is based on a two-job relaxation, which can be solved efficiently by using geometric methods. Results show that the new lower bound improves the classical lower bound if the ratio between the number of machines and number of jobs is large.

Conway [10] investigated priority dispatching rules used in job shop scenarios. According to Conway, a priority rule can be considered to operate by assigning, at the time a selection must be made from queue, a numerical value called a “priority” to each of the waiting jobs and then selecting the job with the smallest priority value. The study was concerned with the desire to complete the processing of all jobs before their due dates. The investigation considered both the various methods that might be used to assign due dates and the various priority rules that might be used to meet the dates assigned. The shop scenario considered consisted of nine machine groups, each with a single machine. All machines were continuously available. The job routings were known in advance. Based on their results, they concluded that if the due dates are assigned by some external agency (outside of the job shop environment), then none of the standard and obvious priority rules’ performance is particularly noteworthy. Overall, he concluded that the shortest processing time priority rule exhibited the best performance of all rules tested.

2.3.4 Virtual Factory Scheduling

Hodgson, et al [15] demonstrated that a conceptually simple simulation-based procedure (first proposed by Lawrence and Morton, 1986) is both effective and efficient in providing optimal or near optimal schedules for minimizing the maximum lateness (L_{max}) in large job shops. This simulation became known as the Virtual Factory. They noted that in general, the Virtual Factory appeared to perform better on larger problems. This paper presented a lower bound for L_{max}, $LB(L_{max})$, by decomposing the *m* machine job shop into *m* single machine problems. The single machine lower bound for machine *m*, $LB_m(L_{max})$, based on work done by Carlier and Pinson, was computed for each of the *m* machines. The overall lower bound for the job shop problem is the maximum lower bound for all *m* single machines: $LB(L_{\max}) = \max_{m}\{LB_m(L_{\max})\}$. One major assumption in the Virtual Factory procedure is that all routes for jobs are known in advance.

Hodgson, et al [16] improved the simulation procedure by identifying the critical jobs (those jobs that were among the latest) and inserting "idle time" into a machine schedule just prior to the arrival of a critical job to that machine. This ensures that no critical jobs are delayed due to noncritical jobs currently under process.

Weintraub, et al [35] developed a tabu search procedure for the Virtual Factory that identified process plans with alternative operations and routings for jobs. This enhancement yielded improved schedules that minimized manufacturing costs while still satisfying job due dates. The findings of this research had several important implications. First, there are significant differences in schedule performance between scheduling with and without alternatives. Second, allowing alternatives can greatly increase the ability to satisfy due dates under various shop floor conditions.

Thoney, et al. [33] developed a simulation technique for scheduling a multi-factory job shop, which included inter-factory transportation (with transportation assets treated as batch processors). She modified the Virtual Factory to demonstrate its applicability to a variety of problems, including the Military Deployment Problem. The parallel between the job shop scenario and the military deployment scenarios is demonstrated as follows: Each piece of unit equipment is a job. The ports through which the cargo (unit equipment) moves are factories through which jobs must process. Cargo, at various ports in the deployment network, is loaded and unloaded on different types of transportation assets. The transportation assets are batch processors moving jobs between factories. The batch processing time is the summation of the load, travel and unload times for each type of asset. Further modifications allowed the Virtual Factory to develop alternate routes for the unit equipment. This ensured that unit equipment was equitably balanced between the various routes; which in turn ensured that no transportation resources were idle while others were in constant use. She demonstrated an iterative technique to establish a lower bound for the maximum lateness using a linear program relaxation of a network flow model. This technique is based on work presented by Horn [18]. It involved establishing a feasible flow schedule with an established maximum lateness. The established or "accepted" maximum lateness was reduced by one and the

LP was resolved. The first infeasible flow established the lower bound for a feasible maximum lateness.

Chapter 3

The Military Deployment Problem and DANTE

The Military Deployment Problem is a very complex problem with two interrelated parts. The first part of the problem is to determine the allocation of transportation assets to routes over time that minimizes the total time for all cargo to reach the final destination. In other words, assign assets to minimize the deployment horizon or closure time. The transportation asset – route allocation can be classified as either static (allocation remains constant for duration of the deployment) or dynamic (allocation changes during the deployment process). The type of allocation in turn determines the type of network; a dynamic deployment network has a dynamic asset – route allocation. Once the initial problem has been solved and the transportation assets are subsequently assigned, the deployment network is finalized and the first part of the problem is completed. For this research, port operating characteristics remain fixed over time and the network is a static deployment network.

The DANTE tool developed by Hodgson et, al [17] was designed to provide a lower bound on the time needed to deliver a given deployment package to the theater of operations ($LB(C_{\max})$) using a static deployment network. This tool is so quick and easy to use that the deployment planner can perform effective sensitivity analysis concerning the impact of the transportation asset composition and assignment on the deployment closure time. The basic assumptions and structure of the DANTE network is developed in detail. Then a proof is given that shows that the minimum cost objective function formulation used by DANTE minimizes the deployment closure time.

3.1 The Deployment Analysis Network Tool Enhanced, DANTE

DANTE models any military deployment scenario on a predetermined and generalized deployment network. It performs network flow analysis to provide a lower bound on the total amount of time needed to transport the deployment package to the final destination, LB(C_{max}). The fundamental network structure of DANTE is depicted in Figure 3.1.

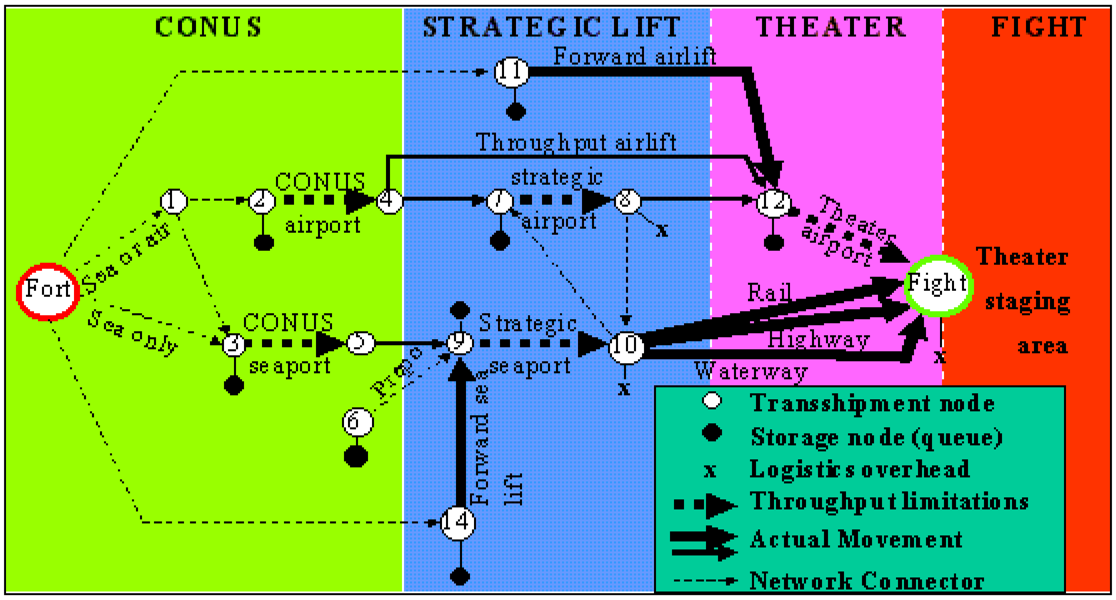


Figure 3.1 DANTE's Fundamental Network Structure

The acronym CONUS refers to the Continental United States. Although multiple forts as well as airports and seaports exist within CONUS, DANTE aggregates all forts, airports and seaports into one fort, airport and seaport. This facility aggregation also occurs within the strategic lift and theater portion of the network. The facility aggregation is a major simplifying assumption DANTE employs which creates an extremely generalized network structure that can represent all deployment scenarios. Another simplifying assumption in DANTE is the aggregation of all unit equipment. The equipment for all deploying units is consolidated into one equipment list. Furthermore, the equipment characteristics of area and volume are completely ignored. Instead, the equipment is represented as a total weight requirement to be shipped. The equipment is flowed through the network via several possible routes. The equipment departs CONUS through the aerial and/or seaport of embarkation (APOE, SPOE). This equipment can either go directly to the Theater ports of debarkation (APODs) or through strategic ports (APODs, SPODs) with further movement to the Theater Staging Area.

The various nodes of the model (air and seaports) are limited in their capacity to process equipment. These limited capacities are the actual (or assumed) capacities of the facilities to be utilized and are expressed as STONs/day. The various transportation routes (air, land and sea) are also limited in their capacities. These limitations are a function of the specific assets available (number of and type of aircraft, ships, etc.) and the distances involved. Because of capacity limitations, material may be delayed (queued) at certain points in the network (port nodes). Also, the flow of material within the transportation links of the model is delayed by the length of time associated with the particular link (i.e., flight time, shipping time, etc.). Although the network depiction in

Figure 3.1 appears fairly simplistic, the actual network structure employed by DANTE is quite a bit more complicated.

DANTE represents time in discrete six-hour segments. Consider the network in Figure 3.1. This network structure is represented for each time period, with periods stacked one on top of another. Figure 3.2 shows a partial view of this network. This stacked network displays movement through time as well as space; a plane, leaving the CONUS APOE at time 0 (node 4, level 1) and taking 12 hours to reach the strategic APOD, reaches that node at time 2 (node 7, level 3). For each possible transportation passage, there is an arc connecting the nodes and spanning time.

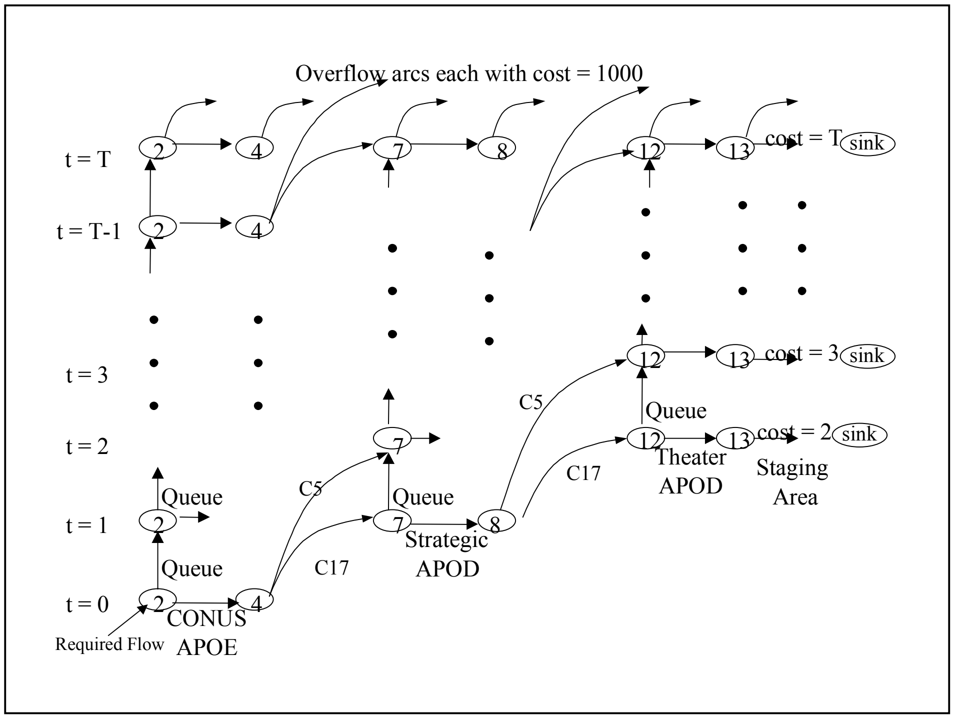


Figure 3.2 Partial Stacked DANTE Network

As mentioned previously, the model is capacitated at a number of points. All queuing arcs have infinite capacity because it is assumed that cargo storage space is unlimited. The port limitations are modeled by splitting each port into an arrival node and a departure node. A capacitated arc connecting the nodes of a particular port represents the port capacity. This capacity is expressed as an amount of short tons (STONs) per period and is a simple input parameter for the tool. The capacities of the transportation links (air and sealifts) are computed from other input data. Each link requires specific input data, such as the number of different types of assets operating on that link, the number of each type of asset, the travel speed, cargo carrying capacity (in STONs), the amount of time required to load and unload the asset, and the distance that asset will travel. For example, the capacity of the strategic air link (arc 4-7) for C17s would be computed using the following formula:

$$StratAirCap_{C17} = \frac{(\#C17)(capacityC17)}{\left[\left(2 * StratAirDistance / speedC17\right) + loadtimeC17 + unloadtimeC17\right]}$$

This formula computes the period carrying capacity of the arc and is expressed in STONs per period. Note that the asset's one way travel time is simply the distance divided by the speed of the asset. The arc's period capacity, however, uses the asset's round trip travel time in addition to the time spent loading and unloading the asset. This total time represents the amount of time required for the asset to return to its origin port, thus becoming available to carry new cargo. The asset is assumed to return empty or deadhead back; the return trip serves to degrade the capacity of the link. All travel time calculations are rounded off to integer amounts. Using the network structure shown in Figure 3.2, DANTE's network flow model is developed in the next section.

3.2 DANTE's Network Flow Model

The problem of determining the flow of equipment through a fixed deployment network in order to minimize the deployment closure time can be mathematically represented by using a multiperiod minimum cost network flow model. Nodes, arcs and time periods characterize the network depiction of the deployment model illustrated in Figure 3.2. Unit equipment (required flow) enters the network at time 0 (level 1 or period 1) at the origin airport and flows through the network depending on the capacity of each arc. All unit equipment is assumed to be available at the same time. The total deployment window, *T*, is broken into discrete time periods, *t*, of equal length representing a six-hour time span. *T* is an estimate of the total amount of time required for the delivery of the complete deployment package. The airports are the nodes of the network while the arcs connecting the nodes represent the movement of all cargo between ports. Note that two movement arcs leave node *4* in each period. This represents the two routes to which C17s and C5s are assigned to fly from the CONUS airport to the strategic airport. The capacity of a movement arc depends on the type and number of planes assigned to the arc and the distance involved. An additional terminal or sink node in each period connected by an arc from the departure node of the final airport closes the network. Flow across this arc represents equipment arrival at the staging area. At the end of the deployment time window, all nodes are connected to the sink node by overflow arcs. These are needed to ensure all cargo can reach the sink node. If any of the overflow arcs have a positive flow, the specified deployment time window, *T*, is too narrow given the number of planes available and the movement requirement. These overflow arcs are needed to maintain feasibility of the mathematical model.

The classic form of the multiperiod minimum cost network flow problem, MMCNFP, (Aronson and Chen [1]) is as follows:

Let N(*t*) be the set of nodes in period *t*, *t* = 1, …, *T*. A node is represented as the two-tuple (*i*,*t*) ∈ N(*t*), where *i* is the node number, for *i* = 1, …, n(*t*), and *t* is the time period. In most models, the node sets do not change from one time period to the next.

Let A(*t*) be the set of arcs which have their origin nodes (*i*,*t*) in N(*t*), i.e., in period *t*. An arc is represented by the two-tuple of nodes, $((i,t),(j,t'))$ ∈ A(*t*), where (j,t') is not defined for $t' > T$.

Let $N_i^+(t) \subseteq N(t)$ denote the set of nodes (j,t') ∈ N(t') for which arc $((i,t),(j,t'))$ ∈ A(*t*), i.e., the set of nodes which have arcs pointing away from node (*i*,*t*). Let $N_i^-(t) \subseteq N(t)$ denote the set of nodes (j,t') ∈ N(t') for which arc $(j,t'),((i,t))$ ∈ A(t'), i.e., the set of nodes which have arcs pointing towards node (*i*,*t*).

Let r_{it} be the requirement of node (*i*,*t*). The unit cost and flow capacity for arc $((i,t),(j,t'))$ are $c_{itjt'}$ and $u_{itjt'}$ respectively. The MMCNFP may be stated as follows:

$$Min \sum_{t=1}^{T} \sum_{(i,t)\in N(t)} \sum_{(j,t')\in N_i^+(t)} c_{itjt'} x_{itjt'}$$

$$s.t. \sum_{(j,t')\in N_i^+(t)} x_{itjt'} - \sum_{(j,t')\in N_i^-(t)} x_{jt'it} = r_{it} \qquad (i,t)\in N(t),\ \ t=1,\ldots,T$$

$$0 \le x_{itjt'} \le u_{itjt'} \qquad ((i,t),(j,t'))\in A(t),\ \ t=1,\ldots,T$$

where $x_{itjt'}$ = amount of flow sent from node (*i,t*) to node (*j,t'*)

The choice of the length of the deployment window, *T*, is critical because it directly affects the number of decision variables in the problem. For example, if the deployment window is 50 days and the time period length is six hours, the number of time periods is 200. Using the network represented in Figure 3.2, there will be 200 arcs alone from node *13* to the sink node over which the amount of flow must be determined. In general, there is an arc for each time period for all connected nodes in the network.

Trainor [34] developed the following linear programming (LP) formulation of DANTE. The subscripts for the decision variables (the amount of flow to send over an arc) specify the origin node *i* and the destination node *j* of an arc and the time periods over which the arc spans. The decision variable is:

- $W_{itj(t+p)}$, the amount of flow, in STONs, that leaves node *i* in period *t* and arrives at node *j* in period *t+p*, for all $itj(t+p)\in S$. *S* is the set of connected arcs for all *i* and *j*, and *p* represents the transit time from *i* to *j*. On movement arcs, these variables include a subscript *a* indicating the type asset that moves the cargo. Specifically, $W_{aitj(t+p)}$ is the amount of flow transiting a movement arc for all *a* and $itj(t+p) \in P$, where *P* is the set of movement arcs. $P \subseteq S$

A primary concern of deployment planners is the time to close the deploying force on the final destination. The objective of this LP formulation of DANTE is a function of time. Specifically, the objective is to minimize the time for all equipment to flow through the network given the constraints specified. To do this, an increasing (with time) cost is assigned to the flows along the arcs from the departure node of the final destination port(s) to the sink node. In Figure 3.2, this is the arc from node *13* to the staging area (sink) in each time period. Flows on overflow arcs, as represented in Figure 3.2, are assigned a significantly larger cost to make these arcs undesirable. The flows over all other arcs have zero cost. This results in the following objective function:

$$Minimize\left(\sum_{itjt \in FD} t * W_{itjt} + (1000) * \left(\sum_{itjt \in OV} W_{itjt} \right) \right)$$

where *FD* is the set of all arcs, excluding overflow arcs, connected to the sink node; i.e., *FD* is the set of arcs connected to the final destination within the specified deployment window (*T*). *OV* is the set of all overflow arcs connected to the sink node; the set of arcs which reach the final destination outside of the deployment window. If the length of the deployment window is too short for the established movement requirement then flow values on the overflow arcs will be positive. The deployment planner will not get a valid estimate of the total time required to close the deployment package. In this case, the length of the deployment window would be extended until all flow occurred on the set of arcs in *FD*. All further discussion is based on the assumption that the deployment window length is of suitable duration for the established network and movement requirement. The objective function is therefore simplified to the following:

$$Minimize\left(\sum_{itjt \in FD} t * W_{itjt} \right) \tag{3.1}$$

The objective function is linear with respect to the decision variables.

The constraint set is primarily defined by two types of equations. The first is a node balance equation for each node for each time period. These equations are of the form "Flow Out – Flow In = 0". For example, consider node *7* from Figure 3.2, the strategic Airbase arrival node. The flows in/out of node *7* are depicted in Figure 3.3.

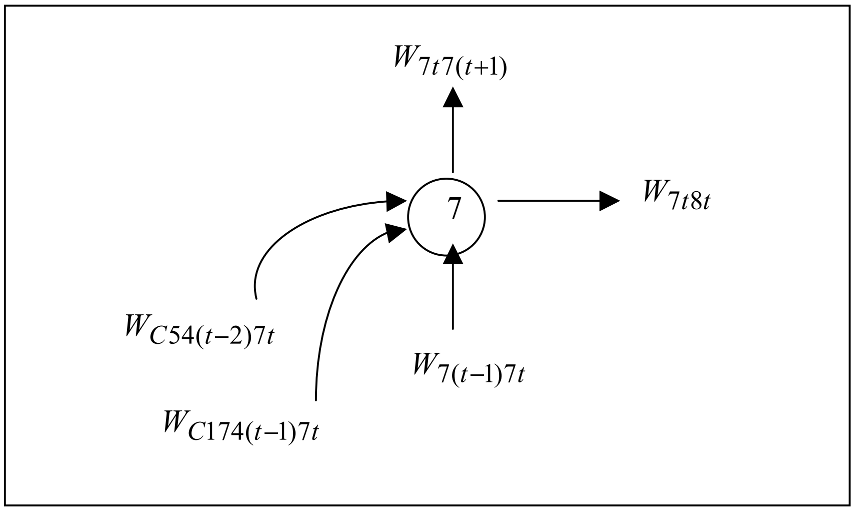


Figure 3.3 Example Node Flows

The flows into node *7* include cargo queued in the previous period, $W_{7(t-1)7t}$, and cargo brought by C5s, $W_{C54(t-2)7t}$, and C17s, $W_{C174(t-1)7t}$. In this example, the time required to move cargo from node *4* to node *7* is two periods for a C5 or one period for a C17.

This cargo movement time includes the one way travel time of the asset (based on distance and asset speed) and the times to load and unload the asset. The flows out of node *7* include cargo that will be put on planes leaving the strategic airport for the theater airport, W_{7t8t}, and cargo that is queued to the subsequent period, $W_{7t7(t+1)}$. The flow balance equation for this node is:

$$W_{7t7(t+1)} + W_{7t8t} - W_{7(t-1)7t} - W_{C174(t-1)7t} - W_{C54(t-2)7t} = 0.$$

The general format of each of the node balance equations is developed next. The movement requirements are flows into an origin port arrival node in the time period that equipment first becomes available to move. The node balance constraints corresponding to these requirements are:

$$W_{iti(t+1)} + W_{itjt} = CARGO_{it}, \forall it \in D \tag{3.2}$$

where *D* is the set of origin port arrival nodes in a given time period at which the movement requirements ($CARGO_{it}$) first become available for movement (*D* represents the set of origin port arrival nodes).

The following equation pertains to all nodes that are not origin port arrival nodes. Node (*i,t*) is the current point and time of reference. The general form of the node balance constraint is:

$$\sum_{j \in J_i} W_{itj(t+p)} - \sum_{o \in O_i} W_{o(t-p)it} = 0, \ \forall it \notin D \tag{3.3}$$

where J_i is the set of all destination nodes for *i*, and O_i is the set of origin nodes for *i*.

The second type of equation models the capacity restrictions for the different types of arcs. Each arc has a fixed capacity; the total flow on that arc can not exceed the arc's capacity. The general form of the arc capacity limitation constraint is:

$$0 \le W_{itj(t+p)} \le U_{itj(t+p)}$$

for each arc, where $U_{itj(t+p)}$ is the upper bound. The ability to store cargo at each port is assumed to be infinite but can be constrained without loss. Hence there is no upper bound constraint on the flows on cargo queuing arcs. All ports have a throughput capacity limitation modeled as:

$$W_{itjt} \le TP_{ij}, \forall t \text{ and } ij \in R \tag{3.4}$$

where TP_{ij} is the period throughput capacity of the port (fixed for all t) represented by arc ij and R is the set of arcs that represent ports. The throughput capacity is a function of the port's MOG, the largest asset able to transit through the port and the asset load time; it is computed as:

$$TP_{ij} = \frac{MOG * Asset\ capacity}{asset\ load\ time}$$

The upper bound on a movement arc capacity, CAP_{aij}, is a function of the type of asset, the number of assets allocated to the arc and the asset return time. The asset return time includes the asset load and unload times as well as twice the asset movement time. An example of the movement arc capacity computation was presented in section 3.1. The cargo movement constraint is modeled as:

$$W_{aitj(t+p)} \le CAP_{aij}, \ \forall a,t \text{ and } itj(t+p) \in P \tag{3.5}$$

where a is the type of transportation asset, CAP_{aij} is the period weight capacity (in STONs) for the combined assets of type a operating on that arc and P is the set of movement arcs. The formulation includes an arc constraint for each port and each asset in each time period.

To insure that all required flow gets through the model, the following constraint is added:

$$\sum_{itj(t+p)\in FD} W_{itj(t+p)} = \sum_{it\in D} CARGO_{it} \tag{3.6}$$

where FD is the set of all arcs connected to the sink node within the deployment window.

The complete LP formulation (problem *DANTE*), with accompanying equation numbers, is summarized next.

Problem *DANTE*

$$Minimize\left(\sum_{itjt\in FD} t * W_{itjt}\right) \tag{3.1}$$

$$Subject\ to:\ W_{itj(t+1)} + W_{itjt} = CARGO_{it}, \forall it \in D \tag{3.2}$$

$$\sum_{j\in J_i} W_{itj(t+p)} - \sum_{o\in O_i} W_{o(t-p)it} = 0, \forall it \notin D \tag{3.3}$$

$$W_{itjt} \le TP_{ij}, \forall t \text{ and } ij \in R \tag{3.4}$$

$$W_{aitj(t+p)} \le CAP_{aij}, \forall a, \text{and } ij \in P \tag{3.5}$$

$$\sum_{itj(t+p)\in FD} W_{itj(t+p)} = \sum_{it\in D} CARGO_{it} \tag{3.6}$$

$$W_{itj(t+p)} \geq 0, \forall ij \text{ and } t \qquad (3.7)$$

where

- FD is the set of all arcs, excluding overflow arcs, connected to the sink node;
- D is the set of arrival port nodes, in a given time period, at which the movement requirements ($CARGO_{it}$) first become available for movement;
- p is the transit time between nodes;
- J_i is the set of destination nodes for node i;
- O_i is the set of origin nodes for node i;
- TP_{ij} is the period throughput capacity of the port represented by arc ij;
- R is the set of arcs that represent ports;
- CAP_{aij} is the period weight capacity (in STONs) of the combined type a transportation assets operating on the assigned route;
- P is the set of movement arcs;
- T is the deployment window of sufficient length.

3.3 DANTE Minimizes Deployment Closure Time

DANTE's problem formulation, although presented as a multiperiod minimum cost network flow model, does minimize the amount of time used to deliver the total movement requirement to the staging area. In other words, DANTE minimizes the deployment closure time. The proof of this statement is presented in the following sections.

3.3.1 Background

To be able to prove that DANTE minimizes deployment closure time, we must first present some new definitions and observations. Three new network structures are defined and illustrated. A relationship between the DANTE network, as defined in section 3.2 and one of the new networks will be shown to exist.

3.3.1.1 Definitions and Observations

Define the Period Closure Capacity of the network as the total flow capable of crossing the arcs connected to the sink node (Final Destination) in a given period. The set of arcs connected to the sink node are the closure arcs. Portions of the total movement requirement become "available" to close after they arrive at the departure node of the final destination port. If flow is available to close at time *t*, then that flow, unless "closed" or delivered to the sink, remains available to close at the next time period.

Let an Original Network be a 2 dimensional picture of the deployment scenario containing all bases and routes from one base to another involved in the deployment scenario. An example Original Network is presented in Figure 3.4.

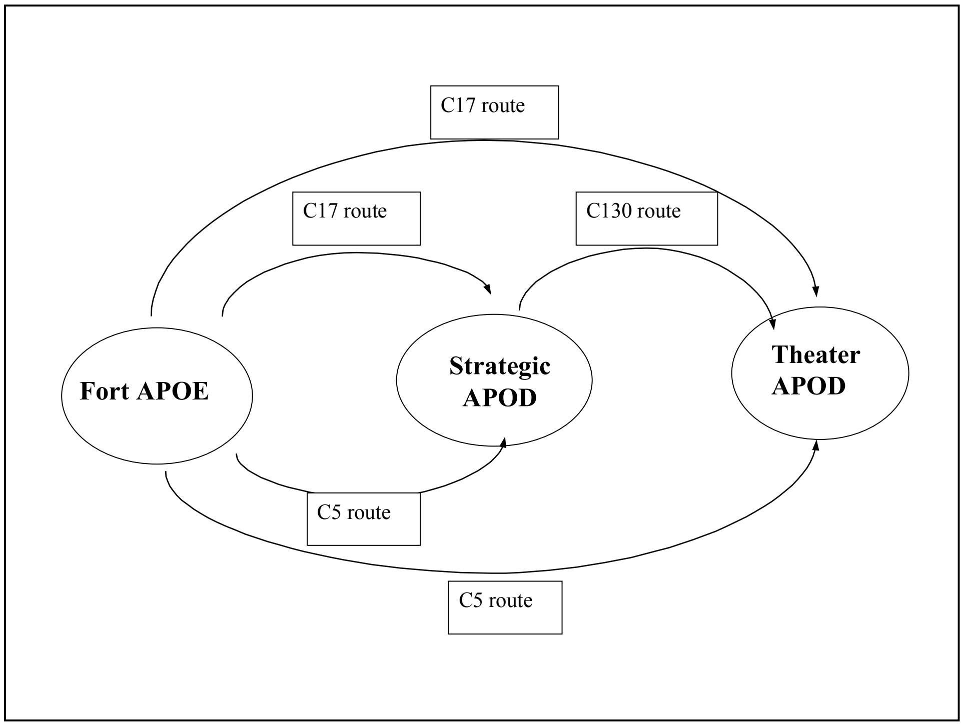


Figure 3.4 Example Original Network

Define a Modified Network to be the same as the Original Network with two modifications. The first is that each port is replaced by two nodes with an arc connecting them of capacity equal to the period throughput as defined in the DANTE Model, section 3.2. The second is that the capacity of all other arcs is equal to the capacity of corresponding arcs in the DANTE Model. An example of the Modified Network is depicted in Figure 3.5. The Fort APOE is represented by nodes F1 and F2 and the port throughput capacity by the connecting arc with capacity Fthru. Similarly, the Strategic APOD is represented by nodes S1 and S2 and the capacitated connecting arc Sthru, while the Theater APOD is represented by nodes Th1 and Th2 and the capacitated connecting arc Ththru. The movement arcs between ports and their capacities are identified by the asset type (C17, C5, or C130) and the nodes between which the asset operates.

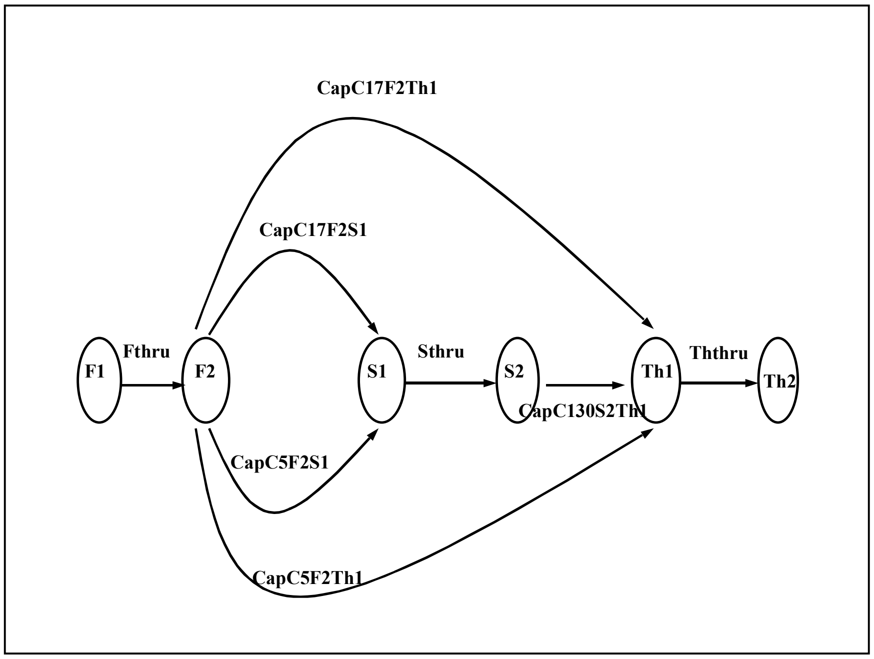


Figure 3.5 Example Modified Network

This modified network has several parameters associated with it; these parameters are as defined in Table 3.1.

Table 3.1 Modified Network Parameters

Parameter	***Definition***
N	total number of distinct paths from an origin node to the destination (a path is distinct if it contains at least one arc that is different from another path)
P_i	path *i* where the paths have been sorted in increasing order of the time it takes to get from its origin to the destination.
$L(P_i)$	time, in periods, path *i* takes to get from its origin to the destination.
$M(S)$	the maximum flow solution to the modified network when only the paths in *S* are included in the network.

There are four distinct paths from the origin to the destination in Figure 3.5. Suppose that paths 1, 2, 3 and 4 take, respectively, 2, 3, 4 and 5 time periods to reach the destination.

Figures 3.6 through 3.9 are Modified Networks in which only certain paths are included. For ease of reading, the arc capacities have been omitted.

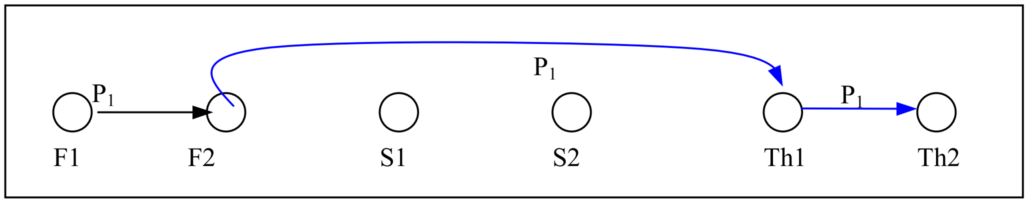


Figure 3.6 Modified Network With Only Path 1 Included

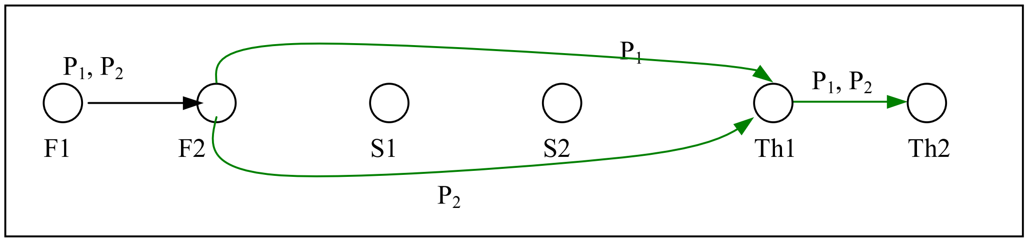


Figure 3.7 Modified Network With Only Paths 1 and 2 Included

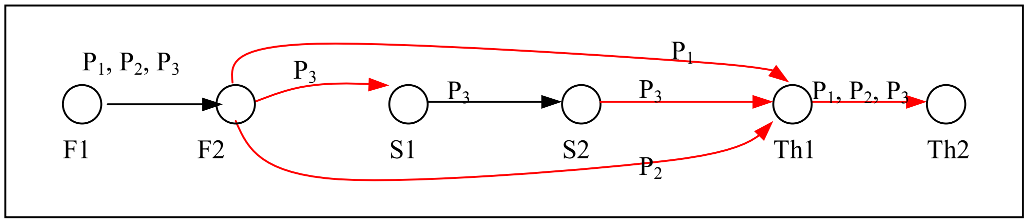


Figure 3.8 Modified Network With Only Paths 1, 2, and 3 Included

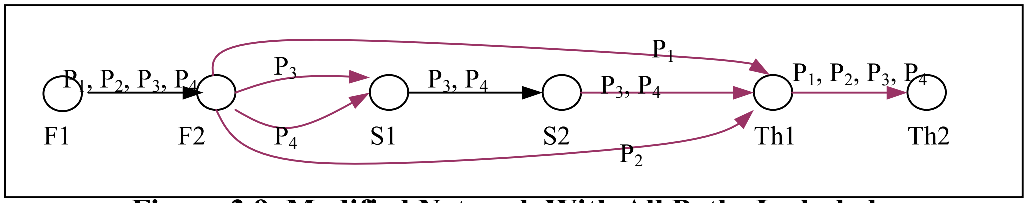


Figure 3.9 Modified Network With All Paths Included

Define a Modified DANTE network to be a DANTE network in which the capacity of all queuing arcs is set equal to zero except those at the origin(s). Then, Figures 3.10 though 3.13 show the Modified DANTE network based on the Original Network (as depicted in Figure 3.4) for different lengths of the deployment window.

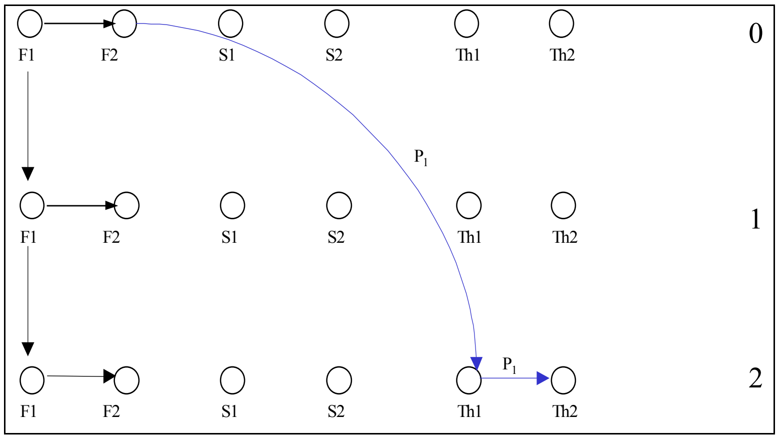


Figure 3.10 Modified DANTE Network from Time 0 to 2

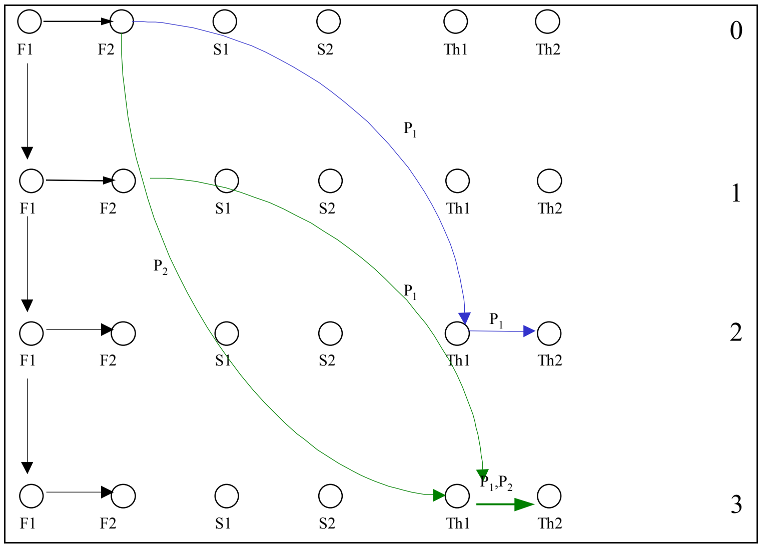


Figure 3.11 Modified DANTE Network from Time 0 to 3

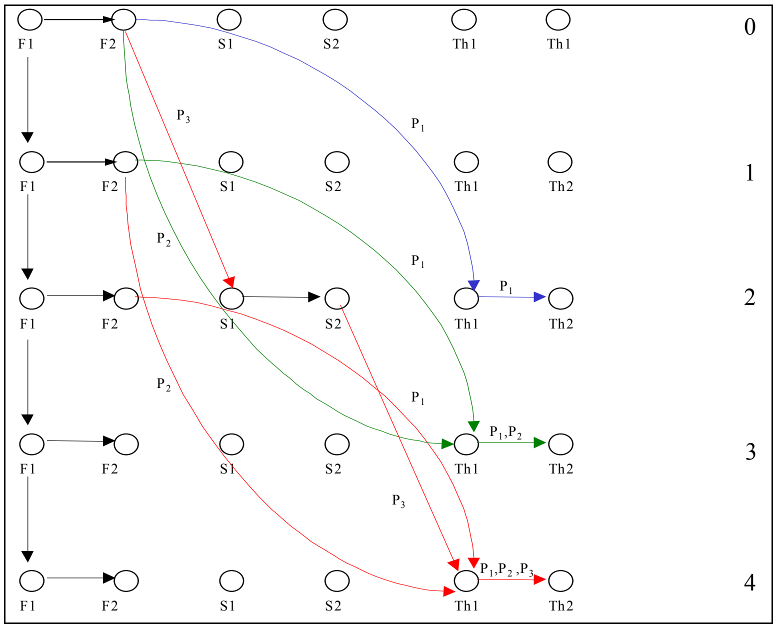


Figure 3.12 Modified DANTE Network from Time 0 to 4

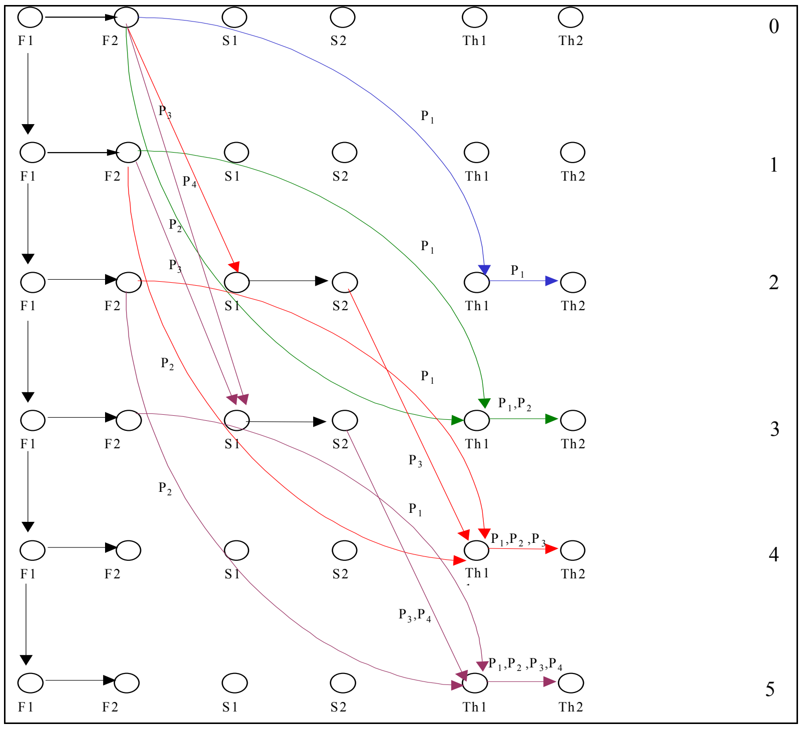


Figure 3.13 Modified DANTE Network from Time 0 to 5

As seen in Figures 3.10 through 3.13, the Modified DANTE network "evolves" over time as paths reach the final destination, until all paths are able to reach that point.

Observe that the Modified DANTE network is essentially an alternate representation of the series of Modified Networks (figures 3.6 through 3.9) where the origins of the Modified Network are connected by queuing arcs. For example, in Figure 3.10, the only path to the destination is the same as depicted in Figure 3.6. Also, in Figure 3.11, the only path that reaches the destination at time 2 is the same as that shown in Figure 3.6, and the only paths that reach the destination at time 3 are the same ones seen in Figure 3.7. The same pattern is evidenced in Figures 3.12 and 3.13.

The evolutionary process of the Modified DANTE network yields different maximum flow solutions at different points in time, based on the specific time interval. These maximum flow solutions are identified in the Table 3.2.

Table 3.2 Modified DANTE Network Maximum Flow Solutions

Time Period Intervals	**Maximum Flow Solution For Each Period**
$[\,0, L(P_1)\,)$	0
$[\,L(P_1), L(P_2)\,)$	$M(P_1)$
$[\,L(P_2), L(P_3)\,)$	$M(P_1, P_2)$
$\vdots$	$\vdots$
$[\,L(P_{N-1}), L(P_N)\,)$	$M(P_1, P_2, \ldots, P_{N-1})$
$[\,L(P_N), \infty\,)$	$M(P_1, P_2, \ldots, P_N)$

Observe that the DANTE network only differs from the Modified DANTE network through the inclusion of several connecting arcs of infinite capacity. The additional arcs are queuing arcs connecting port arrival nodes at successive times, and arcs connecting the departure nodes of the final destination ports to the sink node. These arcs of infinite capacity provide no restrictions to the period closure capacity of the network. The obvious assertion is that the cumulative closure capacity from time period 0 through time period x for the original DANTE network is equal to the sum of the maximum flow solutions per time period in the Modified DANTE network from time period 0 to time period x. For example, if $L(P_1) = 15$, $L(P_2) = 17$, and $L(P_3) = 20$, then the cumulative closure capacity through time period 19 is $2M(P_1) + 3M(P_1, P_2)$. This does not imply that the flow of say time period 18 cannot be greater than $M(P_1, P_2)$. It just implies that if the flow is greater, then the flow in periods prior to time period 18 must be less than their capacity per time period in the table above.

3.3.1.2 DANTE Network and the Modified Network

Because the Modified DANTE network is an alternate representation of the series of Modified Networks for the same Original Network, there is also a relation between the DANTE network and the series of Modified Networks based on the same Original Network.

Assertion: The cumulative closure capacity for time period 0 through time period x in the DANTE network, can be found by summing the Maximum Flow Solutions per time period from time periods 0 to x in the series of Modified Networks based on the same Original Network.

Proof: For each network in the series of Modified Networks associated with time periods 0 to x, there is a corresponding sub-network in DANTE that maintains the same capacity constraints while breaking up the individual paths to represent the time they take. These sub-networks are independent except for the additional queuing arcs of infinite capacity connecting nodes representing the same port in different time periods. The only other arcs in the DANTE model that are not present in the series of Modified Networks are arcs of infinite capacity connecting the destination at different time periods to the sink. Therefore, the combined minimum cuts of the Modified Networks must be equal to the minimum cut in the DANTE Model. Since the minimum cut in a network must equal the maximum flow, the cumulative capacity through time period x in DANTE must be equal to the sum of the maximum flow solutions in the Modified Networks corresponding to periods 0 through x.♦

3.3.2 DANTE Closure Time Proof

Some final observations and assumptions are made before presenting the formal proof that DANTE's objective function, expressed as a minimum cost function, does minimize closure time. First, for the goal of minimizing the cost of the weighted closure flow in DANTE, it cannot be optimal to delay flow that could have finished at an earlier time period to a later one (recall that the weighted penalty corresponds to time of closure). Also, there will be an optimal solution for the goal of minimizing closure time where this non-delay of flow holds. Therefore, in the optimal solution to both of these problems, we will assume the per period closure capacities will not exceed the maximum flow solutions identified in Table 3.2. Consequently, we can translate this problem (to minimize the cost of the weighted closure flow) into a bin-filling problem. Let the period closure capacity be referred to as a bucket of capacity, or a bin to be filled. When there is one origin base, there is essentially one bin in each time period with the capacities in Table 3.2 above to which the weight at the origin can be assigned. When there is more than one origin, there may be multiple bins in each time period, with the sum of the bin capacities equal to the capacity in the above table. If, for example, there are 2 origins, then there may be as many as 3 bins. Each origin may have a bin to which only weight from that origin may be assigned; these represent, primarily, the capacity of paths, on which no flow from another origin may travel over (these paths have no common legs with paths that originate at an alternate origin). In addition, there may be a bin to which weight from either origin can be assigned (paths have common arcs with links to both origins).

Assertion: The optimal solution to minimizing the cost of the weighted closure flow in the DANTE network has minimum closure time.

Proof: Suppose to the contrary, the optimal solution does not have minimum closure time, b, then two conditions exist. First, there is some flow closing at a time $c > b$ that is available to close at or before b. Second, there is unused closure capacity at time $a \leq b$ when there is flow available to close at that time. Take one unit of weight, w, with origin y, that is in a bin corresponding to closure at time c where $c > b$. Then one of three options must apply. Let ■ indicate one full bin; let □ indicate one non-full bin.

Option 1. There is a non-full bin corresponding to closure at time $a \leq b$ in which w can be placed (bin accepts flow from origin y). Figure 3.14 depicts this option for a Single Origin and a Multiple Origin Network.

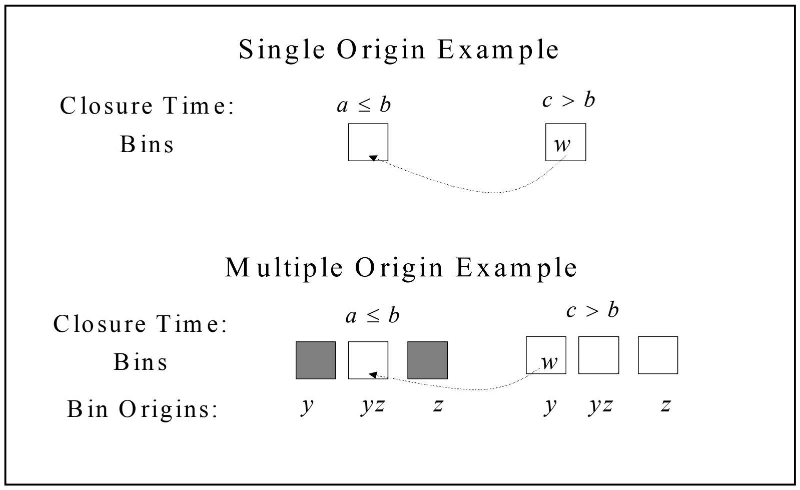


Figure 3.14 Option 1 Examples

Option 2. There is a full bin corresponding to closure at time $a \leq b$ that accepts flow from multiple origins including y from which we may replace a unit of weight x, with origin z, already in that bin and consequently place x in another non-full bin corresponding to closure at some time $a \leq b$. Figure 3.15 depicts this option.

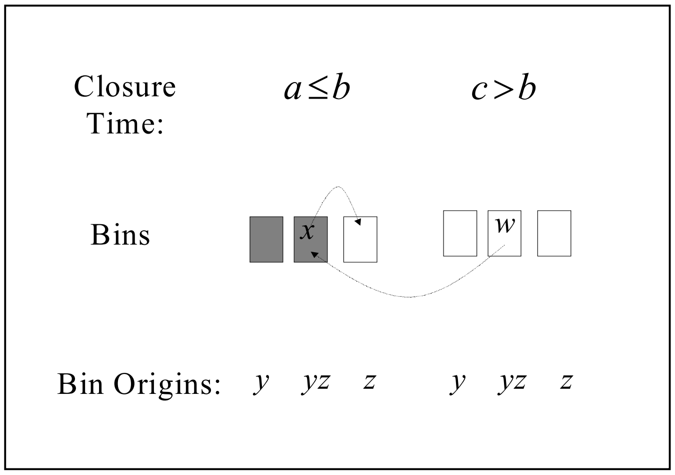


Figure 3.15 Option 2 Example

Option 3. The same situation as in Option 2 occurs except that there is no non-full bin in which to directly place *x*. Then *x* must displace flow from another origin that in turn displaces flow from another origin, etc. The process of shuffling flows between bins continues until a unit of weight can be placed into a non-full bin corresponding to closure at time $a \leq b$.

In all cases the cost of the total weighted flow has been reduced through the shuffling of available flows to earlier bins. Hence, DANTE's objective function ensures that deployment closure time is minimized. ♦

Chapter 4

DANTE's Evolution

As mentioned previously, the Military Deployment Problem is an extremely complex problem with two distinct yet interrelated parts. The first part focuses on the development of a deployment network geared towards minimizing the deployment closure time. DANTE is an effective planning tool used by deployment planners to establish a lower bound for the deployment closure time, LB(C_{max}), for a given deployment package and a generalized, static deployment network. The second part of the deployment problem is to determine the schedule of equipment onto the transportation assets to meet the required delivery dates. In other words, schedule equipment to minimize the maximum lateness of the deployment package, L_{max}.

This chapter is geared towards determining a lower bound on the maximum lateness, LB(L_{max}), of the deployment package, given a fixed transportation asset – route assignment. The lower bound will be developed in the context of a specific military deployment scenario, but remains applicable to all deployment scenarios. First, the specific deployment scenario will be presented. Next a modification of the DANTE tool, presented in detail, will provide a deployment planner with two additional capabilities. The first is the capability of tracking the flow of specifically identified equipment (a unique commodity) through the deployment network. The second is the capability of determining individual commodity closure times. The next modification of this tool will provide the deployment planner the capability of sequencing the individual commodity flows. The final modification allows for an iterative application of this tool to establish the LB(L_{max}) of a deployment package.

4.1 A Military Deployment Scenario

This scenario represents a typical divisional-sized element deployment. A standard set of transportation assets and ports is used in this scenario; however, the solution technique can accommodate any set of assets and ports and any sized unit deployment. The set of transportation assets and their operating characteristics is provided in Table 4.1.

Table 4.1 Transportation Asset Operating Characteristics

Aircraft	**Speed (Knots)**	**Weight Capacity (STONs)**	**Load Time (hours)**	**Unload Time (hours)**	**Maintenance Time (hours)**
C17	410	45	2.25	2.25	2.25
C5	409	61.3	4.25	3.25	3.25
C130	270	12	1.5	1.5	1.5

Knots is a marine term meaning nautical miles per hour. The set of ports and their infrastructure characteristics are provided in Table 4.2. The acronym STON refers to a short ton; one short ton is equivalent to 2,000 pounds. MOG refers to the Maximum-On-Ground capacity of an airfield or the maximum number of aircraft able to be accommodated at any one time. These characteristics, along with the number of transportation assets, by type, operating at each port, are used to compute the port throughput capacity in STONs per period. For the remainder of this research, one time period is equivalent to six hours unless otherwise stated.

Table 4.2 Port Infrastructure Characteristics

Port Facility	# Runways	MOG
Fort Campbell Army Airfield	1	5
Ramstein Air Force Base	1	5
Dhahran International Airport	1	5

This deployment scenario involves moving units by air from Fort Campbell, Kentucky, to Dhahran, Saudi Arabia. An intermediate stop (ISB) is available at Ramstein Air Force Base, Germany. The transportation resources available include three types of Air Force aircraft: the C5 Galaxy, C17 Globemaster and C130 Hercules. The C5 and C17 are long-range aircraft that will fly the US-Germany routes and the US-Saudi Arabia routes. The C130, a significantly smaller and slower plane, is best used on shorter routes. Its movement will be limited to the Germany-Saudi Arabia route. Graphically, this deployment scenario is shown in Figure 4.1.

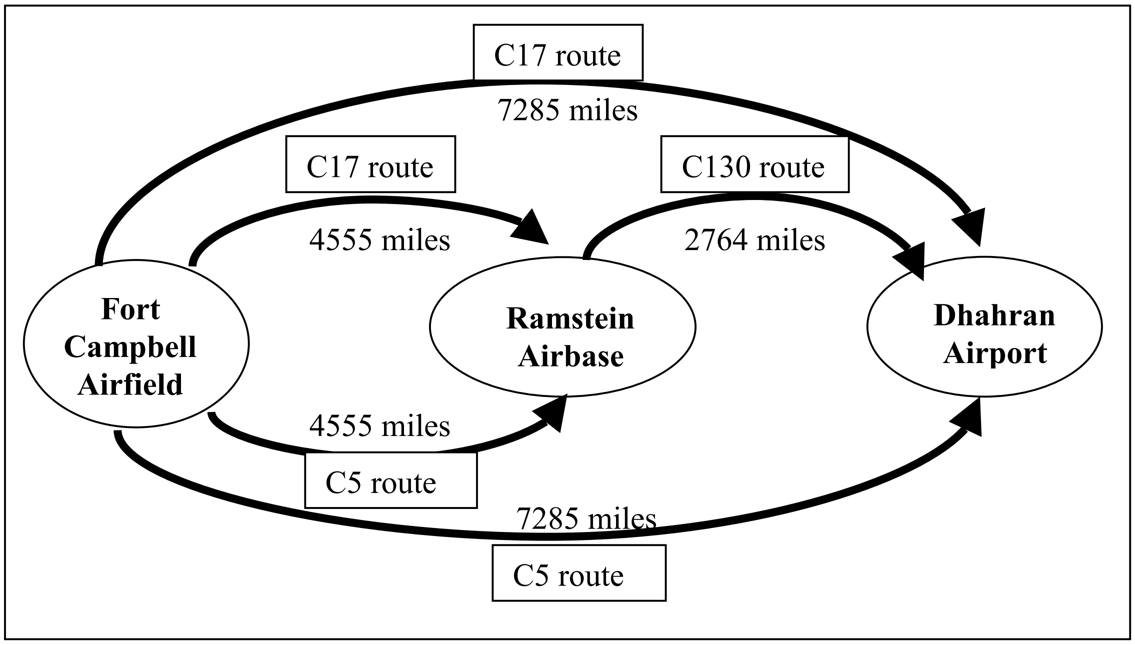


Figure 4.1 Military Deployment Scenario

4.2 Initial Changes to DANTE

Recall that DANTE models any military deployment scenario on a predetermined deployment network. It performs network flow analysis to provide a lower bound on the total amount of time needed to transport the deployment package to the final destination. One of the major deployment problem relaxations employed by DANTE is the port aggregation in various locations throughout the network. For the remainder of this research, all ports will retain their individuality, including geographical relationships to other identified ports. A second relaxation employed by DANTE is the movement requirement relaxation. DANTE ignores all cargo characteristics except weight. It aggregates all equipment into one all encompassing movement requirement identified by the total weight to be moved. A multicommodity variation of DANTE is developed next. In this variation, cargo is aggregated by weight into multiple requirements or commodities specified by certain equipment characteristics. The sum of the commodity weight requirements is equivalent to the total movement requirement. The resultant model, a multiperiod, multicommodity network flow model, also minimizes the deployment closure time, but has the added capability of tracking individual commodities and their closure times. This new model will be referred to as *COMFLOW*.

4.2.1 Computing Port Throughput Capacities

A port throughput capacity expresses the port's ability to process and clear equipment within a specified time period. A port has a limited amount of space for transportation assets. For example, a seaport with three berths can only receive and load or unload three ships simultaneously. This space restriction greatly affects the port throughput rate. Other factors that also impact the rate are the available manpower and specialized cargo handling equipment. A major simplifying assumption made in this model is that all personnel and cargo handling equipment remain available at all times (unless already engaged in loading/unloading the maximum number of transportation assets allowed), i.e. port work personnel never get sick nor does their machinery break down.

DANTE computes a port's throughput rate based on the port's MOG, the largest asset's weight carrying capacity, and the amount of time that asset spends at the port. For example, Campbell's airport throughput rate (in STONs per period) is computed based on using C5 aircraft (it is the largest plane in the scenario that Campbell can accommodate) in the following manner:

$$Cthru = \frac{MOG * Cap_{C5}}{loadtime_{C5}} * \frac{6hrs}{period} = \frac{5 * 61.3}{4.25} * 6 = 432.7059 \ \ stons / period$$

This period throughput rate is approximately equal to 1,731 stons per day. DANTE requires an integer valued daily throughput rate as input.

This method is a very simplistic means of determining a port's throughput rate and has a few shortcomings. First, the amount of time each aircraft spends on the ground varies according to the operation performed, i.e. loading or unloading. Also, habitually, each asset spends time in a maintenance bay after each flight, lengthening the amount of time each asset spends at the port. Another shortcoming of this method is its failure to adequately represent the composition and size of the deployment fleet actually operating at the facility. The actual number of C5's operating at a port might be considerably less than the other assets operating there. In that case, the actual port throughput rate would be quite different. To alleviate these shortcomings, two alternate methods to compute a port's throughput rate are proposed.

Recall that DANTE assumes the transportation asset to route assignment remains fixed over time. Both the methods described will continue to make that assumption. Therefore a given asset will always operate between the same two ports for the duration of the deployment. Each asset *i* has an associated time spent at the origin $O_i = L_i + M_i$ and time spent at the destination $D_i = U_i + M_i$ (where L_i, U_i, and M_i are the load, unload and maintenance time, respectively, of asset *i*).

The first method involves computing the asset throughput rate for each asset type operating at the port. The asset's throughput rate is computed as:

$$AssetRate_i = \frac{Cap_i}{G_i}, \text{ where } G_i = \begin{cases} O_i & \text{if port is the origin of asset } i \\ D_i & \text{if port is the destination of asset } i \end{cases}$$

The port throughput rate is based on the port's MOG and the maximum asset throughput rate. Some ports are quite large and can accommodate many assets. In some cases, a port's MOG is larger than the total number of deployment assets operating there. In these cases, using the MOG would overestimate the actual throughput rate of cargo. Let N be the total number of assets operating at the port. Then the port period throughput rate is computed in the following manner:

$$PortRate = \min(MOG, N) * \max_i(AssetRate_i) * \frac{6hrs}{period}$$

Based on the information presented in Tables 4.1 and 4.2 and the general network structure depicted in Figure 4.1, each port would have a maximum port throughput rate of 300 STONS/period. This rate is based on the C17's capacity and ground clearance times.

This method would tend to overestimate the amount of cargo actually processed through a port in a given period because it is based entirely on the asset that yields the largest throughput rate. This in turn, would ensure that DANTE, using this port throughput rate, would still provide a lower bound for the deployment closure time.

The second method to determine the port throughput rate considers all types of assets operating at the port, the particular operations (loading or unloading) taking place, the fleet size and the fleet composition. This method computes the port throughput rate based on the port MOG and the weighted average asset throughput rate. In addition to the times mentioned previously, each asset has a one way travel time T_i = distance/speed and a roundtrip time $R_i = 2T_i + O_i + D_i$. Based on these times, each asset also has an arrival rate (to a port) of $1/R_i$. Thus the weighted arrival rate is N_i/R_i, where N_i is the number of asset *i* operating on the route. The weighted average throughput rate at each port is then computed as:

$$Throughput = \min(MOG, N) * \frac{\sum \left(\frac{N_i}{R_i} * \frac{Cap_i}{G_i} \right) * 6}{\sum \frac{N_i}{R_i}}$$

$$\text{where } G_i = \begin{cases} O_i \text{ if port is origin of asset } i \\ D_i \text{ if port is destination of asset } i \end{cases}$$

This method assumes that all routes are "active" in all periods. Based on the distances and the assets' speeds involved, the first route capable of delivering cargo to the Dhahran port is the direct route flown by C17 aircraft, delivering cargo in time period 4. During this time period, Dhahran's largest possible port throughput rate is 300 STONs/period. During the next time period, period 5, and many subsequent periods, all routes are able to deliver equipment to the port. The port period throughput capacity for this time span is accurately represented by the described computation method, which yields a smaller throughput rate. This rate is smaller due to the inclusion of the additional asset types with significantly smaller capacity (C130) or longer ground times (C5). Finally, at some point in time, the amount of equipment at the origin base is depleted. From that time until all remaining equipment is delivered to Dhahran, the only cargo arriving at Dhahran is from indirect routes. The amount of equipment actually able to move through the Dhahran port is 120 STONs/period (based on the C130 assets), an amount less than estimated by this computation method.

This method, using a fixed port throughput rate over time, sometimes underestimates the actual port period throughput rate. Then DANTE, using this method to determine port throughput rates, would not provide a guaranteed lower bound for C_{max}. This method does provide a good approximation of the average port throughput rate for the duration of the deployment. The actual port throughput rate changes over time as the deployment network develops and routes become active. The port throughput rate which changes over time is beyond the scope of this research.

4.2.2 Establishing the Multiple Commodities

Similar to the approach used by Kim, et al [20], a commodity will be identified by the origin and due date information associated with each piece of equipment in the deployment package. The number of unique origin-due date pairs is the number of commodities; all equipment with the same origin and due-date will have the same commodity identifier. The weight of each piece of equipment with the same commodity identifier will be aggregated into a commodity weight requirement. The end result is that each commodity is identified by a unique origin-due date pair and has an associated weight or commodity movement requirement. For example, suppose three pieces of equipment were deploying from Fort Bragg and two pieces from Fort Campbell. The due dates and weight requirements of the pieces are identified in Table 4.3, along with a unique commodity identifier.

Table 4.3 Equipment with Commodity Identifiers

Piece of Equipment	Origin	Due Date	Weight (STONs)	Commodity Identifier
1	Bragg	3	2	1
2	Bragg	4	2	2
3	Bragg	3	1	1
4	Campbell	5	1	3
5	Campbell	4	3	4

Aggregating the weight requirements of all pieces with the same commodity identifier results in four commodities with associated commodity weight requirement. These commodities are identified in Table 4.4.

Table 4.4 Commodity Data

Commodity	Origin	Due Date	Requirement (STONs)
1	Bragg	3	3
2	Bragg	4	2
3	Campbell	5	1
4	Campbell	4	3

The total movement requirement identified in both tables is 9 STONs. The equipment characteristics of volume and area are ignored as in DANTE. Similar to DANTE's aggregation of all equipment into a single short tonnage movement requirement, this process aggregates equipment into multiple commodity short tonnage movement requirements based on origin and due date information.

4.3 Developing *COMFLOW*

The problem of determining the flow of multiple commodities through a fixed deployment network in order to minimize the deployment closure time can be graphically represented using a multiperiod, multicommodity network. In fact, both the graphical and mathematical development of this network are very similar to the procedures demonstrated in Chapter 3.

4.3.1 Graphical Representation

Unit equipment (required commodity flow) enters the network at time 0 at the origin airport and flows through the network depending on the capacity of each arc. All equipment (therefore all commodities) is assumed to be available at their respective origins at the same time. The airports are the nodes of the network while the arcs connecting the nodes represent the movement of all cargo (all commodities) between ports.

To model an airport throughput limitation, each airport is modeled as an arrival (1) and departure node (2) with a connecting arc representing the port throughput capacity in a period. Arcs for each type of plane connect the departure node from an airport to the destination airport's arrival node in a later time period determined by the cargo movement time for that type plane. The capacity of a cargo movement arc depends on the type and number of planes assigned to the arc and the distance involved. A sink node connected by an arc to the departure node of the final destination airport closes the network. At the end of the deployment time window, all nodes are connected to the sink node by overflow arcs. If any of the overflow arcs have a positive flow, the specified deployment time window, *T*, is too narrow given the number of planes available and the total movement requirement. Overall, there is a network for each discrete time period creating the "staircase" model depicted in Figure 4.2. For the multicommodity flow model, this network is additionally repeated for each commodity. All commodities compete for "space" on all the capacitated arcs.

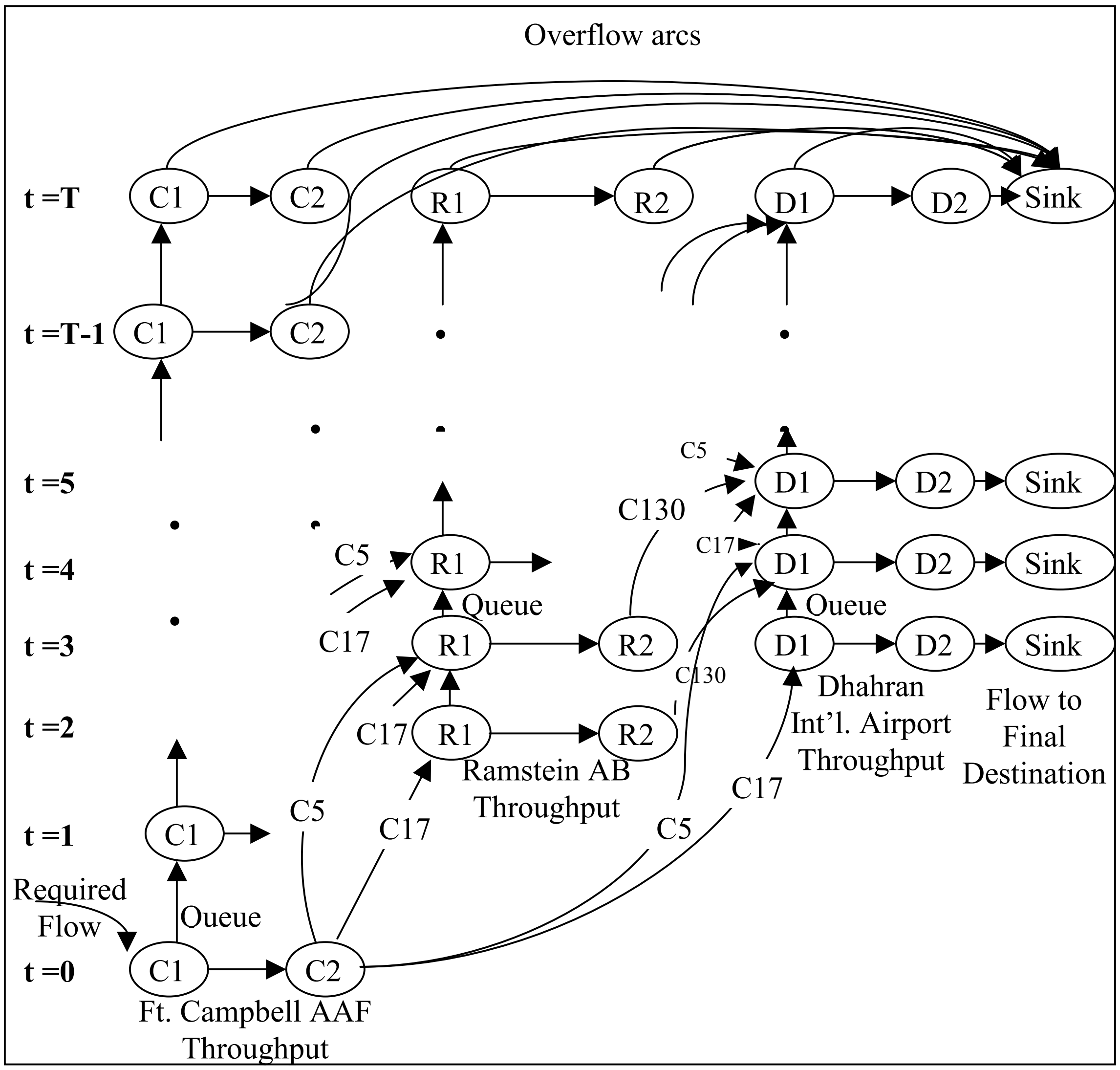


Figure 4.2 Network Representation of the Deployment Scenario

Each cargo movement arc links ports in different time periods based on the amount of time needed to move cargo between the ports. This time includes the amount of time needed to load and unload the asset moving the cargo, as well as the travel time of the asset. This total time represents the amount of time (in hours) the cargo spends on the asset to complete its trip to the next port. It is computed as follows:

$$CargoTime = \frac{\left(\left(\text{distance / speed}\right) + \text{load time} + \text{unload time}\right)}{6 hours / period}$$

As an example, consider the C5 route from Fort Campbell to Dhahran, arc (C2, D1) in Figure 4.2. The flight distance is 7285 miles and the C5 characteristics are as specified in Table 4.1. The C5 cargo time, in periods, is calculated as:

$$CargoTime = \frac{\left(\frac{7285}{409 * 1.1507} + 4.25 + 3.25\right)}{6} = 4$$

where 1.1507 is a factor that converts knots to miles per hour. This calculation is rounded off to integer amounts. Subsequently, the C5 cargo movement arc connects node *C2* in period 1 (time 0) to node *D1* in period 5 (time 4). Likewise, the C5 cargo movement arc connects *C2* in period 2 with *D1* in period 6. This continues until *T* for each cargo movement arc for this asset.

Each cargo movement arc also has an associated round trip time. This time is computed in a similar manner, but involves twice the distance and two maintenance sessions for each asset. This total time represents the amount of time required for the asset to return to its origin and become available to receive another cargo load. This time is used in determining the cargo capacity per period for each cargo movement arc. It is computed as follows:

$$RoundtripTime = \frac{\left(\left(2 * \text{distance / speed}\right) + \text{load time} + \text{unload time} + 2 * \text{maint time}\right)}{6 hours / period}$$

This multiple commodity flow problem can be represented as a multiperiod, multicommodity, minimum cost network flow problem with side constraints. The side constraints model commodity competition for "space" on all capacitated arcs. Each arc in the deployment network has explicit bounds on the total amount of cargo that can flow across it; these are computed as period capacities. All arc lower bounds are set equal to zero.

4.3.2 Mathematical Representation

This commodity flow problem is similar to the equipment flow problem that DANTE was developed to evaluate; however, this problem will provide information concerning individual commodity closure times. The commodity flow problem is still a relaxation of the original deployment problem due to the continued relaxation of the cargo characteristics of area and volume, but the ports are no longer aggregated. This deployment network more accurately represents a given deployment scenario. The subscripts for the decision variable (the amount of commodity *k* to send over an arc) specify the origin node *i* and the destination node *j* of an arc and the time periods over which the arc spans. The decision variable is

- $W_{kitj(t+p)}$, the amount of commodity *k*, in STONs, that leaves node *i* in period *t* and arrives at node *j* in period *t+p*, for all $itj(t+p) \in S$. *S* is the set of connected arcs for all *i* and *j*, and *p* represents the transit time from *i* to *j*. On movement arcs, these variables include a subscript *a* indicating the type asset that moves the cargo. Specifically, $W_{kaitj(t+p)}$ is the amount of commodity *k* flowed on a movement arc for all *a* and $itj(t+p) \in P$, where *P* is the set of movement arcs. $P \subseteq S$

The objective of this formulation remains a function of time. An increasing (with time) cost is assigned to the flows along the arcs from the departure node of the final destination port(s) to the sink node. Flows on overflow arcs are assigned a significantly larger cost. All other flows have zero cost. This results in the following objective function similar to that constructed by Hodgson *et al*. [15]:

$$Minimize\left(\sum_{k\in K}\left(\sum_{itjt\in FD} t * W_{kitjt}\right) + \sum_{k\in K}\left((1000) * \left(\sum_{itjt\in OV} W_{kitjt}\right)\right)\right)$$

where *K* is the set of all commodities. *FD* is the set of all arcs, excluding overflow arcs, connected to the sink node; i.e., *FD* is the set of arcs connected to the sink (final destination) within the specified deployment window (*T*). *OV* is the set of all overflow arcs connected to the sink node; the set of arcs which reach the final destination outside of the deployment window. All further discussion is based on the assumption that the deployment window length is of suitable duration for the established network and total movement requirement. The objective function is therefore simplified to the following:

$$Minimize\left(\sum_{k\in K}\left(\sum_{itjt\in FD} t * W_{kitjt}\right)\right) \tag{4.1}$$

Two types of equations primarily define the constraint set. The first is a node balance equation for each node and commodity. For example, consider node *R1* from Figure 4.2, the Ramstein Airbase arrival node. The commodity 2 flows in/out of node *R1* are depicted in Figure 4.3.

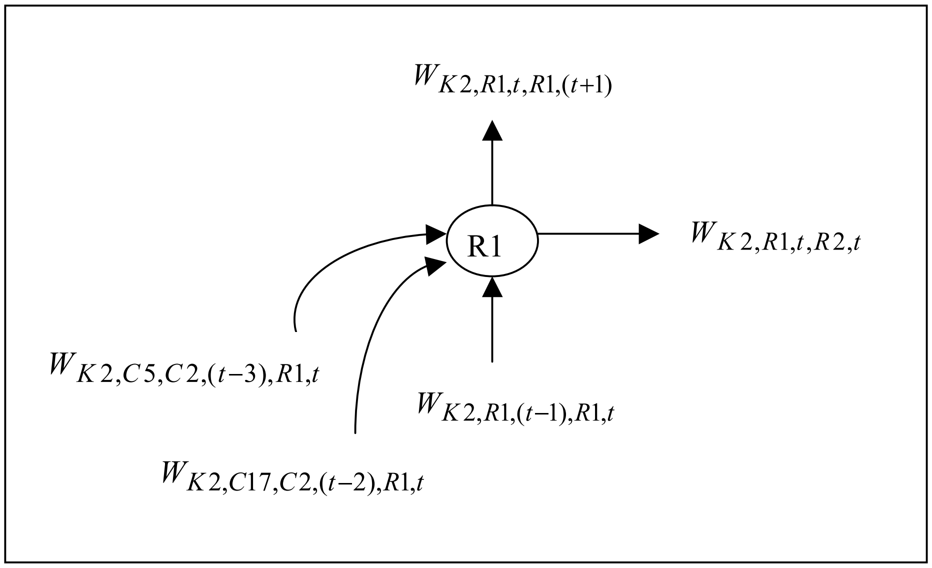


Figure 4.3 Example Node Flows for Individual Commodity

The commodity 2 flows into the arrival node include cargo queued during the previous period, $w_{K2,R1,(t-1),R1,t}$, cargo brought by C5s, $w_{K2,C5,C2,(t-3),R1,t}$, and C17s, $w_{K2,C17,C2,(t-2),R1,t}$. The cargo movement time in this example is three periods for C5s and two periods for C17s. The commodity 2 flows out of the arrival node include cargo that will be put on planes leaving Ramstein for Dhahran, $w_{K2,R1,t,R2,t}$, and cargo that is queued to the subsequent period, $w_{K2,R1,t,R1,(t+1)}$. The commodity 2 balance equation for this node is:

$$\begin{gathered} w_{K2,R1,t,R1,(t+1)} + w_{K2,R1,t,R2,t} - w_{K2,R1,(t-1),R1,t} \\ - w_{K2,C17,C2,(t-2),R1,t} - w_{K2,C5,C2,(t-3),R1,t} = 0. \end{gathered}$$

The second type of equation captures the commodity competition for space on each of the capacitated arcs. These equations are of the form "ΣCommodity Flows ≤ Arc

Capacity". For example, consider the port throughput arc for Ramstein Air Base, arc (R1, R2) from Figure 4.2. Suppose there are a total of three commodities, then the commodity flows through the port are depicted in Figure 4.4.

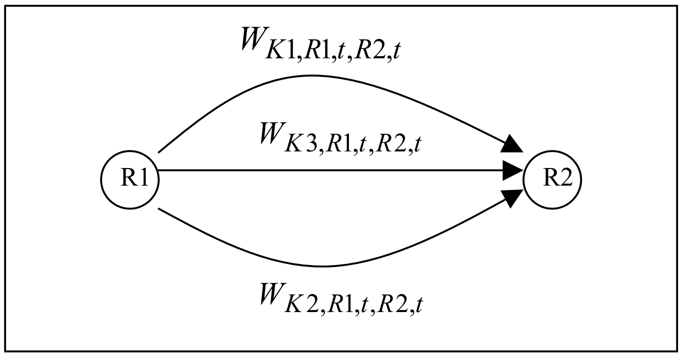


Figure 4.4 Example Commodity Flows Over an Arc

The combined flow of all the commodities through the port in any single period can not exceed the port's period throughput capacity. The resultant port throughput constraint for the Ramstein Air Base with three commodity flows and a period throughput capacity of *Rthru* is:

$$\sum_{k=1}^{3} W_{k,R1,t,R2,t} \le Rthru$$

The general format of each of the node balance equations is developed next. The commodity movement requirements (unit equipment) are commodity flows into an origin port arrival node in the time period that type equipment first becomes available. The commodity node balance constraints corresponding to these requirements are:

$$W_{kiti(t+1)} + W_{kitjt} = CARGO_{kit}, \forall it \in D, \forall k \in K \tag{4.2}$$

where *K* is the set of all commodities and *D* is the set of arrival port nodes in a given time period at which the commodity movement requirements ($CARGO_{kit}$) first become available for movement (*D* represents the set of origin port arrival nodes).

The following equation pertains to all nodes that are not origin port arrival nodes. Node (*i,t*) is the point and time of reference. The general form of the node balance constraint is

$$\sum_{j \in J_i} W_{kitj(t+p)} - \sum_{o \in O_i} W_{ko(t-p)it} = 0, \forall k \in K \text{ and } \forall it \notin D \tag{4.3}$$

where J_i is the set of all destination nodes for *i*, and O_i is the set of origin nodes for *i*. The formulation includes a node balance constraint for each node and each commodity in each time period.

The general format of flows over all arcs is developed next. Each arc has upper bounds on the total amount of flow it can carry in each time period. The general form of the arc capacity limitation constraint is:

$$0 \le \sum_{k \in K} W_{kitj(t+p)} \le U_{itj(t+p)}$$

for each arc in each time period, where $U_{itj(t+p)}$ is the upper bound. All ports have a throughput capacity limitation modeled as

$$\sum_{k \in K} W_{kitjt} \le TP_{ij}, \forall t \text{ and } ij \in R \tag{4.4}$$

where TP_{ij} is the period throughput capacity of the port represented by arc *ij* and *R* is the set of arcs that represent ports. The period throughput capacity can be computed in various manners as described in section 4.2.1. The formulation includes an arc constraint for each port in each period.

All movement arcs have a capacity and are modeled as:

$$\sum_{k \in K} W_{kaitj(t+p)} \leq CAP_{aij} \text{ , } \forall a,t \text{ and } itj(t+p) \in P \tag{4.5}$$

where a is the type of transportation asset, CAP_{aij} is the period weight capacity (in STONs) for the combined assets of type a operating on that arc and P is the set of movement arcs. Each movement arc period capacity is a function of the number of assets operating on that arc and the round trip time of the asset as computed in section 4.3.1. For example, the movement arc period capacity for 6 C5's operating between Campbell and Dhahran would be calculated as:

$$CAP_{C5CD1} = \frac{6*61.3}{\left(2*\left[\frac{7285}{409*1.1507}\right]+4.25+3.25+2*3.25\right)\Big/6} = 49.086 \, \textit{stons/period}$$

This formulation includes an arc constraint for each movement arc in each time period.

To insure that all required flow gets through the model, the following constraint is added:

$$\sum_{itj(t+p) \in FD} W_{kitj(t+p)} = \sum_{it \in D} CARGO_{kit} \text{ , } \forall k \in K \tag{4.6}$$

where FD is the set of all arcs connected to the sink node within the deployment window. There is one constraint for each commodity requirement.

The complete formulation (problem *COMFLOW*), with accompanying equation numbers, is summarized next.

Problem *COMFLOW*

$$Minimize\left(\sum_{k\in K}\left(\sum_{itjt\in FD} t * W_{kitjt}\right)\right) \tag{4.1}$$

Subject to: $W_{kitj(t+1)} + W_{kitjt} = CARGO_{kit}, \forall it \in D, \forall k \in K$ (4.2)

$$\sum_{j\in J_i} W_{kitj(t+p)} - \sum_{o\in O_i} W_{ko(t-p)it} = 0, \forall it \notin D, \forall k \in K \tag{4.3}$$

$$\sum_{k\in K} W_{kitjt} \le TP_{ij}, \forall \text{t and } ij \in R \tag{4.4}$$

$$\sum_{k\in K} W_{kaitj(t+p)} \le CAP_{aij}, \forall a, \text{and } ij \in P \tag{4.5}$$

$$\sum_{itj(t+p)\in FD} W_{kitj(t+p)} = \sum_{it\in D} CARGO_{kit} \quad \forall k \in K \tag{4.6}$$

$$W_{kitj(t+p)} \ge 0, \forall ij \text{ and } t, \forall k \in K \tag{4.7}$$

where

- K is the set of all commodities;
- FD is the set of all arcs, excluding overflow arcs, connected to the sink node;
- D is the set of arrival port nodes, in a given time period, at which the commodity movement requirements ($CARGO_{kit}$) first become available for movement;
- p is the transit time between nodes;
- J_i is the set of destination nodes for node *i*;
- O_i is the set of origin nodes for node *i*;
- TP_{ij} is the period throughput capacity of the port represented by arc *ij*;
- R is the set of arcs that represent ports;

- CAP_{aij} is the period weight capacity (in STONs) of the combined type a transportation assets operating on the assigned route;
- P is the set of movement arcs;
- T is the deployment window of sufficient length.

4.4 Shortfalls and Capabilities

This version of *COMFLOW* puts no emphasis on meeting due dates. Essentially, all commodities have a common weight, and thus all commodity flows are scheduled so as to minimize the total deployment closure time.

In spite of the shortfalls, this model does provide the deployment planner with these capabilities:

- Structure commodity flow according to a simple dispatching rule, all commodities are equal.
- Provide an estimate for each commodity's closure time, C_k.
- Provide a lower bound for deployment closure time, $LB(C_{max})$. The last commodity to close determines the deployment closure time.
- Provide information to compute an estimate for each commodity lateness (L_k) and for $L_{max} = max(L_k)$. $L_k = C_k – d_k$, where d_k is the due date of commodity k.

The information concerning $LB(C_{max})$ and the estimates of C_k, L_k, and $LB(L_{max})$ are gained through post processing *COMFLOW*'s solution.

4.5 Sequencing the Commodity Flows

A simple dispatching rule often used in job shop scenarios is to select the available job with the earliest due date (EDD). To incorporate this idea into *COMFLOW*, different penalties will be assigned to each commodity based on due date. The commodity penalty charge will structure commodity flow to minimize total penalty charge (while still minimizing deployment closure time). If commodity *i* has a higher penalty than commodity *j*, then commodity *i* would have a higher priority of flow than commodity *j*. The result is that the flow of all commodities would be structured according to the EDD policy. The following discussion describes how commodity penalties are assigned.

4.5.1 Priorities and Penalties

Recall that commodity identifiers are based, in part, on due date information. Assign an integer valued priority to each commodity based on its due date. The earlier due date has a higher priority; the highest priority is 1. A tie in due dates is resolved by assigning the commodity with the largest weight requirement the higher priority value. This strategy supports the deployment planner's goal of delivering as much equipment on time as possible.

The commodity penalty is an integer value based on the commodity priority. The largest penalty charge is assigned to the commodity with the highest priority. For example, using the commodity information in Table 4.4, the commodity priorities and penalties assigned are shown in Table 4.5.

Table 4.5 Commodity Priorities and Penalties

Origin	Due Date	Requirement (STONs)	Commodity Identifier	Priority	Penalty
Bragg	3	3	1	1	4
Bragg	4	2	2	3	2
Campbell	5	1	3	4	1
Campbell	4	3	4	2	3

The new objective function including these commodity penalties is:

$$Minimize\left(\sum_{k\in K}\left(\sum_{itjt\in FD}P_k * t * W_{kitjt}\right)\right)$$

where P_k is an integer penalty value for commodity k based on the commodity's due date. The constraint set is not affected and remains unchanged. The commodity penalty and time values operate together to assign a time based penalty charge for each commodity.

4.5.2 Minimizing L_{max}

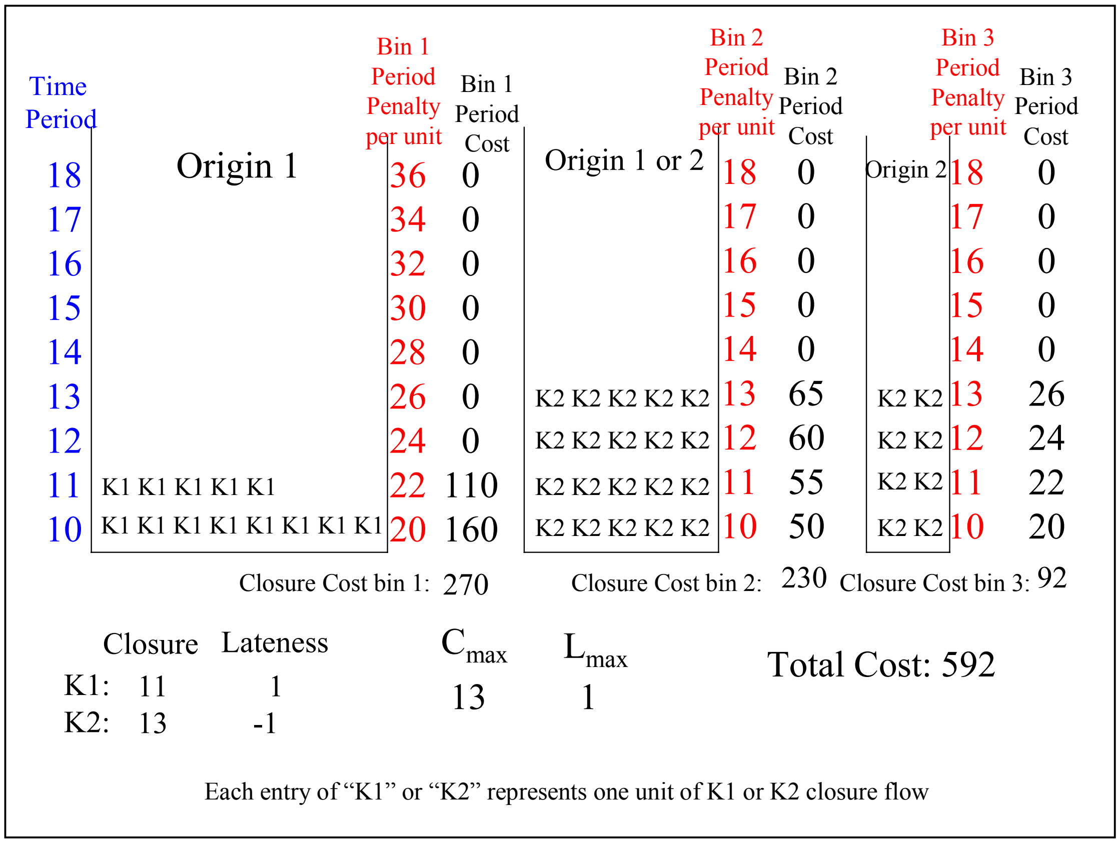
While sequencing jobs according to the EDD rule does minimize L_{max} in a single machine environment, the same is not always true in a multiple machine environment. In the next sections, a comparison between a single/multiple origin deployment network and parallel machines will be made. Finally, in the case of a multiple origin deployment network, a final modification of *COMFLOW* is described. This change creates an iterative process that will usually yield a LB(L_{max}) value within a reasonable amount of time.

4.5.2.1 Environmental Comparisons

Recall from Chapter 3 that the network closure capacity in each period was shown to be equivalent to bins of capacity to be filled. Thus the closure flow scheduling problem is equivalent to a bin filling problem. In a single origin deployment network environment, the total closure capacity in each period is entirely devoted to receiving flows from that origin; a single origin network is a single bin environment. Consider the filling of the single bin as a scheduling of jobs of unit capacity on machines in parallel where all jobs are released at the same time. The EDD rule is known to minimize L_{max} for this problem. Therefore, in a single origin deployment network environment, the sequencing of closure flows according to the EDD rule will also minimize $L_{max.}$

In a multiple origin deployment network, depending on the actual routes within the network, there may be any where from a single bin (access to final port through a common link available to all origins) to at least as many bins as origins. Filling these bins is also equivalent to the scheduling of jobs of unit capacity on parallel machines. The difference is that there are job-machine compatibility restrictions here, i.e. certain jobs cannot be processed on certain machines.

The following two examples (Figures 4.5 and 4.6) show that the schedule of flows with minimum closure time might not have the minimum L_{max}. For both examples, there are two origins; each from which one commodity begins. The first commodity, K1, located at the first origin, has a due date of 10 and a movement requirement of 13 weight units. The second commodity, K2, located at the second origin, has a due date of 14 and a movement requirement of 28 weight units. Flow first becomes available to close in time period 10; these time periods are identified at the far left of each figure. The total closure capacity in each period of 15 weight units is divided among three “bins”. One bin allows access only to flow from the first origin at 8 units per period; another bin allows access only to flow from the second origin at 2 units per period; the final bin allows access to flow from both origins at 5 units per period. These bins are labeled “Origin 1”, “Origin 2” and “Origin 1 or 2” respectively. Since K1 has the earliest due date, it has the larger penalty associated with its period closure times. The period penalty per unit of closure flow and the cost of closure flow for each period are identified to the right of each bin. The total cost of the closure flow assigned to each bin is indicated beneath each bin. The total cost of the closure flow schedule is identified at the bottom right of each figure. Each commodity closure time and associated lateness are identified at the bottom left of each figure. The deployment closure time and maximum lateness (C_{max} and L_{max}) are identified at the bottom center of each figure. Figure 4.5 shows the minimum cost closure flow schedule of these two commodities. This schedule establishes LB(C_{max}) of 13 at a cost of 592; it does not, however, have the minimum L_{max}.



Time Period
Origin 1
Bin 1 Period Penalty per unit
Bin 1 Period Cost
Origin 1 or 2
Bin 2 Period Penalty per unit
Bin 2 Period Cost
Origin 2
Bin 3 Period Penalty per unit
Bin 3 Period Cost
18 36 0 18 0 18 0
17 34 0 17 0 17 0
16 32 0 16 0 16 0
15 30 0 15 0 15 0
14 28 0 14 0 14 0
13 26 0 K2 K2 K2 K2 K2 13 65 K2 K2 13 26
12 24 0 K2 K2 K2 K2 K2 12 60 K2 K2 12 24
11 K1 K1 K1 K1 K1 22 110 K2 K2 K2 K2 K2 11 55 K2 K2 11 22
10 K1 K1 K1 K1 K1 K1 K1 K1 20 160 K2 K2 K2 K2 K2 10 50 K2 K2 10 20
Closure Cost bin 1: 270
Closure Cost bin 2: 230
Closure Cost bin 3: 92
Closure Lateness
K1: 11 1
K2: 13 -1
C_{max} 13
L_{max} 1
Total Cost: 592
Each entry of "K1" or "K2" represents one unit of K1 or K2 closure flow

Figure 4.5 Closure Schedule with Minimum Cost and LB(C_{max})

Figure 4.6 shows a different closure flow schedule. This results in a total cost of 602 and a C_{max} of 14; obviously not a minimum cost nor a minimum closure time schedule. It does, however, have the minimum L_{max} and therefore establishes LB(L_{max}) at 0.

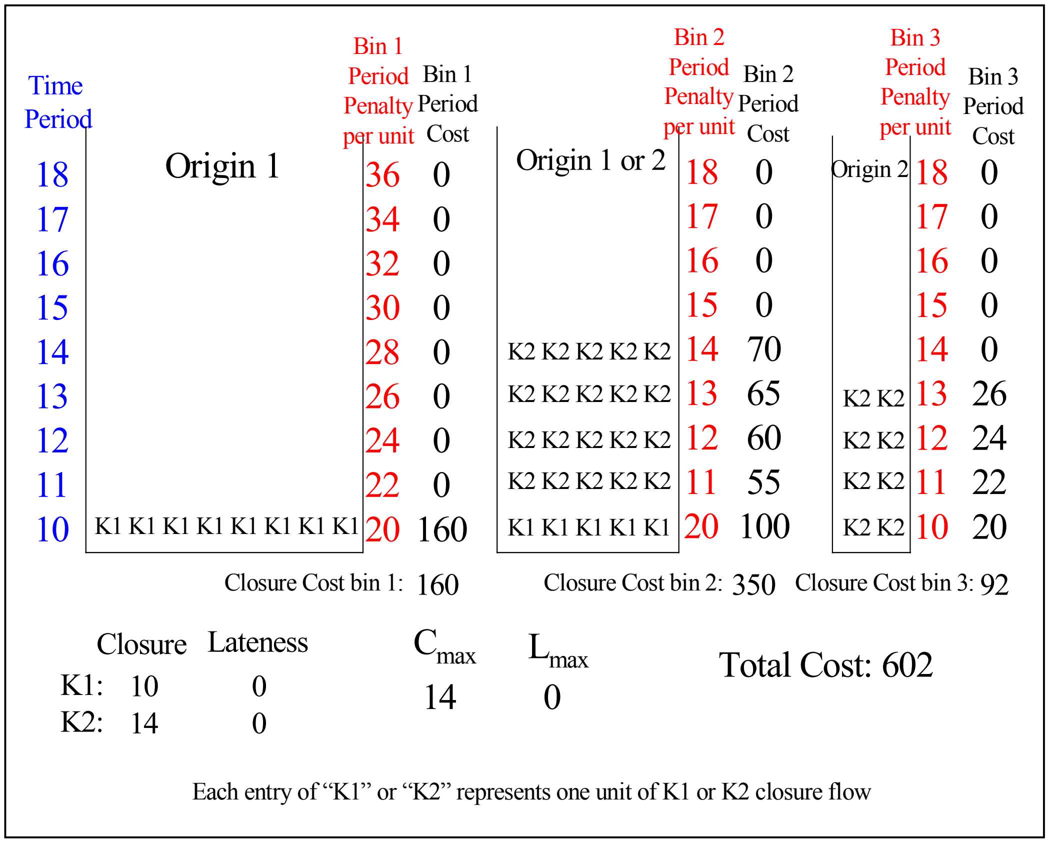


Figure 4.6 Closure Schedule with Minimum L_{max}

Clearly, *COMFLOW* in its current form will not provide an accurate lower bound for L_{max}. It will, however, provide estimates for the lower bound of the lateness of each commodity, as well as the maximum lateness. Some additional modifications of *COMFLOW* are necessary to obtain a lower bound for L_{max}.

4.5.2.2 The Iterative *COMFLOW*

An obvious course of action is to convert *COMFLOW* into a Mixed Integer Problem with the objective of minimizing L_{max}. This could be done relatively easily by the addition of some new constraints. The original set of node balance constraints and arc capacity constraints, equations 4.2 through 4.7 in section 4.3.2 would remain unchanged. For ease of explanation, the case where L_{max} is known to be greater than or equal to zero will be discussed before the general case is presented. The minimum cost formulation of the objective function would be replaced by the objective function: Min L_{max}. The following integer constraints would complete the transformation:

$$W_{kitjt} \leq CapMax * X_{kitjt}, \ \forall k, \forall itjt \in FD \tag{4.8}$$

$$(t - d_k) * X_{kitjt} \leq L_{\max}, \forall k, \forall itjt \in FD \tag{4.9}$$

$$X_{kitjt} \ 0 \text{ or } 1, \forall k, \forall itjt \in FD \tag{4.10}$$

$$0 \leq L_{\max} \tag{4.11}$$

where *CapMax* is a sufficiently large constant (larger than the largest period closure capacity), *t* is the time that arc *itjt* occurs (the time that closure occurs) and d_k is the due date of commodity *k* (then $t - d_k$ is the lateness of commodity *k* associated with closure flow at time *t*). For the general case where it is not known if L_{max} is nonnegative, the problem must be scaled so that all latenesses are nonnegative. This can be achieved by subtracting a large constant, *M*, from all the due dates. Therefore the maximum lateness of the problem is $L_{max} - M$. Unfortunately, as with most integer programming problems, the solution time for large-scale problems is prohibitive, requiring hours, days or longer to obtain the optimal solution. Since deployment planners do not have the luxury of unlimited time, another technique must be developed.

After post processing the solution, *COMFLOW* will provide the deployment planner valuable information. *COMFLOW*'s solution provides a feasible schedule of flows with minimum deployment closure time. Intermediate due dates are attempted to be satisfied through the sequencing of the commodity flows in EDD order. Analysis of the solution will allow the planner to establish the closure time of each commodity (C_k). From this information, the individual commodity lateness (L_k) is computed as $L_k = C_k - d_k$, where d_k is the commodity due date. Finally, $L_{max} = \max(L_k)$. This value is a feasible estimate for $LB(L_{max})$.

Suppose that all commodity closure flows were limited to closure arcs occurring at times such that the difference between each commodity closure flow time and due date was less than the established L_{max} value. If this model were solved and a feasible solution found, then a new closure schedule exists with a smaller maximum lateness value. This idea forms the basis of an iterative approach to determine the $LB(L_{max})$; this approach was adapted from a procedure proposed by Horn [18]. The flow restriction constraints added to *COMFLOW* are:

$$W_{kitjt} = 0, \quad \forall k, \forall itjt \mid t - d_k \geq V \tag{4.12}$$

where V is the current best estimate of L_{max}. These equations force all commodity k closure flows onto closure arcs with times that establish a commodity closure lateness less than V. The proposed iterative technique is described next:

```
Step 1.  Solve COMFLOW.  Analyze solution to determine V.

Step 2.  Solve COMFLOW, with restriction equations (4.12)

    If no feasible solution exists, Stop.
    Else set $LB(L_{max}) = V$.  Go to Step 3.

Step 3.  Set $V = V – 1$ and go to Step 2.
```

Although multiple applications of *COMFLOW* are required, *COMFLOW* remains a linear program with the addition of the constraint set in (4.12). Thus its solution time requires a matter of minutes instead of hours or days. The sequencing of flows according to the EDD rule should provide a good initial estimate of the true lower bound for L_{max}, which should limit the number of applications of *COMFLOW* required. This iterative technique will determine a lower bound for the maximum lateness. Note that given $LB(L_{max})$ is established, *COMFLOW*'s established deployment closure time associated with that $LB(L_{max})$ is a conditional $LB(C_{max})$. The iterative application of *COMFLOW* establishes a $LB(C_{max})$ given that $LB(L_{max})$ is minimized.

Chapter 5

The Deployment Scheduling Analysis Tool (DSAT)

A new software tool (still in development) called the Deployment Scheduling Analysis Tool (DSAT), allows a user to rapidly perform accurate military deployment scheduling and sensitivity analysis on a personal computer [17]. The tool has three main components: a graphic user interface (GUI), a program to efficiently allocate a limited set of transportation resources to routes, and a program to schedule the equipment onto the transportation assets. Both the GUI and the asset allocation program are written in Visual Basic. Through the GUI, the planner defines the movement requirement and the deployment network. To define the movement requirement, the planner first selects the units comprising the deployment package and then identifies the required delivery dates of the selected units. Selecting the deploying units also selects the equipment and identifies the units' usual ports of departure (aerial and sea); this default port selection may be changed. The planner then selects any intermediate nodes (transshipment nodes) required in addition to the final destination nodes. Next, the planner allocates transportation assets to each leg of the network. The asset to route assignment completes the deployment network. Once the deployment network has been defined, DSAT creates an input text file for the scheduler routine; this routine is included as an executable file. The scheduler routine is Thoney's [33] modification of the Virtual Factory simulator for application to the military deployment problem. It is written in C++. The routine schedules the individual pieces of equipment onto the transportation assets, simulates the deployment process using that equipment schedule and determines critical information concerning the resultant deployment schedule. Once the scheduler routine is completed, DSAT processes the information into explanatory graphs and reports for the deployment planner. The planner then has several options available in conducting sensitivity analysis. This chapter describes the scheduler routine in detail, including the different heuristics it uses. Alternate heuristics are developed in an attempt to improve the overall performance of DSAT.

5.1 Scheduler Routine Terminology

Thoney developed the military deployment scheduling procedure as a specialized extension of the Virtual Factory. The focus of her work was job shop scheduling with batch processors. In terms of the Virtual Factory, each piece of unit equipment in the deployment package represents one *real job* that must be scheduled for movement. *Fake jobs* are created by the scheduler routine as “space fillers” for the various transportation assets. The various ports in the deployment network are the *factories* through which the jobs process; the transportation assets are the *batch processors* in which a job moves between factories.

Some additional specialized terminology is needed to complete the translation of the deployment scheduling problem into a job shop scheduling problem for the Virtual Factory. Figure 5.1 displays the deployment scenario of Chapter 4 with some new labels.

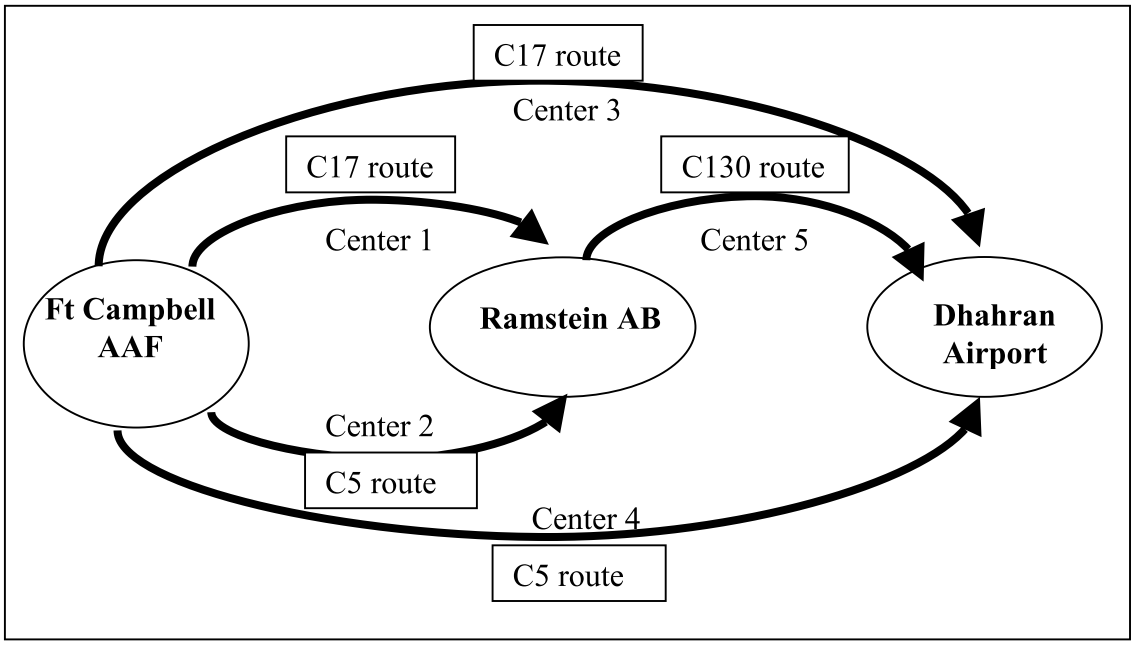


Figure 5.1 Center-Route Representation of Deployment Scenario

Each arc represents a distinct route between two ports flown by a particular type asset. A unique *center* number identifies the asset type and the assigned route. Each center has a number of assets assigned; a *vehicle* is one asset in a center. A sequence of routes (or centers) from a base to the final destination is a *path* on which a job may move. As shown in the figure, there are two routes to the Ramstein Airbase; these routes are identified as centers 1 and 2. If there were ten C17s assigned to fly from Campbell AAF to Ramstein AB, then center 1 would have ten vehicles. One path from Campbell to Dhahran uses centers 2 and 5; another is a direct route using only center 3. A port or number of ports in the same general location is a *base*. For example, Fort Campbell is a base with two ports, one airfield and one rail depot (not shown). Ramstein is a different base with a single port, the airbase.

5.2 Scheduler Routine Heuristics

The scheduler routine uses a plan that incorporates two unique heuristics. These heuristics allocate the jobs to the paths in an attempt to minimize the maximum lateness of all jobs. All jobs are pre-sorted in increasing due date order. To minimize maximum lateness, the job to path allocation attempts to schedule the jobs' arrival at the final destination in due date order. The plan performs one iteration of an initial allocation heuristic and then ten iterations of a reallocation heuristic. The initial allocation heuristic constructs the path that each job will travel over based on the average travel time to complete the path. The jobs are initially assigned to the path that is anticipated to be the fastest. The reallocation heuristic makes use of information gained through simulating the previous allocation. In the simulation, as vehicles arrive and are unloaded at the base(s), jobs are completed and arrival times are recorded. Jobs usually do not arrive in exact due date order. Therefore, based on this information, the jobs are reallocated to paths according to actual arrival at the final destination. This reallocation heuristic is performed ten times and the best results are saved. These heuristics and two proposed alternate heuristics are discussed in detail in the following sections.

5.2.1 Initial Allocation Heuristic

The initial allocation heuristic builds each job's path based on the structure of the deployment network. The jobs are initially assigned to a path that is anticipated to be the fastest. To build the paths, multiple passes through the network are required. During the first pass, a backward traverse of the network, the average "travel" time from each base to the final destination is computed. During the next passes of the network, forward traverses, each job at the current base, base *i*, is allocated to a vehicle that has the earliest anticipated final destination arrival time and can accommodate the job. A simplistic example is provided below, prior to the detailed algorithm and required definitions. This example uses the deployment network of Figure 5.1 and additional center information. The nodes of the network represent bases and are identified by the letters *C*, *R* and *D* for Campbell, Ramstein and Dhahran, respectively. Node *C* is the origin node; node *D* is the final destination node. The number of vehicles assigned to each center and the job time to the center's destination are identified in Table 5.1.

Table 5.1 Center Information for Deployment Network

Center *i*	**Asset Type**	**# Vehicles**	**Job Time, J_i**
1	C17	2	5
2	C5	3	6
3	C17	3	8
4	C5	2	9
5	C130	5	4

The center's job time represents the amount of time jobs spend on the center to travel to the next destination. The job time is the amount of time needed to load and unload the center in addition to the actual one way travel time needed by the center to reach that next destination.

During the first pass through the network, the backward traverse, the average remaining times (R_i) from node *i* to the final destination base are computed. R_i is based on all paths possible from node *i*. The "travel" time for a path from a base to the final destination consists of a center job time to the next destination plus the average remaining time from that base. The sum of each path's "travel" time multiplied by the number of vehicles in the initial center is divided by the total number of vehicles operating at that base. In the deployment scenario, there is one path from Ramstein and four paths from Campbell. The following average remaining times are computed in Table 5.2:

Table 5.2 Average Remaining Travel Times to Final Destination

Node *i*	**R*i***
D	0
R	$\frac{5(4)}{5}=4$
C	$\frac{3(8)+2(5+R_R)+3(6+R_R)+2(9)}{10}=9$

During the remaining passes, forward traverses, jobs are allocated to the center vehicles. At each base, the jobs are sorted in due date order. The jobs are allocated based on which path is anticipated to reach the final destination the earliest among the vehicles in which the job can fit. Once the job reaches the next destination in the path, it is inserted into the list of jobs at that base. Job scheduling at base *i* can not begin until all jobs at bases preceding base *i* have been scheduled. Then all jobs scheduled to move through base *i* are on base *i*'s job list, and job allocation at that base can now begin. The anticipated final destination arrival time of jobs allocated on a vehicle is the vehicle's job time plus the average remaining time to reach the final destination base from that location. The earliest that subsequent trips for each vehicle can start is after the vehicle returns to it's origin base. So, at node C, the jobs scheduled on the first trip of a center 1 vehicle are anticipated to arrive at the final destination at time 5 + R_R; the jobs scheduled on the second trip of that vehicle are anticipated to arrive at the final destination at time 15 + R_R, and so on.

The following definitions are necessary for both the initial and the reallocation heuristics:

N	number of jobs
U_i	area of job i
W_i	weight of job i
V_i	volume of job i
Q_i	current path (in terms of centers) of job i
B_{il}	Boolean value indicating if job i can be placed on center l
M	number of centers
m_j	number of vehicles within center j
TU_j	total area capacity of each vehicle in center j if j is a ship, otherwise $TU_j = -1$
TW_j	total weight capacity of each vehicle in center j
TV_j	total volume capacity of each vehicle in center j if j is a not a ship, otherwise $TV_j = -1$
K_j	vehicle type of center j
J_j	time job spends on a vehicle from center j (load time + one way travel time + unload time)
H_j	time to complete return trip of vehicle from center j (maintenance time + one way travel time + maintenance time)
T	trip number of a vehicle
R_b	average remaining time from base b to the final destination base
A_{jm}	the anticipated final destination arrival time if a job is scheduled on the mth trip of a vehicle in center j
D_j	destination base of center j
$C(b)$	set of centers at base b
$L(b)$	set of jobs currently located at base b
$F(b)$	set of fake jobs currently located at base b

Note the conditional definitions of TU_j and TV_j. The scheduler routine maintains information concerning the weight and area capacities for all ships throughout the simulation. The scheduler also maintains information concerning the weight and volume capacities for all other types of transportation assets.

The first trip of a vehicle in center j is anticipated to arrive at the final destination at time $A_{j1} = J_j + R_{Dj}$. The second trip of a vehicle in center j is anticipated to arrive at the final destination at time $A_{j2} = J_j + H_j + J_j + R_{Dj}$. Consequently, the Tth trip of a vehicle in center j is expected to arrive at time $A_{jT} = T(J_j) + (T\text{-}1)H_j + R_{Dj}$. The average remaining time from base b to the final destination base is 0 if b is the final destination base, otherwise it is calculated as: $R_b = \dfrac{\sum_{x \in C(b)} m_x \left(J_x + R_{D_x}\right)}{\sum_{x \in C(b)} m_x}$

The algorithm for the initial allocation heuristic has three parts; each part is carried out for every base. The first part involves creating a list of vehicle nodes sorted in increasing order of anticipated arrival time at the final destination. There is one vehicle node list created for each base. Let node (s, t, u, v, w) be a vehicle node where s is the anticipated arrival time at the final destination, t is the center number, u is the available area capacity, v is the available volume capacity, and w is the available weight capacity. For each vehicle, one node is placed on the list representing its first arrival at the final destination. Assume that the jobs are sorted in order of increasing due dates. The first part of the heuristic (the backward traverse) is detailed as follows:

```
For each base $b$:
    For each $j \in C(b)$:   (for each center operating at base $b$)
        Step 1.  Calculate $A_{j1}$
```

```
            Step 2.  Set vehicle $k$ within center $j$ equal to 1
            Step 3.  Insert node ($A_{j1}$, $j$, $TU_j$, $TV_j$,$TW_j$) on the vehicle node list
            Step 4.  if $k < m_j$, set $k$ to $k + 1$, go to Step 3
        Next center $j$
    Next base $b$
```

The completion of this part of the initial allocation heuristic results in a vehicle node list at each base sorted in increasing order of anticipated arrival times at the final destination. A node for each vehicle in each center is placed on this list. Each entry details the cargo's first anticipated arrival time at the final destination, the center number and the vehicle's available capacities. This provides a structure for initially assigning jobs to vehicles based on the job due dates.

The second part of the initial allocation heuristic (a forward pass) allocates the real jobs to the vehicles and adds subsequent vehicle trips on the list as necessary. Each real job is placed at its origin base. This part's algorithm is as follows:

```
    For each base $b$:  (starting at origin base)
        For each $i \in \{L(b) - F(b)\}$:   (for each real job at base $b$)
            Step 1.  Scan vehicle node list from beginning; find the first
            node ($s$, $t$, $u$, $v$, $w$) simultaneously satisfying the following
            conditions:

                $U_i \le u$ (if $u \ne -1$)  (job area fits into remaining vehicle
                                        capacity)
                $V_i \le v$ (if $v \ne -1$)   (job volume fits into remaining vehicle
                                        capacity)
                $W_i \le w$  (job weight fits into remaining vehicle capacity)
                $B_{it}$ = True  (job $i$ fits on center $t$)
            If no such node exits, go to the next real job.

            Step 2.  If $u = TU_j$ and $v = TV_j$ and $w = TW_j$, insert
            node $(s + J_t + H_t, t, TU_t, TV_t, TW_t)$ on list (create next trip)

            Step 3.  For the node found in Step 1, let $u = u - U_i$ (if $u \neq -1$),      $v$
            $= v - V_i$ (if $v \neq -1$) and $w = w - W_i$ . (This decrements the
            remaining vehicle weight and area or volume capacities.)

            Step 4.  Let $Q_i = Q_i + \{t\}$    (augment current path with center $t$)

            Step 5.  Set $L(D_t) = L(D_t) + \{i\}$    (insert job on job list at next
            destination)

        Next $i$

    Next $b$
```

This part of the initial allocation heuristic assigns each job to move on a specific center vehicle. If the job fits into the remaining volume and weight capacities of the vehicle and is compatible with the center, then that job is allocated to that vehicle. If the vehicle is empty, a subsequent trip for that vehicle is added to the vehicle node list (Step 2), while the remaining capacities of the vehicle for its current trip are decremented (Step 3). The path of the job is updated to indicate job travel on that center (Step 4). Finally, the job is added to the job list at the center's destination base (Step 5). Any job that does not fit into any vehicle originating at that base is essentially removed from that base's job list. They are assigned a path at the end of the initial allocation heuristic. This results in each job having a specific path built for it. The path is expressed as the sequence of centers the job travels on to get from its origin to its final destination.

In creating the initial job to path allocation, the initial allocation heuristic could over utilize (use to the exclusion of all or some other paths) or under utilize a particular path. This situation, in turn, reduces the options available to the reallocation procedure that shifts jobs to alternate paths. To alleviate this problem, *fake jobs* are created for each path. These fake jobs have very low priority (i.e. their due dates are much later than real job due dates). Allocating these fake jobs to paths creates additional path arrival times at the final destination. Thus, the reallocation procedure has more options available for all paths in considering job rerouting. Consequently, the third part of the initial allocation heuristic creates the fake jobs and their paths. Once again, the bases are treated individually, traversing in a forward manner. One fake job, i, is initially placed at each origin base with $U_i = V_i = W_i = \infty$. The procedure is as follows:

```
for each base $b$:
    for each $i \in F(b)$: (for each fake job)
        Step 1. Let $temp = 0$
        for each $j \in C(b)$: (for each center)
            Step 2. set $temp = temp + 1$
            If $temp \neq |C(b)|$ (if $temp \neq$ number of centers at base)
                Step 3. Create a fake job, $d$
                Step 4. Let $Q_d = Q_i + \{j\}$
                Step 5. If $TU_j \neq -1$, let $U_d = \min\{U_i, TU_j\}$,
                If $TV_j \neq -1$, let $V_d = \min\{V_i, TV_j\}$,
                Let $W_d = \min\{W_i, TW_j\}$
                Step 6. Let $L(D_j) = L(D_j) + \{d\}$
            Else
                Step 7. Let $Q_i = Q_i + \{j\}$
                Step 8. If $TU_j \neq -1$, let $U_i = \min\{U_i, TU_j\}$,
                If $TV_j \neq -1$, let $V_i = \min\{V_i, TV_j\}$,
                Let $W_i = \min\{W_i, TW_j\}$

                Step 9.  Let $L(D_j) = L(D_j) + \{i\}$

            End if

        Next $j$

    Next $i$

Next $b$
```

When this procedure is complete, each path has one fake job assigned to it that has the maximum possible area, weight and volume a job can have if it is allocated to that path. The assignment of fake jobs to paths is done in both Steps 3 through 6 and Steps 7 through 9. Steps 3 through 6 clone each fake job that is currently at the base. The number of clones for a given fake job currently at the base is equal to the number of centers at that base minus one. The cloned jobs are assigned to the same path taken by the current fake job in arriving at the current base. Then each of the cloned jobs is designated for transport by a unique center at the current base. Steps 7 through 9 assign the job that was cloned for transport by the remaining center.

After this procedure is complete, the largest possible area, volume and weight that can fit in all assets on all of the paths is found. Each fake job is then cloned 50 times and the clones' area, volume and weight are set equal to this smaller value. The purpose for having 50 smaller jobs and one large job on each path is to help ensure that at least one extra trip is created for each vehicle on the path and that capacity on partially filled final trips of real jobs is completely filled with fake jobs. The purpose of this is to produce additional arrival times at the destination for each path that provides the reallocation procedure more options in rescheduling jobs. In forming the reallocation list, blocks are created from aggregating job capacity regardless if a job is real or fake. In contrast, only real jobs are rescheduled in the blocks because the fake jobs have a fixed path.

The final part of this heuristic creates paths for real jobs that have not yet been allocated. Each remaining real job is compared with the fake jobs (each fake job has a fixed path) until a compatible path is found. The job is then allocated to that path.

5.2.2 The Reallocation Heuristic

Once the jobs are scheduled by an allocation heuristic, that job allocation is simulated by the scheduler routine. Information is gathered on the actual arrival times of each job to the final destination. The jobs do not necessarily arrive in due date order. Each time a job arrives at the final destination, its area, volume and weight is added to the aggregate area, volume and weight of jobs that already arrived by way of the same path at the same time. The arrival time information is thus used to create *blocks of used capacity* that arrive at the final destination at given times. The reallocation heuristic reschedules the jobs (sorted in order of increasing due dates) to these blocks of used capacity sorted in increasing order of final destination arrival time.

This allocation heuristic has two parts. Each part is performed once. The first part involves developing a list of capacity blocks during the simulation sorted in increasing order of final destination arrival times. The second part of the heuristic entails the actual reallocation of the jobs to the vehicles. The reallocation heuristic uses many of the terms defined for the initial allocation heuristic, but several additional terms are defined next. Let block (s, t, u, v, w) be a “used capacity” block where s is the final destination arrival time, t is the tth arrival at time s, u is the current used area capacity, v is the current used volume capacity and w is the current used weight capacity for the block. The following additional definitions are also necessary:

P_{qr}	current path (in terms of centers) of the rth block arriving at time q
$E(s)$	set of jobs arriving at the final destination at time s
NB_s	total number of blocks arrived at time s

The first part of the heuristic records the final destination arrival times and aggregates certain used capacities into blocks of used capacities. It is detailed as follows:

```
Step 1.  Let $x$ be the first time in which at least one job arrives at the final
destination.

Step 2.  Set $NB_x$ equal to 1.  (initialize total number of blocks arriving at time $x$)

Step 3.  For each $i \in E(x)$:  (for each job that arrives at the final destination at time
$x$)

        a.  Search list of all blocks ($s$, $t$, $u$, $v$, $w$), where $s = x$ to see if $P_{st} = Q_i$ for
        any $t$. (find a block arriving at time $x$ with the same path as job $i$)

        b.  If a block exists at some $t$ (number of arrival, not time), let $u = u + U_i$
        (if $u \neq -1$), let $v = v + V_i$ ( if $v \neq -1$) and let $w = w + W_i$
        Else increment $NB_x$ by 1 and add block ($x$, $NB_x$, $U_i$, $V_i$, $W_i$) to the list.

next $i$

Step 4.  If all jobs are not at the final destination, let $x$ be the next time at least one
job arrives at the final destination and go to Step 2.
```

The result of this first part of the reallocation heuristic is a list of used capacity blocks sorted in increasing arrival time at the final destination. Each used capacity block is then inflated by 20%; i.e. the used area, volume and weight are increased by 20%. The purpose of this inflation is to reduce the amount of unused capacity on the blocks' path in preparation for the eventual job rescheduling.

The second part of the algorithm performs the rescheduling of the real jobs (sorted in increasing due date order) onto the identified used capacity blocks. If a real job cannot be rescheduled it remains on its current path. The fake jobs always keep their initial path. The second part of the heuristic is as follows:

Step 1. Set the job, i, equal to 1. (current job to be scheduled)

Step 2. Search the list of blocks to find the first block (s, t, u, v, w) that simultaneously satisfies the following conditions:

> block has same origin base as job i
> $U_i \leq u$ (if $u \neq -1$) (job fits into remaining used area capacity)
> $V_i \leq v$ (if $v \neq -1$) (job fits into remaining used volume capacity)
> $W_i \leq w$ (job fits into remaining used weight capacity)
> $B_{ik} = \text{True} \; \forall \; k$ in P_{st} (job fits onto all centers in path)

Step 3. If no such block exists, go to Step 6, otherwise, go to Step 4.

Step 4. If $u \neq -1$, let $u = u - U_i$. If $v \neq -1$, let $v = v - V_i$. Let $w = w - W_i$ (decrement remaining block capacities)

Step 5. Let $Q_i = P_{st}$ (redesignate path for job i)

Step 6. If $i < N$, increment i by 1 and go to Step 2.

5.2.3 Proposed Alternate Allocation Heuristics

The original heuristics are lengthy and complicated. The path construction for the jobs requires one complete traverse of the deployment network and several computations even before the first job's path can be constructed. Different schemes might also construct a good path while using less computer run time to execute. These alternate allocation heuristics, a second initial allocation heuristic and a second reallocation heuristic, are proposed and described in detail in the following sections.

5.2.3.1 Initial Allocation Heuristic 2

For many items of equipment, the overriding characteristic that determines the mode of travel (and hence the path) is its weight and/or volume. Regardless of when it is due, the job might have to go by a slower vehicle because it is too large or heavy to fit on a faster vehicle. The original initial allocation heuristic computed the average travel time to reach the final destination from the current location. At each base, all centers were included in this computation. It disregarded vehicle/job compatibility. Therefore, a seemingly faster path might have been based on travel times of incompatible assets. This second constructive heuristic only considers centers that are compatible with the job. It does not compute the anticipated average remaining travel time to the final destination. Instead, it builds a job's path based on the job's fastest anticipated arrival time at the *next* destination on assets that can accommodate it. Unlike the original, this heuristic does not require a backward traverse of the network. Instead, it performs only forward passes.

Similar to the original initial allocation heuristic, this second heuristic uses the same definitions. The only "new" definition is a redefinition of A_{jm}. Under this heuristic, A_{jm} is the anticipated next destination arrival time if a job is scheduled on the mth trip of a vehicle in center j. This information is passed into the scheduler routine, so no calculation is actually performed. This second heuristic creates a list of vehicle nodes sorted in increasing order of anticipated arrival time at the next destination. Let node (s, t, u, v, w) be a vehicle node with s the anticipated arrival time at the next destination, t the center number, u the current remaining area capacity, v is the current remaining volume capacity and w is the current remaining weight capacity of the vehicle. Fake jobs are treated in exactly the same manner as in the original initial allocation heuristic. The first part of this heuristic creates the initial vehicle nodes for every vehicle in every center. Then, all real jobs are allocated to the vehicle nodes and subsequent vehicle trips are created as necessary. The actual algorithm is very similar to that previously shown:

```
For each base $b$:
    For each $j \in C(b)$:  (for each center operating at base $b$)
        Step 1.  Calculate $A_{j1}$
        Step 2.  set vehicle $k$ within center $j$ equal to 1
        Step 3.  Insert node ($A_{j1}$, $j$, $TU_j$, $TV_j$,$TW_j$) on the vehicle node list
        Step 4.  if $k < m_j$, set $k$ to $k + 1$, go to Step 3
    Next center $j$  (creates list of 1st trip vehicle nodes for all vehicles)
    For each $i \in \{L(b) - F(b)\}$  (for each real job at base $b$)
        Step 5.  Scan vehicle node list from beginning; find the first
        node ($s$, $t$, $u$, $v$, $w$) that simultaneously satisfies the following
        conditions:
            $U_i \leq u$  (if $U_i \neq -1$)  (job area fits into remaining vehicle
                                        capacity)
            $V_i \leq v$  (if $V_i \neq -1$)  (job volume fits into remaining vehicle
                                        capacity)
            $W_i \leq w$  (job weight fits into remaining vehicle capacity)
            $B_{it}$ = True  (job $i$ fits on center $t$)
```

```
                If no such node exists, go to next real job.

                Step 6.  If $u = TU_j$ and $v = TV_j$ and $w = TW_j$,
                insert node $(s + J_t + H_t, t, TU_t, TV_t, TW_t)$ on list (create next trip)

                Step 7.  For the node found in Step 1, If $u \neq -1$, let $u = u - U_i$.
                If $v \neq -1$, let $v = v - V_i$.  Let $w = w - W_i$.  (This decrements the
                remaining vehicle weight and volume capacities.)

                Step 8.  Let $Q_i = Q_i + \{t\}$    (augment current path with center $t$)

                Step 9.  Set $L(D_t) = L(D_t) + \{i\}$    (insert job on job list at next
                destination)

        Next $i$

    Next base $b$
```

The second part is identical to the third part of the original initial allocation heuristic. This part creates the fake jobs and their fixed routes. It ensures that each path has one large and 50 small fake jobs assigned to it. As in the original initial allocation heuristic, any remaining real jobs not yet allocated to a path are given a compatible path at this time.

5.2.3.2 Reallocation Heuristic 2

This reallocation heuristic also uses information learned through the simulation of the previous allocation. This second heuristic is based on identifying the critical jobs and what their paths were. A critical job has a job lateness equivalent to the maximum lateness. Once the critical jobs are identified, matching jobs with smaller job lateness values are found. A matching job has the same origin base, weight, volume and area as the critical job. This heuristic simply switches the paths of the critical and matching jobs in an attempt to reduce the maximum lateness. The following additional definitions are needed:

L_i = lateness of job i
$L_{\max}$ = maximum lateness of all the jobs
CP = critical path; a path traveled by a critical job ($L_i = L_{\max}$)
O_i = Origin base of job i
JL = list of all real jobs
MAX = maximum number of reallocation iterations to be performed
reallocationiters = number of reallocation iterations already performed

The algorithm is as follows:

```
For each $i \in JL$:

    Step 1. Check job.
    If $L_i$ = L$_{\max}$ go to step 2.

        Step 2. Set $CP$ equal to $Q_i$

        Step 3. Locate a matching job.
        Scan $JL$ from beginning. Find first job $j$ that simultaneously
        satisfies the following conditions:

            $O_j = O_i$
            $V_j = V_i$
            $W_j = W_i$
            $U_j = U_i$
            $L_j < L_i$

        Step 4. If matching job found set $Q_i = Q_j$ and $Q_j = CP$
        Else set $reallocationiters = MAX$
    Else
Next $i$
```

In Step 4, if no matching job is found then no alteration in job paths will improve the current schedule. In this case, the iteration counter value is set to MAX. This will terminate remaining reallocation iterations since no improvement is possible.

5.3 New Scheduler Routine Plans

The addition of these two alternate heuristics provides the ability to create four different basic scheduler routine plans for the job allocation problem. The scheduler routine originally had one plan consisting of one iteration of the original initial allocation heuristic followed with ten iterations of the original reallocation heuristic. Denote this plan as Plan 1. The additional new plans will also consist of a single iteration of an initial allocation heuristic followed with ten iterations of a reallocation heuristic. They differ in the pairing of the two heuristics. Plan 2 consists of the original initial heuristic and the alternate reallocation heuristic. Plan 3 consists of the alternate initial heuristic and the original reallocation heuristic. Plan 4 consists of the alternate initial heuristic and the alternate reallocation heuristic. These four plans are summarized in Table 5.3. The original heuristics (initial and reallocation) are labeled with 1 while the alternate heuristics (initial and reallocation) are labeled with 2.

Table 5.3 New Scheduler Routine Plans

Plan Identifier	Initial Heuristic	Reallocation Heuristic
1	1	1
2	1	2
3	2	1
4	2	2

5.4 Critical Aspects of Scheduler Plans

Regardless of which combination of initial and reallocation heuristics are used to create the four different scheduler plans, all plans have at least two common aspects. The first aspect concerns the scheduler's rules used in loading the vehicles. The second aspect concerns the number of reallocation iterations used in the scheduling plans. These aspects could greatly affect the performance of the scheduler routine and are now presented in greater detail.

5.4.1 Vehicle Loading Rules

Recall that the scheduler routine allocates jobs to paths according to the previously defined heuristics. The path is defined as the sequence of centers the job uses to get to its final destination. During simulation, the Virtual Factory maintains multiple job lists at each base in the network; there is one job list for each center that operates at that base. Each job list is maintained in increasing due date order. For each vehicle being loaded, the Virtual Factory works from the top of the appropriate center job list. Once the job is loaded onto the vehicle, it is removed from the center list. If a job does not fit into the remaining available vehicle capacity, it is counted as a “failure” and the next job is considered for loading. The original vehicle loading rule loads jobs onto a vehicle until five “failures” occur, then the loading operation ceases. The vehicle is then released for movement to the center’s next destination. Under this particular loading rule, a non-full vehicle could depart a base even if some remaining jobs could fit. The following proposed loading rules affect how extensively each center job list is searched while conducting the vehicle loading operation.

The first new loading rule is simply to load each vehicle until 75 “failures” are counted. Although the entire center job list is not searched, this does greatly extend the length of the search for a fitting job. The second new loading rule is goal oriented.

Under this rule, the entire center job list is searched unless the current vehicle load meets a predefined goal. If the goal has been met, the loading operation ceases and the vehicle is released for movement even if remaining jobs could fit. This goal is defined based on the type of vehicle. An aircraft usually is "weighted out" before being "cubed out". This means that typically, an aircraft cargo load fills the aircraft's weight capacity leaving remaining available space. A ship usually runs out of available floor space well before filling it's weight capacity. The vehicle loading goal is met if, for an aircraft or train, 90% of its weight or volume capacity is filled. The goal for a ship is 90% utilization of its area or weight capacity. A third new loading rule has a goal of 95% vehicle utilization. The last new loading rule is simply to always search the entire center job list when loading a vehicle. The loading rules are called: Fail5 (the original rule), Fail75, Goal90, Goal95 and WholeList.

5.4.2 Number of Reallocation Iterations Performed

The original structure of all the scheduler plans includes one iteration of an initial allocation heuristic followed by 10 reallocation heuristic iterations. Instead of performing a fixed number of reallocation iterations, a variable number will be used. The first variation will perform reallocation iterations, always recording the best results, until 20 consecutive iterations without improvement are performed. The remaining variations differ only by the number of consecutive no-improvement iterations performed. The other methods stop after 30, 40 and 50 consecutive no-improvement iterations. These variations are called 10Iters (the original), 20Consec, 30Consec, 40Consec and 50Consec.

5.5 Improving DSAT Performance

The DSAT tool was designed to meet the deployment planners' need for a fast and effective scheduling and analysis tool. The performance of the scheduler routine has a direct impact on the overall performance of DSAT. The scheduler performance is defined by its results (in terms of deployment closure time and maximum lateness) and the amount of time needed to obtain the results. Typically, effective results are gained through an increase in run time.

In addition to the four different scheduler plans presented in the previous section, the two critical aspects of each plan will be evaluated. The evaluation will focus on improving the overall scheduler performance by determining the best plan to use as well as which loading rule and the best number of reallocation iterations to perform. The best plan is that which yields the best results (smallest deployment closure time and maximum lateness) in the best time. All experiments were performed on a personal computer. The computer used was an HP pavilion 7845 model with a pentium III, 866 MHz processor and 128 MB of RAM.

5.5.1 Deployment Models

The different plans will be evaluated in the context of three different deployment models. These models will describe increasingly more complex deployments. The first model, Model A, describes the deployment of the 101st Air Assault Division located at Fort

Campbell, Kentucky, to Saudi Arabia. This movement will be done entirely by air; this is the same model presented in Chapter 4 and is depicted again in Figure 5.2.

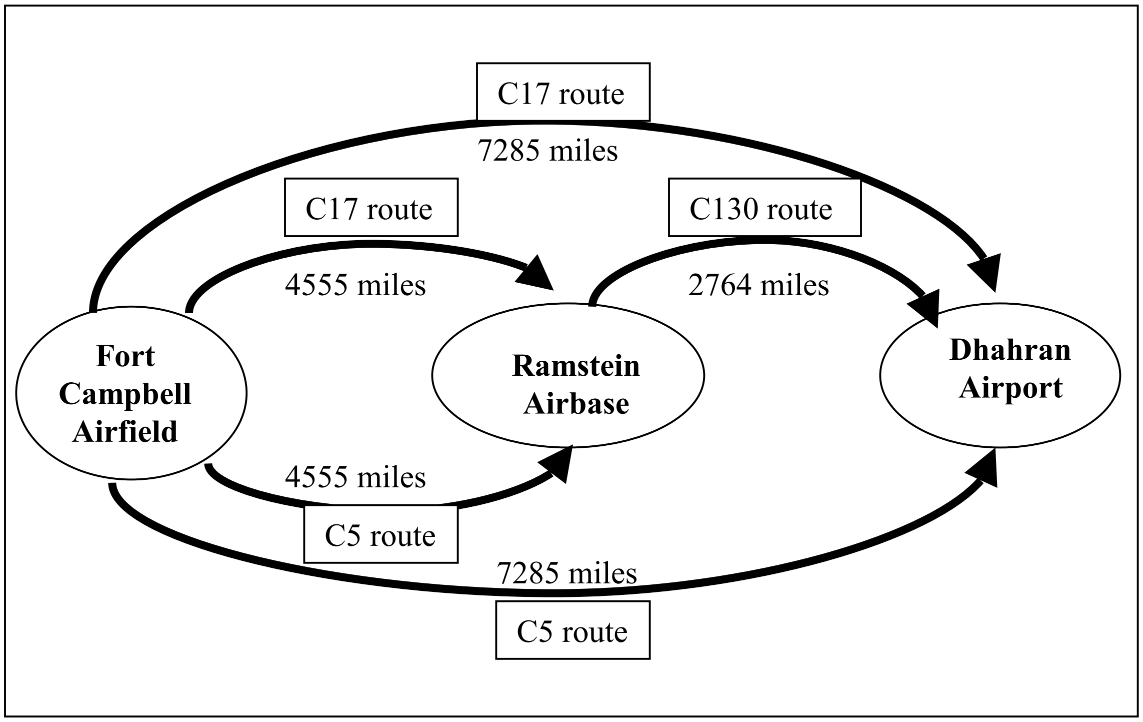


Figure 5.2 Model A – One Origin; Air Deployment

The second model, Model B, describes the deployment of two divisions, the 101st and the 1st Armored Division located at Fort Hood, Texas. As in Model A, this deployment will be entirely conducted by air. Additional routes are added from Fort Hood to Ramstein and to Dhahran. While the entire 101st is deployable by air, only some of the 1st is air deployable. If the planner selects the 1st to deploy by air, DSAT informs the planner that some equipment can not deploy by air. The planner is then given several choices, either select sea movement as an option, or leave the non-airdeployable equipment behind. The use of Model B assumes that all non-airdeployable equipment is left behind. Model B is depicted in Figure 5.3.

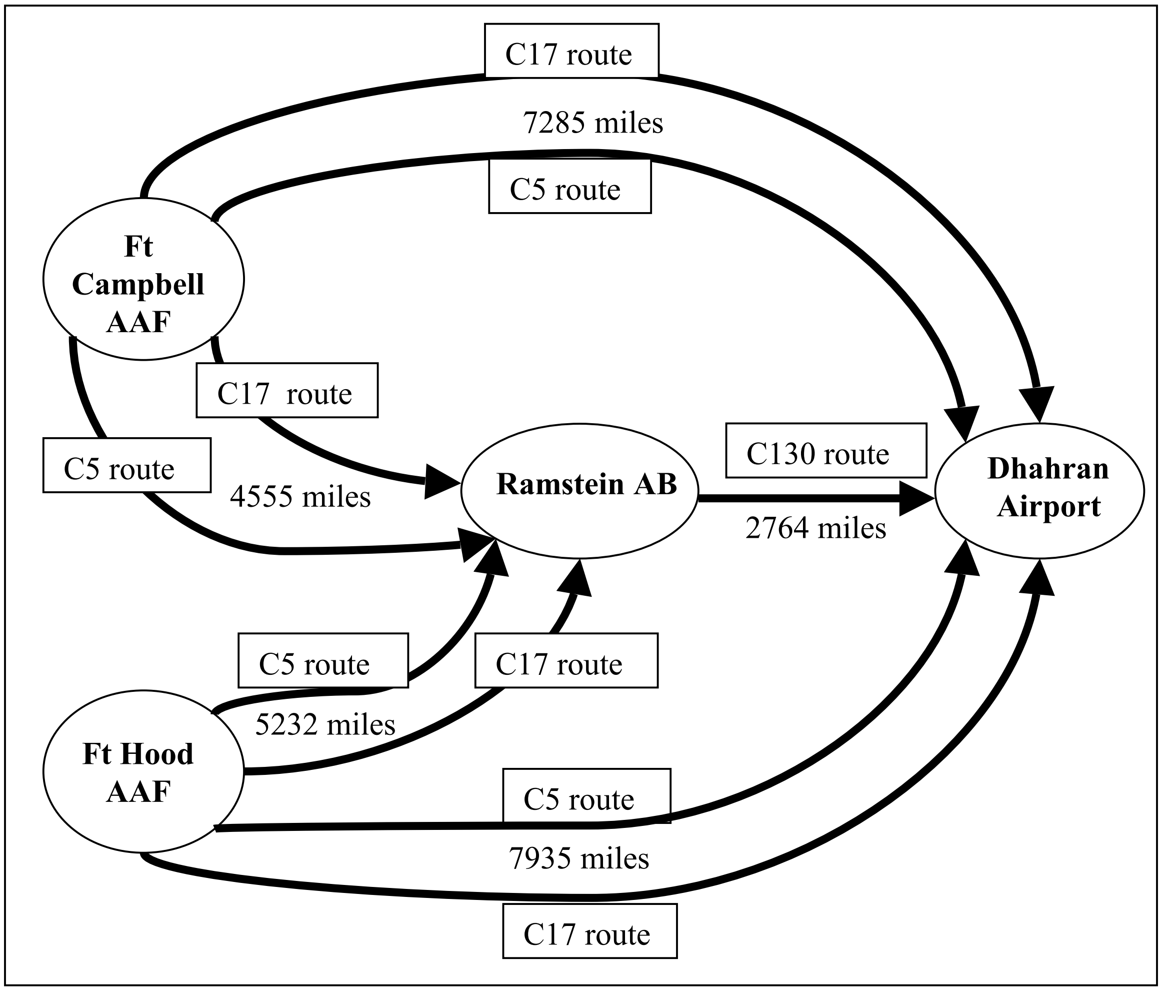


Figure 5.3 Model B – Two Origins; Air Deployment

The final model, Model C describes the deployment of the 101st and the 1st using air, rail and sea movement. The air routes remain as depicted in Model B. Each Division can ship out of either the Beaumont or the Savannah seaports, located in Texas or Georgia, respectively. Fast Sealift Ships, FSS, operate on the sea routes between the Beaumont and the Dammam seaports and between the Savannah and Dammam ports. A rail route connects each fort to each CONUS seaport. Trains, each consisting of 100 standard 60-foot flat cars, operate on the rail routes. This model is shown in Figure 5.4.

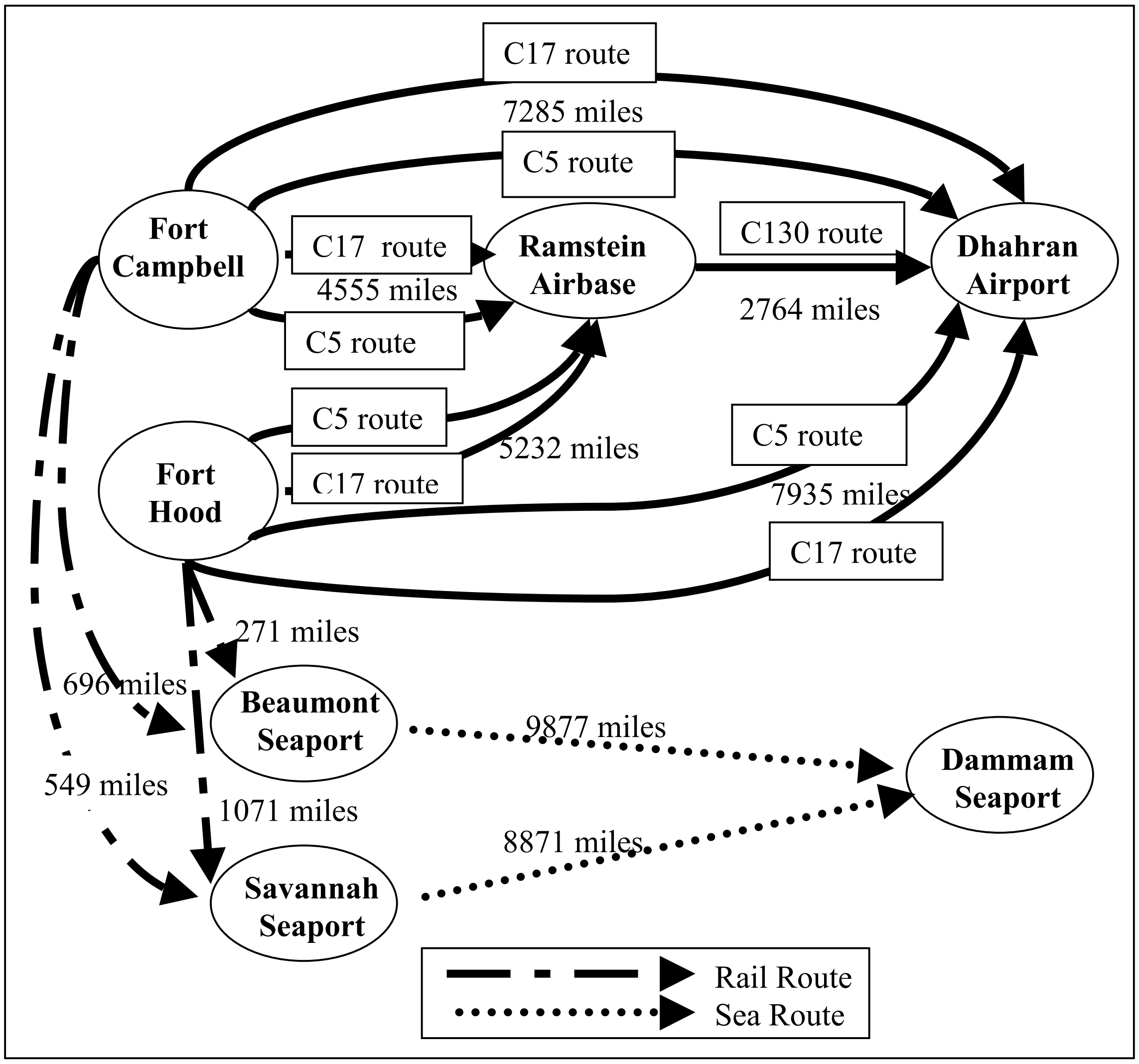


Figure 5.4 Model C – Two Origins, Rail, Sea and Air Deployment

Tables 5.4 and 5.5 contain the standard asset operating characteristics and the port infrastructure characteristics, respectively. DSAT does not compute a port throughput rate; instead, the port infrastructure is passed into the scheduler routine via the generated text input file. The Virtual Factory simulates the deployment process through the port, given the port infrastructure.

Table 5.4 Transportation Asset Operating Characteristics

Asset	Speed*	Weight Capacity (STONs)	Load Time (hours)	Unload Time (hours)	Average Maintenance Time (hours)
C17	410	45	2.25	2.25	2.25
C5	409	61.3	4.25	3.25	3.25
C130	270	12	1.5	1.5	1.5
Fast Sealift Ship (FSS)	27	28560	48	48	24
Train***	22	6750	24	24	0**

* All speeds are in knots except the train speed which is in miles per hour.
** Maintenance performed during load/unload times.
*** One train consists of 100 standard 60-foot flatcars.

Table 5.5 Port Infrastructure Characteristics

Port	Number of Runways	MOG or Number of Berths
Ft. Campbell Army Airfield	1	5
Ft. Hood Army Airfield	1	5
Ramstein Air Force Base	1	5
Dhahran International Airport	1	5
Beaumont, Texas	NA	6
Savannah, Georgia	NA	6
Dammam, Saudi Arabia	NA	13

5.5.2 Methodology

The following comparisons will be made for each deployment model. First, a baseline performance level for each of the four basic scheduler plans will be established. This baseline performance evaluation uses each scheduler plan in conjunction with the original vehicle loading rules and the original reallocation iteration count. This represents the original scheduler routine using the four proposed scheduling plans as described in section 5.3. Initially, only the vehicle loading rules will be varied for each plan. The performance of each option will be measured with respect to the smallest deployment closure time and maximum lateness established as well as the amount of computer run time needed to obtain the results. As a result of this first comparison, any plans that are clearly dominated will be eliminated. Next, the reallocation iteration count will be varied for each remaining plan. Again, the performance of each option will be measured with respect to closure time, maximum lateness and computer run time. As a result of this second comparison, a recommendation regarding which scheduler plan, including vehicle loading rule and reallocation iteration count, should be incorporated into future DSAT versions is made.

5.5.3 Results

The methodology was applied to varying scenarios for each deployment model described earlier. Table 5.6 shows the input parameters for each Model A scenario; these are in addition to the asset operating characteristics and the port infrastructures.

Table 5.6 Model A DSAT Input Parameters

Parameters	***Scenario 1***	***Scenario 2***	***Scenario 3***	***Scenario 4***
# STONs	2,235	11,136	21,097	32,211
# Jobs	511	2,869	4,853	8,180
# Vehicles to Each Route				
C17	5	50	60	70
C5	5	50	60	70
C130	15	90	90	100

The baseline results for each of the four scheduler routine plans are included in Table 5.7.

Table 5.7 Model A Baseline Results

Scenario	***Plan***	C_{max}	L_{max}	***Run Time***
1	*1*	7.39	1.39	0.15
	2	7.79	1.79	0.11
	3	7.39	1.39	0.13
	4	9.66	3.66	0.09
2	*1*	22.88	15.88	1.15
	2	23.84	16.84	1.07
	3	22.06	15.06	1.18
	4	22.66	15.66	0.71
3	*1*	34.84	13.84	2.67
	2	42.88	21.88	1.24
	3	34.73	13.73	3
	4	43.83	22.83	1.49
4	*1*	56.61	25.61	6.52
	2	62.39	31.39	2.37
	3	57.8	26.8	7.12
	4	68.38	37.38	4.47
Note: Run Time in seconds, all others in days				

The results using the Fail75 loading rule are displayed in Table 5.8. The relative change data for C_{max} and L_{max} shows the percentage of change between the Fail75 loading rule and the original rule. For example, the Fail75 relative change in C_{max} is computed as:

$$RelativeChange = \frac{Fail75\,C_{\max} - Baseline\,C_{\max}}{Fail75\,C_{\max}}$$

A negative number indicates the percentage of reduction in C_{max} gained while a positive number indicates the percentage of increase in C_{max} gained. The relative change for L_{max} values is computed in a similar manner. The data for the runtime simply shows the difference in runtimes between the two options. A change of 10% or greater is considered significant. Using the Fail75 loading rule resulted in an improved C_{max} and L_{max} value for every plan and every scenario; almost all of the L_{max} improvements were significant. The Run Time increase with this loading rule was insignificant.

Table 5.8 Model A Fail75 Results

					Relative Change**		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	6.28	0.28	0.17	-0.177	-3.964	0.02
	2	7.63	1.63	0.15	-0.021	-0.098	0.04
	3	6.26	0.26	0.17	-0.181	-4.346	0.04
	4	9.34	3.34	0.12	-0.034	-0.096	0.03
2	*1*	21.23	14.23	1.34	-0.078	-0.116	0.19
	2	22.27	15.27	0.92	-0.070	-0.103	-0.15
	3	20.72	13.72	1.39	-0.065	-0.098	0.21
	4	21.91	14.91	0.85	-0.034	-0.050	0.14
3	*1*	32.44	11.44	2.95	-0.074	-0.210	0.28
	2	40.07	19.07	1.67	-0.070	-0.147	0.43
	3	32.89	11.89	3.2	-0.056	-0.155	0.2
	4	42.09	21.09	1.86	-0.041	-0.083	0.37
4	*1*	53.20	22.2	6.76	-0.064	-0.154	0.24
	2	58.36	27.36	2.58	-0.069	-0.147	0.21
	3	53.19	22.19	7.68	-0.087	-0.208	0.56
	4	63.07	32.07	3.92	-0.084	-0.166	-0.55
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

The remaining loading rules, Goal90, Goal95 and WholeList had identical results in C_{max} and L_{max}. The Run Times for Goal90 and Goal95 were essentially the same. Of these three loading rules, the WholeList loading rule had the fastest run time for every run. The results for the WholeList loading rule are displayed in Table 5.9. The relative change data shows the difference between the WholeList loading rule and the Fail75 rule.

Table 5.9 Model A WholeList Results

					Relative Change**		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	6.26	0.26	0.173	-0.003	-0.077	0.00
	2	7.63	1.63	0.197	0.000	0.000	0.05
	3	6.26	0.26	0.172	0.000	0.000	0.00
	4	9.34	3.34	0.121	0.000	0.000	0.00
2	*1*	20.72	13.72	1.993	-0.025	-0.037	0.65
	2	21.53	14.53	1.567	-0.034	-0.051	0.65
	3	20.2	13.2	2.009	-0.026	-0.039	0.62
	4	21.51	14.51	1.332	-0.019	-0.028	0.48
3	*1*	31.44	10.44	5.41	-0.032	-0.096	2.46
	2	38.39	17.39	4.232	-0.044	-0.097	2.56
	3	31.81	10.81	5.715	-0.034	-0.100	2.52
	4	37.8	16.8	4.057	-0.113	-0.255	2.20
4	*1*	51.33	20.33	13.651	-0.036	-0.092	6.89
	2	55.26	24.26	10.11	-0.056	-0.128	7.53
	3	50.99	19.99	15.144	-0.043	-0.110	7.46
	4	59.03	28.03	11.588	-0.068	-0.144	7.67
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

The WholeList loading rule yielded better results than the Fail75 rule in almost every instance. Only the first scenario using plans 2, 3 and 4 displayed no additional improvement. While there was only one significant improvement in C_{max}, there were several significant improvements in L_{max} in the larger two scenarios. The increased computer run time gained through use of this loading rule is acceptable. Over 500 jobs are being scheduled onto multiple machines in a matter of seconds. The comparison of the various loading rules used with Model A scenarios indicates that always checking the entire job list (WholeList loading rule) is the best option to use. Under all loading rules, either plan 1 or plan 3 was the plan with the best results with regards to C_{max} and L_{max}.

The same comparison was done between the various loading rules for scenarios using deployment Model B. Table 5.10 shows the input parameters for the different Model B scenarios.

Table 5.10 Model B DSAT Input Parameters

Parameters	***Scenario 1***	***Scenario 2***	***Scenario 3***	***Scenario 4***
# STONs	12,152	59,015	62,928	106,227
# Jobs	1,559	9,067	9,016	17,579
# Vehicles to Each Route	▨	▨	▨	▨
C17	50	70	85	90
C5	50	70	85	90
C130	100	100	100	100

The baseline results for each scenario under Model B and each scheduler plan are shown in Table 5.11.

Table 5.11 Model B Baseline Results

Scenario	*Plan*	*Cmax*	*Lmax*	*Run Time*
1	*1*	18.55	8.55	0.67
	2	18.84	8.84	0.38
	3	17.89	7.89	0.61
	4	20.58	10.58	0.4
2	*1*	81.85	51.85	8.54
	2	90.14	60.14	2.78
	3	85.65	55.65	9.15
	4	98.56	68.56	2.45
3	*1*	90.69	55.69	6.49
	2	102.83	67.83	2.55
	3	94.55	59.55	7.72
	4	103.08	68.08	3.73
4	*1*	151.01	101.01	42.17
	2	179.99	129.99	5.79
	3	156.59	106.59	40.73
	4	184.87	134.87	5.07
Note: Run Time in seconds, all others in days				

The various loading rules offered results similar to those with Model A scenarios. First are the results using the Fail75 loading rule shown in Table 5.12. The relative change data, computed as for the Model A analysis, shows the difference between the Fail75 rule and the original loading rule.

Table 5.12 Model B Fail75 Results

					Relative Change**		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	17.00	7.00	0.92	-0.091	-0.221	0.25
	2	17.85	7.85	0.59	-0.055	-0.126	0.21
	3	16.45	6.45	0.71	-0.088	-0.224	0.10
	4	18.73	8.73	0.50	-0.099	-0.212	0.10
2	*1*	77.52	47.52	8.93	-0.056	-0.091	0.39
	2	85.59	55.59	4.27	-0.053	-0.082	1.49
	3	80.70	50.70	8.66	-0.061	-0.098	-0.50
	4	91.68	61.68	3.47	-0.075	-0.112	1.02
3	*1*	84.16	49.16	7.05	-0.078	-0.133	0.56
	2	96.77	61.77	3.56	-0.063	-0.098	1.01
	3	87.51	52.51	8.28	-0.080	-0.134	0.56
	4	96.40	61.40	3.41	-0.069	-0.109	-0.32
4	*1*	140.63	90.63	36.80	-0.074	-0.115	-5.37
	2	162.84	112.84	7.84	-0.105	-0.152	2.05
	3	146.49	96.49	35.00	-0.069	-0.105	-5.73
	4	171.37	121.37	7.03	-0.079	-0.111	1.96
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

All Model B scenarios had improvements in C_{max} and L_{max} for all plans. Again, almost every improvement in L_{max} was significant. Increases in the run time required were insignificant.

The remaining loading rules, Goal90, Goal95 and WholeList, had identical results in C_{max} and L_{max} for all plans and all scenarios. Of these three rules, the WholeList option had the shortest run times. The results for the WholeList option are displayed in Table 5.13. The relative change shows the difference between the WholeList rule and the Fail75 rule.

Table 5.13 Model B WholeList Results

					*Relative Change***		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	16.70	6.70	1.05	-0.018	-0.045	0.13
	2	17.16	7.16	0.73	-0.040	-0.098	0.14
	3	15.45	5.45	0.95	-0.065	-0.184	0.25
	4	17.53	7.53	0.78	-0.068	-0.159	0.29
2	*1*	70.38	40.38	21.30	-0.102	-0.177	12.37
	2	79.21	49.21	21.16	-0.081	-0.130	16.89
	3	73.48	43.48	19.49	-0.098	-0.166	10.84
	4	87.14	57.14	15.23	-0.052	-0.079	11.76
3	*1*	76.58	41.58	16.71	-0.099	-0.182	9.66
	2	89.44	54.44	15.04	-0.082	-0.135	11.48
	3	79.51	44.51	18.49	-0.101	-0.180	10.21
	4	88.67	53.67	14.44	-0.087	-0.144	11.03
4	*1*	126.78	76.78	86.86	-0.109	-0.180	50.07
	2	159.30	109.30	80.00	-0.022	-0.032	72.16
	3	131.18	81.18	85.16	-0.117	-0.189	50.16
	4	156.04	106.04	61.98	-0.098	-0.145	54.95
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

The additional improvements in L_{max} were significant in almost every instance. The improvements in many C_{max} results were also significant. The increase in run time incurred by using the WholeList loading rule remained small for the first three scenarios, but was quite significant for the last scenario with an increase of about one minute. Overall, the number of significant improvements in C_{max} and L_{max} warranted the increase in run time. The WholeList option was the best loading rule in all Model B scenarios. In comparing the scheduler plans used (using the WholeList rule), either plan 1 or plan 3 yielded the best results for C_{max} and L_{max}.

The same comparison was performed on different scenarios of Model C. The input parameters for the different Model C scenarios are presented in Table 5.14.

Table 5.14 Model C DSAT Input Parameters

Parameters	*Scenario 1*	*Scenario 2*	*Scenario 3*	*Scenario 4*
# STONs	20,583	36,217	77,374	129,590
# Jobs	1,690	6,416	7,479	17,942
# Vehicles to Each Route	▨	▨	▨	▨
C17	40	80	60	90
C5	40	80	60	90
C130	100	100	100	100
FSS	2	2	2	2
Train	2	2	2	2

The baseline results for the Model C scenarios are presented in Table 5.15.

Table 5.15 Model C Baseline Results

Scenario	*Plan*	*Cmax*	*Lmax*	*Run Time*
1	*1*	22.72	-2.28	0.92
	2	22.72	-2.28	0.51
	3	23.56	3.56	0.79
	4	23.56	3.56	0.48
2	*1*	43.38	3.38	3.27
	2	53.00	13.00	1.65
	3	46.15	6.15	3.41
	4	58.25	18.25	1.85
3	*1*	63.30	33.30	5.57
	2	86.26	56.26	2.17
	3	63.52	33.52	4.85
	4	75.63	45.63	1.98
4	*1*	125.62	70.62	26.13
	2	172.04	117.04	5.01
	3	110.56	55.56	25.20
	4	140.19	85.19	4.60
Note: Run Time in seconds, all others in days				

Using the Fail75 loading rule led to some interesting results for the Model C scenarios. The results for this rule are shown in Table 5.16.

Table 5.16 Model C Fail75 Results

					Relative Change*		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	22.72	-2.28	0.90	0.000	0.000	-0.01
	2	22.72	-2.28	0.64	0.000	0.000	0.13
	3	23.56	3.56	0.91	0.000	0.000	0.12
	4	23.56	3.56	0.61	0.000	0.000	0.13
2	*1*	41.17	1.17	3.38	-0.054	-1.889	0.11
	2	50.17	10.17	2.10	-0.056	-0.278	0.45
	3	42.49	3.56	3.82	-0.086	-0.728	0.41
	4	54.40	14.40	2.38	-0.071	-0.267	0.53
3	*1*	61.58	31.58	6.27	-0.028	-0.054	0.71
	2	80.92	50.92	2.95	-0.066	-0.105	0.78
	3	60.97	31.09	5.33	-0.042	-0.078	0.48
	4	72.95	42.95	2.57	-0.037	-0.062	0.59
4	*1*	101.96	46.96	24.11	-0.232	-0.504	-2.02
	2	158.99	103.99	6.86	-0.082	-0.125	1.85
	3	112.37	57.37	24.33	0.016	0.031	-0.87
	4	130.24	75.24	6.17	-0.076	-0.132	1.56
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

For the first scenario, there was no change in C_{max} or L_{max} for any of the scheduler plans. For the fourth scenario, using scheduler plan 3 resulted in a slight increase in both C_{max} and L_{max}. All other scenarios and scheduler plans showed a reduction in both C_{max} and L_{max}; some reductions were very significant. All run times were only slightly longer using this loading rule.

The remaining loading plans resulted in identical C_{max} and L_{max} values. Of the three rules, the WholeList option had the shortest run times. The results for the WholeList rule are displayed in Table 5.17. The relative changes reflect the difference between the WholeList rule and the Fail75 rule.

Table 5.17 Model C WholeList Results

					*Relative Change***		
Scenario	***Plan***	***Cmax****	***Lmax****	***Run Time****	***Cmax***	***Lmax***	***Run Time***
1	*1*	22.72	-2.28	1.11	0.000	0.000	0.21
	2	22.72	-2.28	0.78	0.000	0.000	0.14
	3	23.56	3.56	1.06	0.000	0.000	0.16
	4	23.56	3.56	0.77	0.000	0.000	0.16
2	*1*	36.54	-1.94	6.08	-0.127	-1.604	2.70
	2	49.01	9.01	5.43	-0.024	-0.129	3.33
	3	43.25	3.56	6.97	0.018	0.000	3.15
	4	51.10	11.10	5.91	-0.065	-0.297	3.53
3	*1*	59.54	29.54	11.92	-0.034	-0.069	5.64
	2	72.65	42.65	12.62	-0.114	-0.194	9.68
	3	58.15	31.09	9.87	-0.048	0.000	4.54
	4	66.97	36.97	7.12	-0.089	-0.162	4.55
4	*1*	105.30	50.30	61.69	0.032	0.066	37.58
	2	142.47	87.47	63.89	-0.116	-0.189	57.03
	3	108.07	53.07	54.86	-0.040	-0.081	30.53
	4	119.14	64.14	41.05	-0.093	-0.173	34.89
* Run Time in seconds, all others in days							
** Positive value indicates increase; negative value indicates decrease							

In all but one case, the L_{max} value either remained the same or was improved. Most of these improvements were significant. In two cases, the C_{max} values increased slightly. Although using the WholeList loading rule does significantly increase the computer run time, the results this rule presents are significantly superior to the Fail75 results. Using the WholeList loading rule, scheduler plan 1 or 3 always yielded the best results for both C_{max} and L_{max}.

Based on the above analysis, the following conclusions and recommendations can be made. First, the WholeList rule was the best loading rule for all three deployment models and should be incorporated into the scheduler code for later DSAT versions. All remaining work concerning scheduler routine improvement uses scheduler plans that incorporate the WholeList loading rule. Second, two scheduler plans, plans 2 and 4, were clearly dominated and should be removed from further consideration. Scheduler plans 1 and 3 are the only plans considered in further analysis.

The remaining analysis deals with determining the preferred number of reallocation iterations to use within a given scheduling plan. The different options under consideration all use a variable number of iterations. The stopping rule halts further execution after *x* consecutive iterations with no improvement found. The different versions are: 20, 30, 40 and 50 consecutive iterations. Again, the different options will be compared with respect to C_{max}, L_{max} and computer run time. The results for the Model A scenarios are shown in Table 5.18. The relative change data for the various options are computed for C_{max}, L_{max} and computer run time. The computer run time information is simply the difference in runtimes between the 20Consec and Original options, the 30Consec and 20Consec options, the 40Consec and 30Consec options and the 50Consec and 40Consec options, respectively. A positive number indicates an increase in run time while a negative number indicates a decrease in computer run time. The relative change in C_{max} for various options is computed by dividing the difference in C_{max} values between two options by the C_{max} value of the indicated option. For example, the 20Consec relative change in C_{max} is computed as:

$$Relative\,Change = \frac{20Consec\,C_{\max} \text{ - } Original\,C_{\max}}{20Consec\,C_{\max}}$$

A negative number indicates the percentage of reduction in C_{max} gained while a positive number indicates the percentage of increase in C_{max} gained. The relative change for L_{max} values are computed in a similar manner. A 10% reduction in either C_{max} or L_{max} is considered a significant change. The relative change data in Table 5.18 indicates that increasing the iteration count beyond the 20Consec option yields insignificant improvements in either C_{max} or L_{max}. The corresponding increases in run time usually remained insignificant, but some run time increases were over one minute.

Table 5.18 Model A Iteration Count Results

						*Relative Change***		
Iterations	***Scenario***	***Plan***	C_{max*}	L_{max*}	***Run Time****	C_{max}	L_{max}	***Run Time***
20	*1*	*1*	6.26	0.26	0.48	0.000	0.002	0.31
		3	6.26	0.26	0.35	0.000	0.002	0.18
	2	*1*	20.53	13.53	6.80	-0.009	-0.014	4.80
		3	19.91	12.91	11.21	-0.015	-0.023	9.20
	3	*1*	31.33	10.33	18.48	-0.003	-0.010	13.07
		3	31.81	10.81	13.45	0.000	0.000	7.73
	4	*1*	49.72	18.72	56.31	-0.032	-0.086	42.66
		3	49.15	18.15	97.54	-0.038	-0.102	82.40
30	*1*	*1*	6.26	0.26	0.48	0.000	0.000	0.00
		3	6.26	0.26	0.47	0.000	0.000	0.12
	2	*1*	20.53	13.53	8.29	0.000	0.000	1.49
		3	19.91	12.91	12.87	0.000	0.000	1.67
	3	*1*	31.33	10.33	23.13	0.000	0.000	4.66
		3	31.81	10.81	18.09	0.000	0.000	4.64
	4	*1*	49.72	18.72	66.08	0.000	0.000	9.77
		3	49.15	18.15	107.11	0.000	0.000	9.57
40	*1*	*1*	6.26	0.26	0.61	0.000	0.000	0.13
		3	6.26	0.26	0.60	0.000	0.000	0.12
	2	*1*	20.53	13.53	9.86	0.000	0.000	1.57
		3	19.91	12.91	14.35	0.000	0.000	1.48
	3	*1*	31.33	10.33	27.88	0.000	0.000	4.74
		3	31.66	10.66	51.67	-0.005	-0.015	33.58
	4	*1*	49.72	18.72	77.51	0.000	0.000	11.42
		3	48.93	17.93	168.39	-0.004	-0.012	61.28
50	*1*	*1*	6.26	0.26	0.73	0.000	0.000	0.12
		3	6.26	0.26	0.72	0.000	0.000	0.12
	2	*1*	20.53	13.53	11.40	0.000	0.000	1.54
		3	19.91	12.91	15.91	0.000	0.000	1.56
	3	*1*	31.33	10.33	32.64	0.000	0.000	4.76
		3	31.66	10.66	56.08	0.000	0.000	4.41
	4	*1*	49.72	18.72	88.69	0.000	0.000	11.18
		3	48.93	17.93	177.52	0.000	0.000	9.13
** Negative number indicates decrease; Positive Number indicates increase								
*Run Time measured in seconds; all others in days								

The results for Model B scenarios are shown in Table 5.19. The relative change data is computed in the same manner as for Table 5.18. This data shows no significant changes in C_{max} or L_{max} at iteration counts beyond the 20Consec option. The largest reduction in C_{max} or L_{max} for the 20Consec option occurred in the fourth scenario with plan 3; this indicated a 3% reduction in L_{max}.

The results for Model C scenarios are shown in Table 5.20. The relative change data is computed in the same manner as for the Models A and B data. This data also shows no significant improvements in C_{max} or L_{max} at iteration counts exceeding the 20Consec option.

Table 5.19 Model B Iteration Count Results

						*Relative Change***		
Iterations	*Scenario*	*Plan*	*C_{max}**	*L_{max}**	*Run Time**	*C_{max}*	*L_{max}*	*Run Time*
20	*1*	*1*	16.70	6.70	2.21	0.000	0.000	1.16
		3	15.45	5.45	2.15	0.000	0.000	1.19
	2	*1*	70.38	40.38	47.28	0.000	0.000	25.99
		3	73.33	43.33	90.90	-0.002	-0.003	71.40
	3	*1*	75.77	40.77	46.08	-0.011	-0.020	29.38
		3	78.83	43.83	64.25	-0.009	-0.015	45.76
	4	*1*	125.08	75.08	219.49	-0.014	-0.023	132.63
		3	128.50	78.50	553.21	-0.021	-0.034	468.06
30	*1*	*1*	16.70	6.70	3.08	0.000	0.000	0.87
		3	15.45	5.45	2.90	0.000	0.000	0.76
	2	*1*	70.38	40.38	61.83	0.000	0.000	14.55
		3	73.33	43.33	107.14	0.000	0.000	16.25
	3	*1*	75.77	40.77	59.07	0.000	0.000	12.99
		3	78.83	43.83	75.63	0.000	0.000	11.38
	4	*1*	125.08	75.08	275.95	0.000	0.000	56.45
		3	128.50	78.50	610.05	0.000	0.000	56.84
40	*1*	*1*	16.70	6.70	3.99	0.000	0.000	0.91
		3	15.45	5.45	3.61	0.000	0.000	0.71
	2	*1*	70.38	40.38	78.61	0.000	0.000	16.77
		3	73.33	43.33	122.94	0.000	0.000	15.79
	3	*1*	75.77	40.77	72.43	0.000	0.000	13.36
		3	78.83	43.83	89.27	0.000	0.000	13.64
	4	*1*	125.08	75.08	333.03	0.000	0.000	57.08
		3	128.50	78.50	666.68	0.000	0.000	56.63
50	*1*	*1*	16.70	6.70	4.83	0.000	0.000	0.84
		3	15.45	5.45	4.37	0.000	0.000	0.75
	2	*1*	70.38	40.38	92.45	0.000	0.000	13.84
		3	73.33	43.33	136.93	0.000	0.000	13.99
	3	*1*	75.77	40.77	85.67	0.000	0.000	13.25
		3	78.83	43.83	101.96	0.000	0.000	12.69
	4	*1*	125.08	75.08	388.80	0.000	0.000	55.77
		3	128.50	78.50	726.35	0.000	0.000	59.67
** Negative number indicates decrease; Positive Number indicated increase								
*Run Time measured in seconds; all others in days								

Table 5.20 Model C Iteration Count Results

						*Relative Change***		
Iterations	***Scenario***	***Plan***	C_{max*}	L_{max*}	***Run Time****	C_{max}	L_{max}	***Run Time***
20	1	1	22.72	-2.28	2.27	0.000	0.000	1.16
		3	23.56	3.56	2.07	0.000	0.000	1.00
	2	1	32.38	-2.28	14.90	-0.128	-0.150	8.82
		3	43.25	3.56	12.42	0.000	0.000	5.46
	3	1	59.30	29.30	40.44	-0.004	-0.008	28.52
		3	58.93	28.93	44.93	0.013	-0.075	35.06
	4	1	99.60	44.60	137.07	-0.057	-0.128	75.38
		3	99.89	44.89	138.31	-0.082	-0.182	83.45
30	1	1	22.72	-2.28	3.26	0.000	-0.001	0.99
		3	23.56	3.56	2.95	0.000	-0.001	0.88
	2	1	32.38	-2.28	19.47	0.000	0.000	4.57
		3	43.25	3.56	15.17	0.000	-0.001	2.74
	3	1	59.30	29.30	47.19	0.000	0.000	6.75
		3	58.93	28.93	53.48	0.000	0.000	8.54
	4	1	99.60	44.60	168.63	0.000	0.000	31.56
		3	99.89	44.89	168.94	0.000	0.000	30.63
40	1	1	22.72	-2.28	4.25	0.000	0.000	0.99
		3	23.56	3.56	3.79	0.000	0.000	0.84
	2	1	32.38	-2.28	22.67	0.000	0.000	3.20
		3	43.25	3.56	18.11	0.000	0.000	2.94
	3	1	59.30	29.30	56.33	0.000	0.000	9.14
		3	58.93	28.93	62.83	0.000	0.000	9.35
	4	1	99.60	44.60	199.29	0.000	0.000	30.66
		3	99.89	44.89	199.34	0.000	0.000	30.39
50	1	1	22.72	-2.28	5.22	0.000	0.000	0.97
		3	23.56	3.56	4.64	0.000	0.000	0.86
	2	1	32.38	-2.28	26.67	0.000	0.000	4.00
		3	43.25	3.56	21.07	0.000	0.000	2.96
	3	1	59.30	29.30	65.33	0.000	0.000	9.00
		3	58.93	28.93	71.13	0.000	0.000	8.30
	4	1	99.60	44.60	228.89	0.000	0.000	29.60
		3	97.75	42.75	387.13	-0.022	-0.050	187.79
** Negative number indicates decrease; Positive Number indicated increase								
*Run Time measured in seconds; all others in days								

Based on the above results, the best number of reallocation iterations to perform is the 20Consec option. Although some reductions in either C_{max} or L_{max} occur with the other options, these changes were insignificant. Furthermore, the increases in run time were significant for all Models using the other options.

The plans' performance in respect to C_{max} and L_{max} for all Models using the 20Consec option is now considered. Table 5.21 summarizes both plans' performance. The "Best Results" column identifies, for each Model and scenario, the plan that yielded the smallest C_{max} and L_{max} values. The "Difference" columns show the difference in C_{max} or L_{max} values between the best plan (plan 1 or plan 3) and the other plan (plan 3 or plan 1). If the difference was less than 1 day, the plans were deemed equal in performance. The last column shows the best plan for each scenario and model. The run times for both plans were essentially equivalent in all cases and are not considered in this comparison.

Table 5.21 Scheduling Plan Performance Summary

Model	***Scenario***	***Best Results***	***Difference C_{max}***	***Difference L_{max}***	***Best Plan***
A	1	1,3	0.00	0.00	Tie
	2	3	0.63	0.63	Tie
	3	1	0.48	0.48	Tie
	4	3	0.57	0.57	Tie
B	1	3	1.25	1.25	3
	2	1	2.96	2.96	1
	3	1	3.06	3.06	1
	4	1	3.42	3.42	1
C	1	1	0.84	5.84	1
	2	1	10.87	5.84	1
	3	3	0.37	0.37	Tie
	4	1	0.29	0.29	Tie

Notice that for the twelve scenarios presented, plan 3 was the clear leader in performance in only one instance. Furthermore, when plan 1 was clearly better, it beat plan 3 by at least 3 days. When plan 3 beat plan 1, it did so by slightly more than one day. In all other instances the plans' performance was equivalent. Based on these observations, plan 1 performs better than plan 3 with respect to C_{max} and L_{max} values. Finally, the overall recommendation concerning future DSAT versions is to use the scheduler's original pair of heuristics in conjunction with the WholeList loading rule and the 20 consecutive iterations stopping rule.

Chapter 6

Evaluating DSAT's Performance

This chapter will focus on evaluating the effectiveness of the Deployment Scheduling Analysis Tool. The tool's effectiveness will be determined by two methods. The first method entails the determination of how well DSAT utilizes the transportation assets. The second method entails the comparison of DSAT's results in C_{max} and L_{max} to DANTE's lower bound for C_{max} and *COMFLOW*'s lower bound for L_{max} and the corresponding conditional lower bound for C_{max}.

6.1 Asset Utilization Analysis

For this paper, asset utilization will refer to how effectively a transportation asset moves cargo over a specific route. Each asset has several capacities to consider when loading its cargo, these are its weight, volume and area capacities. Rarely does an asset's cargo fully occupy all three capacities simultaneously. An asset can be "full" with respect to one capacity, say weight, and, at the same time, less than full with respect to the other capacities. An asset will be considered "fully utilized" if at least one of its capacities is sufficiently utilized. What is meant by "sufficiently utilized" is described next.

6.1.1 TEA Deployment Analysis Information

The MTMC_TEA Pamphlet 700-5, Deployment Planning Guide, is published by the Transportation Engineering Agency of the Military Traffic Management Command. This pamphlet provides broad transportation planning data designed to help planners make gross estimates about transportation requirements. It identifies the types and amounts of transportation assets needed to deploy various military units. The unit compositions used in this guide are generic in structure and might not match actual units. The unit deployment requirements data presented in the pamphlet was generated by running MTMC_TEA's Transportability Analysis Reports Generator (TARGET) on a unit's Tables of Organization and Equipment (TOE). A unit's TOE is a generic description of its organizational structure and composition. It includes a standardized list of equipment with its characteristic data (weight, height, width, length). TARGET estimated the number of missions, by asset type, necessary to move the selected unit equipment from the CONUS installations to the ports of embarkation (POEs), from the POEs to the ports of debarkation (PODs), and from the PODs to the tactical assembly area (TAA). A mission is one trip of a transportation asset to deliver cargo to a destination.

The Deployment Planning Guide does not provide information concerning the weight capacities of FSSs. Instead, ship mission planning requirements are based solely on the unit square footage requirements and the individual ship's square footage capacities. A FSS has an average useable space of 152,774 square feet. There are 8 FSSs in service today; based on their design characteristics, the average cargo deadweight capacity is 16,156 STONs.

The amount of weight a plane can carry for any mission is referred to as the allowable cabin load (ACL). The ACL for any plane is established by several factors including the weather and the flight distance. The greater the flight distance, the more fuel the plane must carry, thus reducing the amount of cargo it can carry. Using a 3,200 nautical mile leg, the following TEA data is based on an ACL of 178,000 pounds for a C5 and 130,000 pounds for a C17. The C130 ACL of 38,200 pounds is based on a 1,000 nautical mile leg.

The TARGET estimated mission requirements to deploy an air assault division and an armored division are identified in Table 6.1. This table also identifies the units' weight and square footage requirements.

Table 6.1 TEA Extract of TARGET Analysis

Unit Data			**Strategic Missions**			**Theater Missions**	
Division Type	**STONs**	**SQ FT**	**C17[1]**	**C5[2]**	**FSS[4]**	**STONs[5]**	**C130[3]**
Air Assault	34,938	913,387	766	600	7	22,819	1,999
Armored	99,921	1,406,284	1,722	1,284	9.8	26,540	2,234

[1]Based on Allowable Cabin Load of 130,000 lbs. for C17, critical leg of 3,200 nm.

[2]Based on Allowable Cabin Load of 178,000 lbs. for C5, critical leg of 3,200 nm.

[3]Based on Allowable Cabin Load of 38,200 lbs. for C130, critical leg of 1,000 nm.

[4]Based on Average Useable Square Footage of 152,774 sq. ft. for FSS.

[5] TEA table indicated this weight moved by C130 missions.

The number of missions identified in the strategic missions columns reflects the estimated number of asset specific missions required to deploy 100% of the unit to the strategic APOD or SPOD by that specific asset. The number of missions identified in the theater missions column reflects the estimated number of C130 missions required to deploy the identified unit weight requirement to the theater APOD.

The information presented in Table 6.1 was used to compute the average percentage of asset weight or area capacity used per mission during the deployment of a particular type division. This information represents the accepted average percentages that deployment planners use in planning their various deployments. This information is presented for both strategic and intra-theater lift assets in Table 6.2.

Table 6.2 TEA Analysis: Asset Average Percentage of Capacity Utilization

Unit Data	**Strategic Lift Analysis**				
Division Type	*Asset*	*Average Payload Weight (STONs / Missions)*	*Average Payload Area (Sq. Ft / Missions)*	*Average % Weight Capacity Utilized (Payload / ACL)*	*Average % Area Capacity Utilized (Payload / Useable Area)*
Air Assault	C17	45.61	▨	0.70	▨
	C5	58.23	▨	0.65	▨
	FSS	4,991.14	130,483.86	0.31	0.85
Armored	C17	58.03	▨	0.89	▨
	C5	77.82	▨	0.87	▨
	FSS	10,196.02	143,498.37	0.63	0.94
Unit Data	**Theater Lift Analysis**				
Division Type	*Asset*	*Average Payload Weight (STONs / Missions)*	*Average Payload Area (Sq. Ft / Missions)*	*Average % Weight Capacity Utilized (Payload / ACL)*	*Average % Area Capacity Utilized (Payload / Useable Area)*
Air Assault	C130	11.42	▨	0.60	▨
Armored	C130	11.88	▨	0.62	▨

The average percent of weight capacity utilized per mission for the FSS is based on the average cargo deadweight capacity of today's FSS fleet. The values in Table 6.2 will be used for comparison with the asset utilization percentages computed as a result of DSAT's deployment schedule. If the asset utilization percentages are at least equivalent to those in Table 6.2, then DSAT's schedule would "sufficiently utilize" the deployment assets.

6.1.2 DSAT Deployment Analysis Information

Special care must be taken to construct deployment scenarios for DSAT so that an accurate comparison of asset utilization percentages can be made. For this portion of DSAT analysis, five special deployment scenarios are presented. The first four scenarios involve either the 101st Air Assault Division or the 1st Armored Division; the unit will deploy entirely by air. A strategic lift (from home base to Ramstein AB) will utilize only either C17 or C5 assets. An intra-theater lift (from Ramstein to Saudi Arabia) will use a mix of assets similar to the mix used in MTMC_TEA's pamphlet. The last scenario involves both units deploying by rail to separate seaports and strategic sea lift (from port to Saudi Arabia) using FSSs. In all scenarios, DSAT used the ACLs and useable area as stated in the previous section. The special scenarios are summarized in Table 6.3.

Table 6.3 Special Deployment Scenarios

Scenario	*Unit*	*Strategic Lift Asset*	*Intra theater Asset*
1	101st	C17	C130, C17
2	101st	C5	C130, C5
3	1st	C17	C130, C17
4	1st	C5	C130, C5
5	1st and 101st	FSS	NA

As the scheduler routine in DSAT simulated the deployment, information concerning each asset's cargo load was obtained. DSAT captured information concerning weight and volume utilization of air and rail assets and weight and area utilization of sea assets. This information was printed out in a text file. Post processing the information contained in this text file allowed one to compute the number of asset missions conducted, the total weight moved by that asset type, the average asset payload in STONs and volume or area and the average percentage of asset capacity utilized. This information is presented in Table 6.4.

Table 6.4 DSAT Analysis: Asset Average Percentage of Capacity Utilization

Unit Data	**Strategic Lift Analysis**				
Division Type	*Asset*	*Average Payload Weight (STONs / Missions)*	*Average Payload Volume or Area (capacity / Missions)*	*Average % Weight Capacity Utilized (Payload / ACL)*	*Average % Volume or Area Capacity Utilized (Payload / Capacity)*
Air Assault	C17	52.89	10,944.10	0.81	0.83
	C5	69.88	15,681.44	0.79	0.83
	FSS	9,958.78	146,642.43	0.62	0.96
Armored	C17	62.33	7,483.36	0.96	0.57
	C5	85.36	10,648.45	0.96	0.56
	FSS	12,261.67	140,872.00	0.76	0.92
Unit Data	**Theater Lift Analysis**				
Division Type	*Asset*	*Average Payload Weight (STONs / Missions)*	*Average Payload Volume or Area (# / Missions)*	*Average % Weight Capacity Utilized (Payload / ACL)*	*Average % Volume or Area Capacity Utilized (Payload / Capacity)*
Air Assault	C130	13.52	2,811.25	0.71	0.94
Armored	C130	16.32	2,639.43	0.85	0.89

These asset utilization percentages are comparable to the standard planning factors for asset utilization as generated by TARGET. The DSAT generated deployment schedule has a higher weight capacity utilization percentage for all strategic and intra theater air assets. These results were higher by at least 7%. The strategic sea lift asset utilization percentages for both weight and area capacity are also comparable. The DSAT generated deployment schedule had a higher FSS weight capacity utilization percentage for all sea lift assets. This utilization was larger by at least 13%. Regarding the ship's area capacity utilization percentages, the DSAT generated schedule for the air assault division was higher by 11%. The TARGET generated utilization percentage for the armored division was higher by a mere 2%. Overall, DSAT appears to sufficiently utilize all assets.

These results, however, are based on very simplistic deployment scenarios. Each unit had dedicated assets for its deployment. In no scenario are deployment assets shared between units. Furthermore, most scenarios involve only one mode of deployment, air. There is only one scenario with multiple modes, rail and sea. DSAT's power and versatility comes from its ability to rapidly schedule large deployments using all modes of transportation. Recall the three deployment models presented in Chapter 5. The following selection of graphs detail capacity utilization for individual missions for various assets employed in the different models. Each graph shows the simultaneous weight and volume (for air and rail assets) or weight and area (for sea assets) capacity utilization percentage for individual missions performed throughout the duration of the deployment. The VF time interval in each graph refers to the Virtual Factory's internal calendar time at which the asset was unloaded at its destination. The Virtual Factory counts time in 15 minute intervals.

The asset utilization information for all C130 aircraft operating in a Model A deployment scenario is shown in Figure 6.1. The initial missions unloading from time period 130 to 163 are characterized by a significantly lower percentage of utilized capacity in either category. As the amount of equipment at the Ramstein AB increased, so does the C130 utilization. More available equipment creates more choices to fill remaining capacity for later missions. The remaining missions, except the last, met or exceeded MTMC_TEA's planning factors for average percentage of capacity utilization.

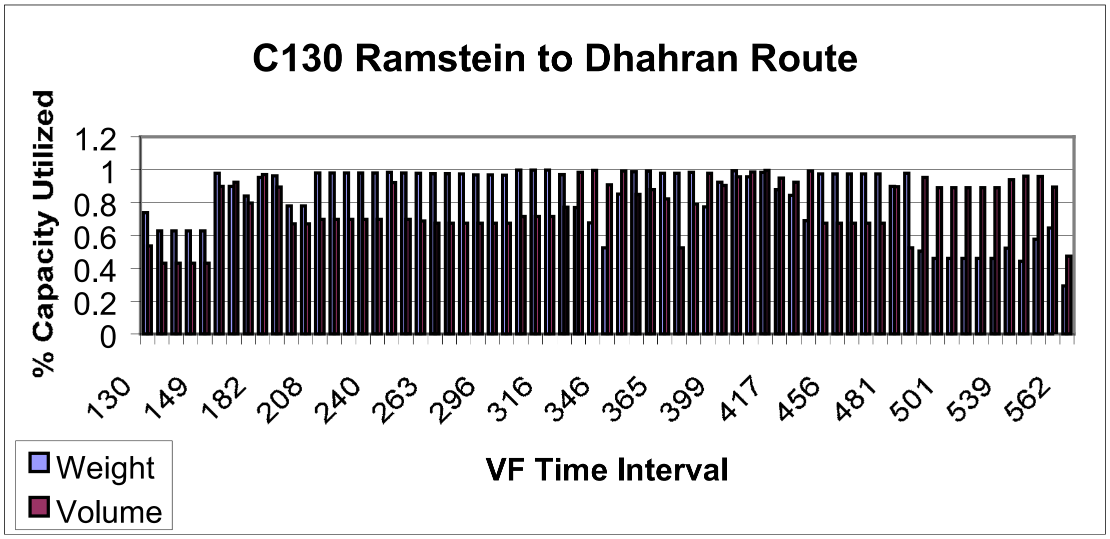


Figure 6.1 Model A C130 Asset Utilization Information

The asset utilization information for C5's operating on the Ft. Hood AAF to Ramstein AB route in a Model B scenario is shown in Figure 6.2. Note that every mission, except the last one, was completely filled with respect to either weight or volume.

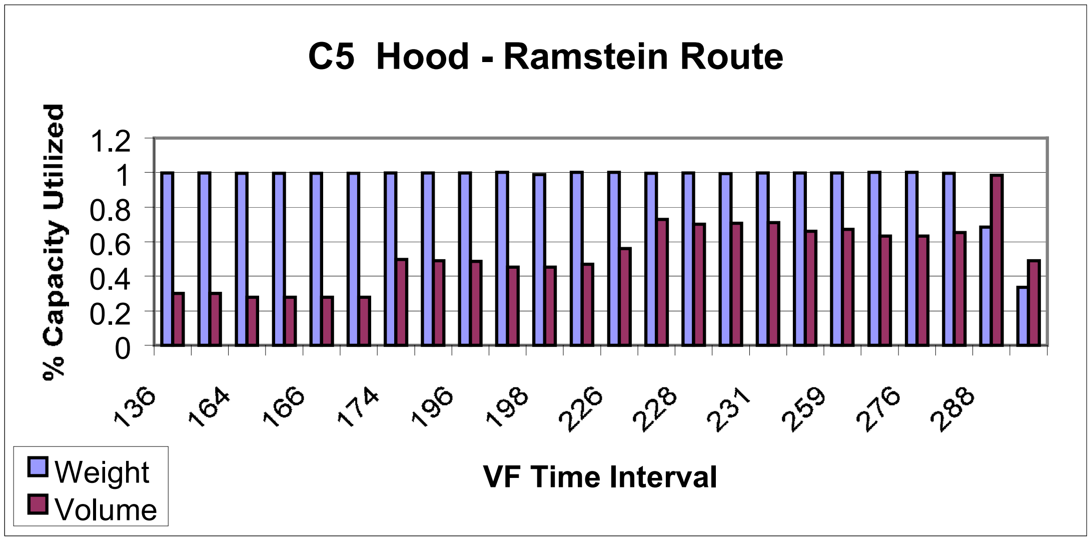


Figure 6.2 Model B C5 Asset Utilization Information

The asset utilization information for C17's operating from Ft Campbell AAF to Dhahran in a Model B scenario is shown in Figure 6.3. Note that every mission, except the last one, was completely filled with respect to either weight or volume.

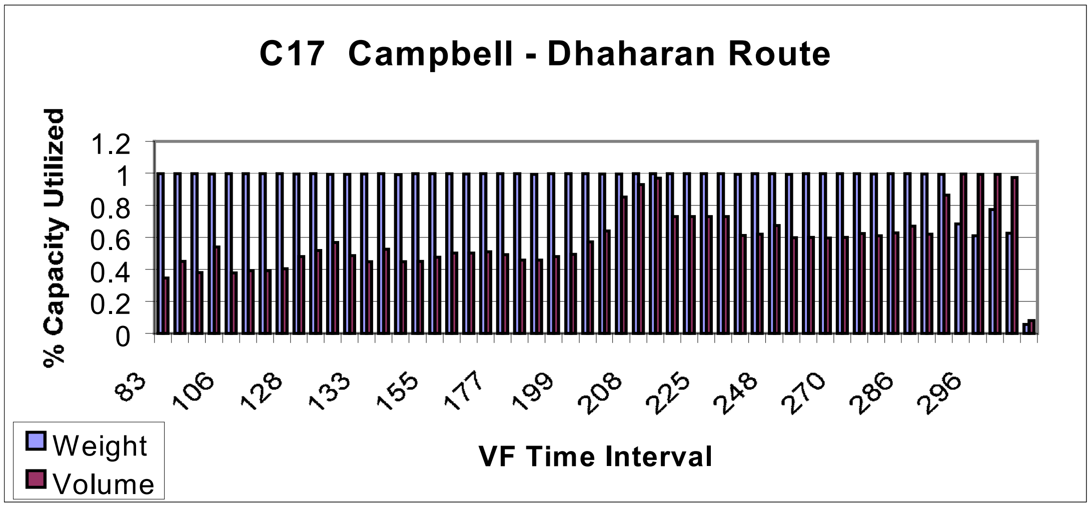


Figure 6.3 Model B C17 Asset Utilization Information

Rail utilization from Ft Campbell to the Beaumont seaport is shown in Figure 6.4. Each mission, except the last is fully utilized with respect to either weight or volume capacity.

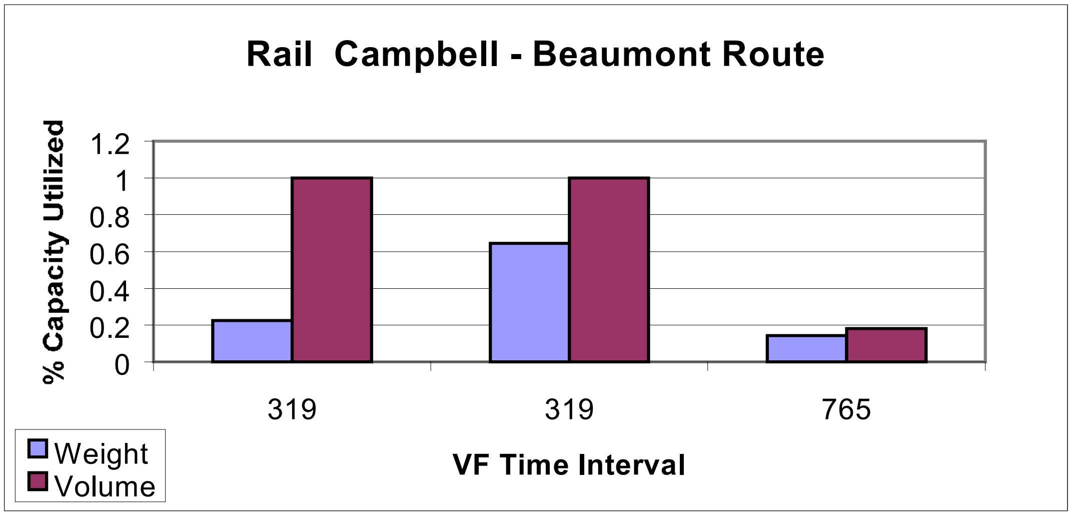


Figure 6.4 Model C Rail Asset Utilization Information

Finally, the FSS utilization for ships operating between Beaumont and Dammam is shown in Figure 6.5. This chart shows that each FSS mission utilized below 50% of the available weight capacity while the area capacity is fully utilized, except the last one.

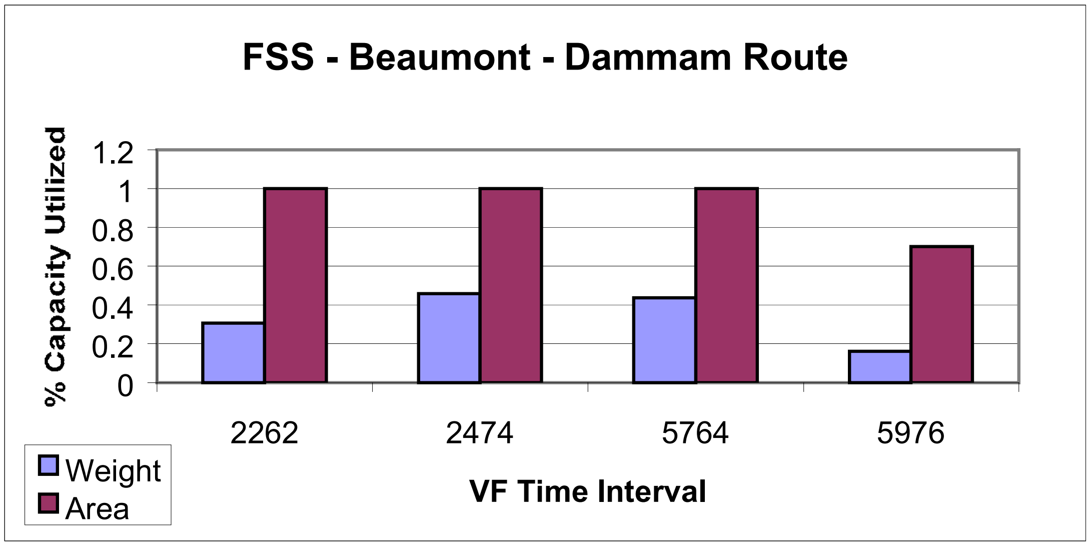


Figure 6.5 Model C FSS Asset Utilization Information

These various graphs represent typical asset utilization performance achieved by all transportation assets in the deployment scenario as a result of DSAT's deployment schedule. These graphs indicate that DSAT, even in complex multi-mode deployments, creates deployment schedules which effectively utilize all transportation assets.

6.2 DSAT Comparison to Different Lower Bound Tools

This section will concentrate on determining how well DSAT schedules equipment to meet its due date. To do this, DSAT's schedule results for C_{max} and L_{max} will be compared to similar results from different planning tools. The first comparison will be made between DSAT and DANTE. DANTE will establish a lower bound for the deployment closure time that can be compared to DSAT's closure time. The next comparisons will be made between DSAT and *COMFLOW*. Each comparison using

COMFLOW will showcase a different method of computing port throughput rates. Initially, *COMFLOW* will use the same method used by DANTE, this will establish a lower bound for both deployment closure time and maximum lateness. These values will be compared to DSAT's results and will show any differences between DANTE's and *COMFLOW*'s closure time estimation. Next, *COMFLOW* will use both the alternate port throughput rate computation methods described in Chapter 4. These comparisons will help to establish the effect that port throughput rates have on the effectiveness of a lower bound. The deployment models and scenarios developed in Chapter 5 are used for each comparison.

6.2.1 DSAT and DANTE

All Model A scenarios are easily converted into an equivalent DANTE problem. All assets operating on the Campbell to Ramstein route are Strategic Air Assets. Those assets operating on the Ramstein to Dhahran route are Theater Air Assets, and finally, those assets operating on the Campbell to Dhahran route are Through Air Assets. The distances between the ports remain the same, as do the speeds of the assets. Because the load times, unload times and capacities of each asset must be integer values in DANTE, they are rounded as necessary. The largest air asset that can be accommodated at each port is the C5. Therefore, all ports have the same throughput rate, expressed as STONs per day. The MOG for each port is 5 so that each port's throughput rate is 1,731 STONs per day. The DANTE input data, common to all Model A scenarios, is in Table 6.5. The information specific to each scenario is in Table 6.6.

Table 6.5 Model A DANTE Standard Input

Strategic Air Distance (nm)	3958
Thru Air Distance (nm)	6330
Theater Air Distance (nm)	2402
Speed C17 (knots)	410
Speed C5 (knots)	409
Speed C130 (knots)	270
Capacity C17 (stons)	45
Capacity C5 (stons)	61
Capacity C130 (stons)	12
Load time C17 (hrs)	2
Load time C5 (hrs)	4
Load time C130 (hrs)	2
Unload time C17 (hrs)	2
Unload time C5 (hrs)	3
Unload time C130 (hrs)	2
CONUS AP capacity (stons/day)	1731
Strategic AP capacity (stons/day)	1731
Theater AP capacity (stons/day)	1731

Table 6.6 Model A Scenario Specific Data

Assets	Scenario 1	Scenario 2	Scenario 3	Scenario 4
# Strategic C17	5	50	60	70
# Strategic C5	5	50	60	70
# Thru C17	5	50	60	70
# Thru C5	5	50	60	70
# Theater C130	15	90	90	100
Lift (# stons)	2,233	11,134	21,095	32,210

The following figure, Figure 6.6, displays both DSAT's closure time and DANTE's closure time for all Model A scenarios.

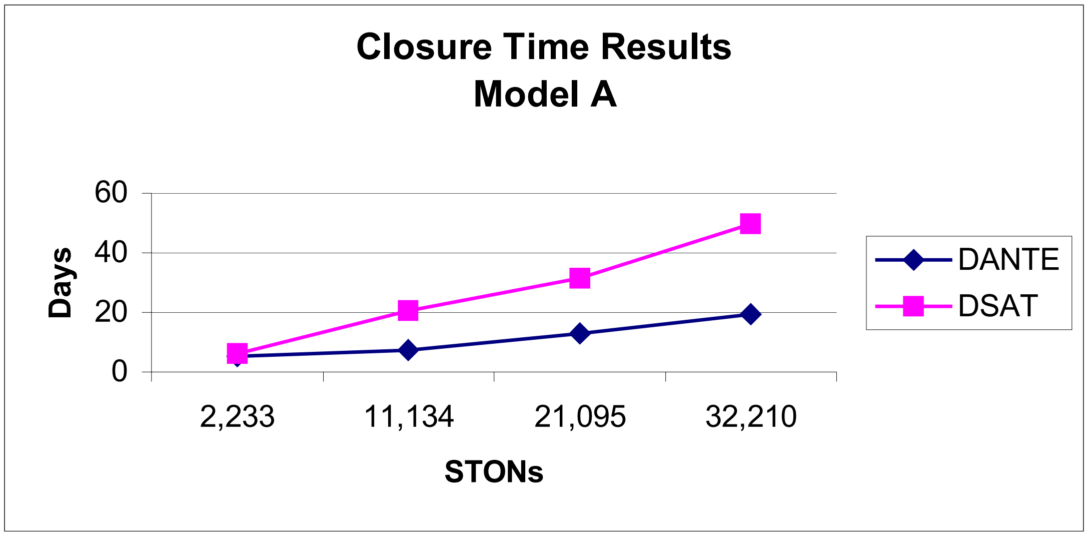


Figure 6.6 DSAT and DANTE Closure Times; Model A

Converting the Model B scenarios to equivalent DANTE problems was slightly more complicated. There are two strategic air distances (Ft. Campbell to Ramstein and Ft. Hood to Ramstein) as well as two throughput air distances (Campbell to Dhahran and Hood to Dhahran). The larger distance was used in both cases. The number of strategic air assets, by type, was the sum of all strategic air assets, by type, operating from Campbell and Hood. The throughput air assets were similarly combined. The two CONUS airport capacities were merged into one port with a capacity equal to the sum of both. The DANTE standard input data and scenario specific data are shown in Tables 6.7 and 6.8, respectively.

Table 6.7 Model B DANTE Standard Input

Strategic Air Distance (nm)	4546
Thru Air Distance (nm)	6895
Theater Air Distance (nm)	2402
Speed C17 (knots)	410
Speed C5 (knots)	409
Speed C130 (knots)	270
Capacity C17 (stons)	45
Capacity C5 (stons)	61
Capacity C130 (stons)	12
Load time C17 (hrs)	2
Load time C5 (hrs)	4
Load time C130 (hrs)	2
Unload time C17 (hrs)	2
Unload time C5 (hrs)	3
Unload time C130 (hrs)	2
CONUS AP capacity (stons/day)	3462
Strategic AP capacity (stons/day)	1731
Theater AP capacity (stons/day)	1731

Table 6.8 Model B Scenario Specific Data

Assets	Scenario 1	Scenario 2	Scenario 3	Scenario 4
# Strategic C17	100	140	170	180
# Strategic C5	100	140	170	180
# Thru C17	100	140	170	180
# Thru C5	100	140	170	180
# Theater C130	100	100	100	100
Lift (# stons)	12,151	59,013	62,925	106,224

Figure 6.7 shows the closure time results for both DSAT and DANTE for the Model B scenarios.

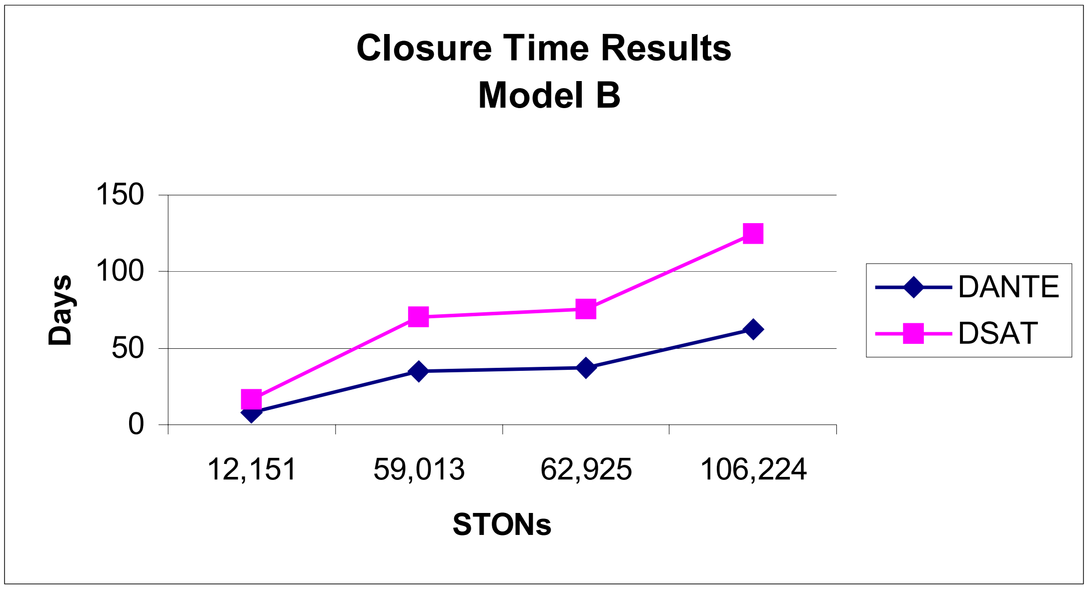


Figure 6.7 DSAT and DANTE Closure Times, Model B

Converting the Model C scenarios into equivalent DANTE problems followed a similar procedure regarding sea port consolidation. However, there were several key differences between the DANTE and the Model C network for rail and sea movement. DANTE does not model CONUS rail movement. Instead, the origin node or "Fort" is connected to the CONUS seaport by a network connector with no time or capacity restrictions. The strategic seaport is connected to the "Fight" or network terminal node by theater rail (or highway or waterway routes) which have both time and capacity restrictions based on the distance. The Model C network, as far as rail and sea movement, is almost completely structurally reversed. The origin nodes or "forts" are connected to various CONUS seaports, each connection with associated time and capacity restrictions based on the distance. The strategic seaport node is connected to the network terminal node by a network connector with no time or capacity restrictions. To model the theater rail movement in DANTE, Model C's longest CONUS rail distance was used to determine the time and capacity restrictions. All CONUS train assets were consolidated into theater rail assets. To model the ship movement between the CONUS and strategic seaports, the longest Model C sea distance was used. All ship assets were consolidated onto this route and the resulting time and capacity restrictions were used. DANTE's standard input data for the Model C scenarios is shown in Table 6.9. This is in addition to the standard input air data shown in Table 6.7.

Table 6.9 Model C DANTE Standard Input

Strategic Sea Distance (nm)	8583
Speed FSS	27
Load time FSS (hrs)	48
Unload time FSS (hrs)	48
Capacity FSS (stons)	28560
CONUS SP capacity (stons/day)	57120
Strategic SP capacity (stons/day)	57120
Theater rail time (days)	4
Theater rail capacity (stons/day)	4953

The Model C scenario specific data is shown in Table 6.10.

Table 6.10 Model C Scenario Specific Input

Assets	Scenario 1	Scenario 2	Scenario 3	Scenario 4
# Strategic C17	80	160	120	180
# Strategic C5	80	160	120	180
# Thru C17	80	160	120	180
# Thru C5	80	160	120	180
# Theater C130	100	100	100	100
# Strategic FSS	4	4	4	4
# (Theater) Trains	4	4	4	4
Lift (# stons)	20,582	36,215	77,371	129,587

Figure 6.8 shows the results in closure time for the Model C scenarios.

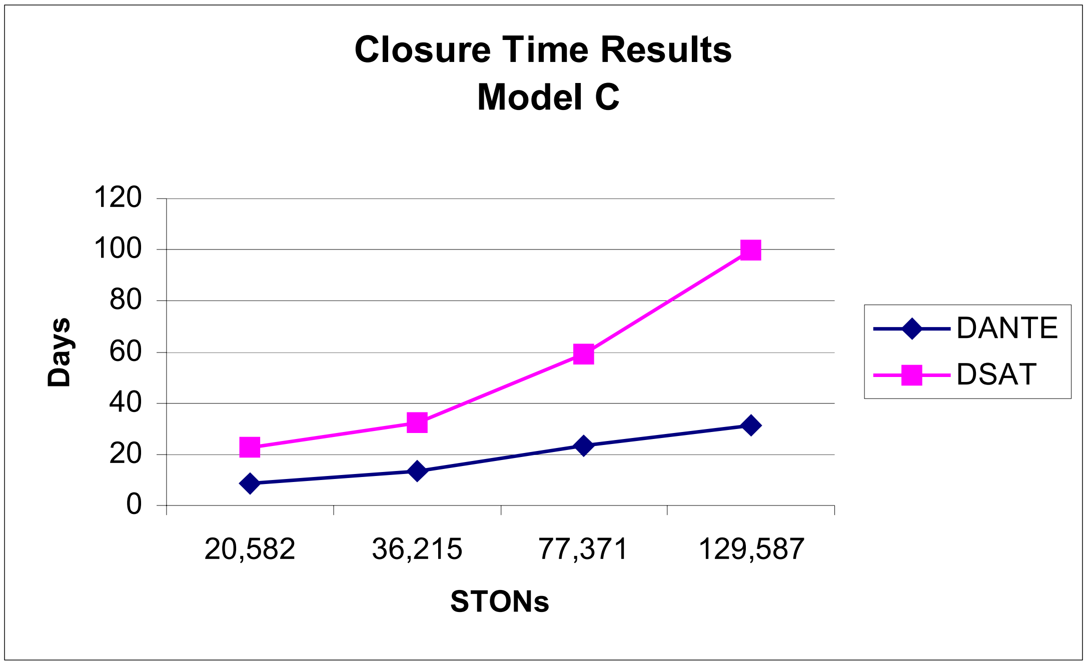


Figure 6.8 DSAT and DANTE Closure Times; Model C

The results demonstrate that DANTE does indeed provide a lower bound for the deployment closure time, LB(C_{max}), for a given deployment scenario. In most cases, however, this lower bound is weak. *COMFLOW* was developed to provide a lower bound for C_{max} and for the maximum lateness, L_{max}.

6.2.2 DSAT, DANTE and COMFLOW

DSAT's results in C_{max} and L_{max} for all models and scenarios will be compared to different *COMFLOW* results for the same scenarios. The initial comparison, *COMFLOW*(1), will involve results based on using the same port throughput rate computed for the previous DANTE analysis. The *COMFLOW*(1) analysis is designed to show any improvement in the estimation of LB(C_{max}) gained over DANTE's estimation. An iterative application of *COMFLOW*(1) will also provide a value for LB(L_{max}).

The subsequent comparisons involve *COMFLOW* analyses using the proposed port throughput computation methods described in Chapter 5. These analyses focus on further improving the LB(C_{max}) and LB(L_{max}) values. *COMFLOW*(2) uses a port throughput rate based on the maximum individual asset throughput rate. At all airports the maximum asset throughput rate was that of the C17 aircraft at 300 STONs per period. The seaport asset throughput rates were based on the FSS throughput rate and the number of assets operating at the individual ports. The Beaumont and Savannah ports each had two ships operating there, yielding a port throughput rate of 7,140 STONs per period. The Dammam port has a total of four ships operating there for a port throughput rate of 14,280 STONs per period. As mentioned in Chapter 3, using this port throughput rate in

DANTE will provide a lowerbound for C_{max}. This same rate used in *COMFLOW* will provide a lowerbound for L_{max} and a corresponding conditional lowerbound for C_{max}.

COMFLOW(3) uses a weighted average port throughput rate based on the size and composition of the deployment asset fleet operating at each port. The computed port throughput rate for each port in each scenario is identified in Table 6.11.

Table 6.11 Computed Port Throughput Rates (STONs per Period)

Model	***Scenario***	***Campbell***	***Hood***	***Ramstein***	***Dhahran***	***Savannah***	***Beaumont***	***Dammam***
A	1	274.589	▨	178.744	166.929	▨	▨	▨
	2	274.589	▨	199.762	186.176	▨	▨	▨
	3	274.589	▨	207.598	193.736	▨	▨	▨
	4	274.589	▨	209.696	195.798	▨	▨	▨
B	1	274.589	274.436	222.916	209.732	▨	▨	▨
	2	274.589	274.436	236.275	223.931	▨	▨	▨
	3	274.589	274.436	243.338	231.732	▨	▨	▨
	4	274.589	274.436	245.311	233.949	▨	▨	▨
C	1	274.589	274.436	213.515	200.151	7140	7140	14280
	2	274.589	274.436	241.19	229.338	7140	7140	14280
	3	274.589	274.436	230.309	217.503	7140	7140	14280
	4	274.589	274.436	245.311	233.949	7140	7140	14280

Using this port throughput rate in DANTE or *COMFLOW* does not guarantee establishing a lowerbound for C_{max} or L_{max} since this method could underestimate the actual port throughput rate in early periods.

The final comparison, *COMFLOW*(4), uses the same airport throughput rates as identified in the table above. The sea port throughput rates, however, are based on the assumption that only 50% of the FSS weight capacity is used. This assumption yields a throughput rate of 3,570 STONs per period for Beaumont and Savannah and a throughput rate of 7,140 STONs per period for Dammam.

6.2.2.1 Model A Analysis

The input parameters for *COMFLOW* require individual commodity movement requirements and deployment horizon lengths in addition to the number of transportation assets assigned to each route. Table 6.12 shows the scenario specific input parameters for all Model A scenarios. These parameters are in addition to the standard asset operating characteristics identified in Chapter 5 and the computed port throughput rates.

Table 6.12 Model A COMFLOW Input Parameters

Scenario	***com id***	***stons***	***due***	***C5***	***C17***	***C130***	***T***
1	1	1,265	5	5	5	15	10
	2	968	6				
2	1	1,265	5	50	50	90	25
	2	5,330	6				
	3	4,539	7				
3	1	4,049	15	60	60	90	45
	2	9,605	17				
	3	2,219	19				
	4	5,222	21				
4	1	4,049	15	70	70	100	60
	2	9,605	19				
	3	7,442	23				
	4	5,784	27				
	5	5,330	31				

Based on the input parameters, the *COMFLOW* analysis resulted in a linear program (LP) characterized by a number of variables, constraints, and a solution time. A single application of *COMFLOW* was required to ultimately establish LB(L_{max}) for the Model A scenarios. Table 6.13 shows the resultant LP size, and solution time for each *COMFLOW* iteration and the number of *COMFLOW* iterations needed to establish the lower bound for maximum lateness.

Table 6.13 Model A LP Information

Scenario	***# variables***	***# constraints***	***Runtime***	***# Iterations***
1	920	928	0:00:01	1
2	3,540	2,959	0:00:03	1
3	8,560	6,430	0:00:10	1
4	14,300	10,021	0:00:21	1

The deployment closure time and maximum lateness results (in days) for the different scenarios and models are included in Table 6.14.

Table 6.14 Model A Results: DSAT, DANTE and COMFLOW

	DSAT		DANTE	COMFLOW(1)		COMFLOW(2)		COMFLOW(3)	
Scenario	Closure	Lateness	Closure	Closure	Lateness	Closure	Lateness	Closure	Lateness
1	6.26	0.26	5.25	5.75	-0.25	5.75	-0.25	5.75	-0.25
2	20.53	13.53	7.25	7.25	0.25	10.25	3.25	15.75	8.75
3	31.33	10.33	13.00	13.00	-8.00	18.50	-2.50	28.00	7.00
4	49.72	18.72	19.50	19.50	-10.00	27.75	-3.25	42.00	11.00

Note that in most instances, the closure time results for DANTE and *COMFLOW*(1) were equivalent. *COMFLOW*(1) provided a slightly better closure time estimate for the first scenario because, for this instance only, the port throughput rate was not the most restrictive element in the network. For this instance, the asset movement arcs with their associated capacities were the limiting factor. DANTE's movement arc capacity was computed based on an integer approximation of the roundtrip time while *COMFLOW* computed the movement arc capacity based on the computed roundtrip time and included maintenance sessions. The models' results are graphically presented in the following figures. Figure 6.9 shows the deployment closure time results established by DSAT, DANTE and each *COMFLOW* version. Note that the *COMFLOW* established LB(C_{max}) is actually a conditional value; it is the LB(C_{max}) given that LB(L_{max}) is minimized. Figure 6.10 shows the maximum lateness results established by DSAT and each *COMFLOW* version.

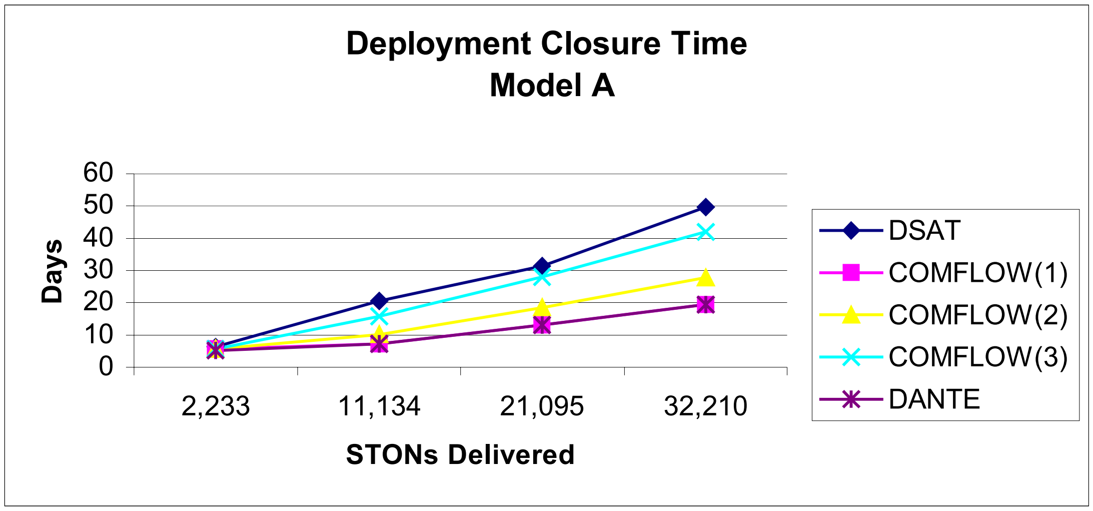


Figure 6.9 Model A Deployment Closure Time Results

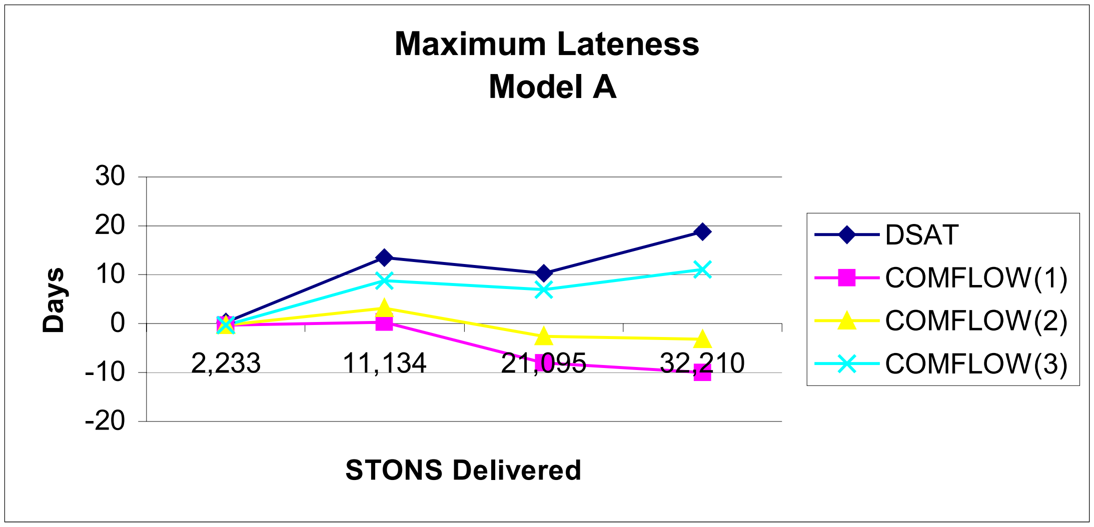


Figure 6.10 Model A Maximum Lateness Results

The different port throughput computation methods greatly improved the effectiveness of the lower bound estimate for C_{max} and L_{max}. It also appears that DSAT quite effectively schedules these small scale deployments in order to meet established delivery dates.

6.2.2.2 Model B Analysis

The Model B scenario specific input parameters for the different *COMFLOW* versions are contained in Table 6.15. The resultant LP size, solution time for each *COMFLOW* iteration and the required number of iterations to establish the lower bound for maximum lateness are contained in Table 6.16.

Table 6.15 Model B COMFLOW Input Parameters

Scenario	*Origin*	*Com Id*	*Stons*	*Due*	*C5*	*C17*	*C130*	*T*
1	C	1	2,234	5	50	50	100	18
	H	2	9,917	10				
2	C	1	4,469	20	70	70	100	70
	C	2	10,291	25				
	H	3	19,834	25				
	H	4	24,419	30				
3	C	1	8,900	25	85	85	100	100
	C	2	5,784	35				
	H	3	24,752	20				
	H	4	15,534	25				
	H	5	7,955	30				
4	C	1	7,555	20	90	90	100	120
	C	2	7,207	30				
	C	3	17,448	40				
	H	4	24,752	25				
	H	5	22,630	35				
	H	6	26,632	50				

Table 6.16 Model B LP Information

Scenario	*# Variables*	*# Constraints*	*Runtime*	*# Iterations*
1	1,681	2,124	0:00:02	1
2	13,346	11,702	0:00:15	1
3	23,879	19,129	0:00:40	1
4	34,419	25,840	0:01:29	1

The deployment closure time and maximum lateness results (in days) for the different scenarios and models are included in Table 6.17.

Table 6.17 Model B Results: DSAT, DANTE and COMFLOW

	DSAT		DANTE	COMFLOW(1)		COMFLOW(2)		COMFLOW(3)	
Scenario	Closure	Lateness	Closure	Closure	Lateness	Closure	Lateness	Closure	Lateness
1	16.70	6.70	8.00	8.00	-2.00	11.00	1.00	15.25	5.25
2	70.38	40.38	35.00	35.00	5.00	50.00	20.00	66.75	36.75
3	75.77	40.77	37.25	37.25	4.25	53.25	18.50	68.75	33.75
4	125.08	75.08	62.25	62.25	12.25	89.50	39.50	114.50	64.50

Note that in all scenarios the port throughput rate was the restricting network element. DANTE and *COMFLOW*(1) had identical deployment closure time results. The models' results are graphically presented in the following figures. Figure 6.11 shows the deployment closure time results established by DSAT, DANTE and each *COMFLOW* version. Figure 6.12 shows the maximum lateness results established by DSAT and each *COMFLOW* version.

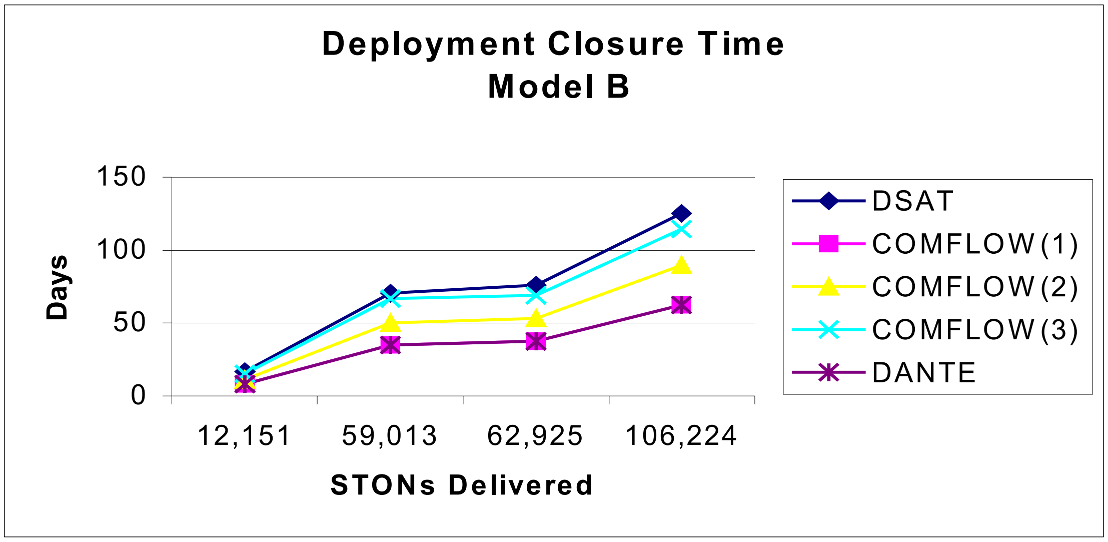


Figure 6.11 Model B Deployment Closure Time

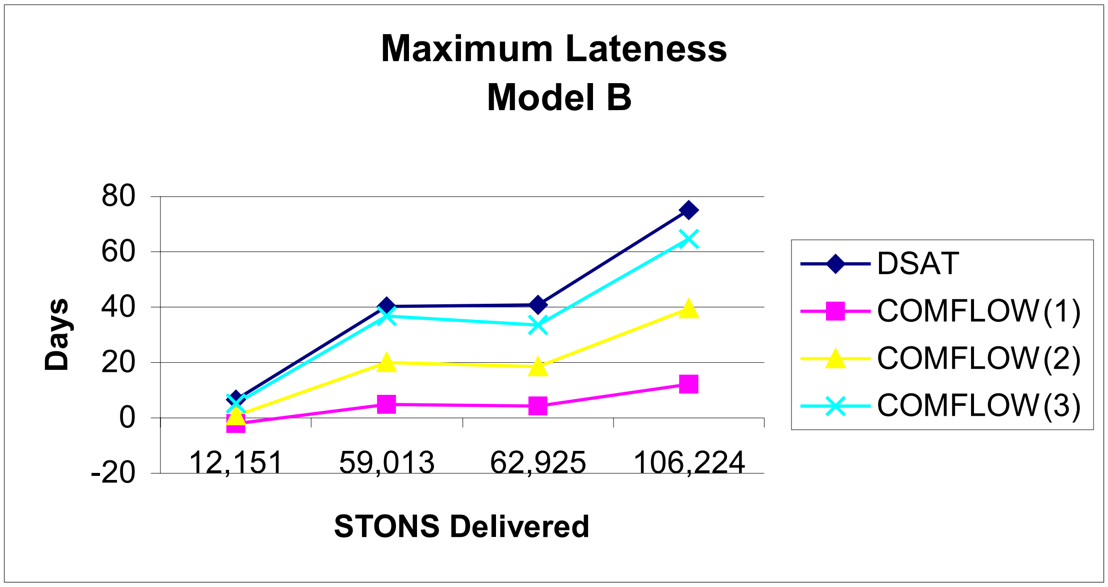


Figure 6.12 Model B Maximum Lateness Results

The different port throughput rate computation methods again provided better lower bound estimates for these deployment scenarios. DSAT appears to effectively schedule these increasingly larger and more complex deployments to meet established due dates.

6.2.2.3 Model C Analysis

The Model C scenario specific input parameters for the different *COMFLOW* versions are contained in Table 6.18. The resultant LP size, solution time for each *COMFLOW* iteration and the required number of iterations to establish the lower bound for maximum lateness are contained in Table 6.19.

Table 6.18 Model C COMFLOW Input Parameters

Scenario	***Origin***	***Com Id***	***Stons***	***Due***	***C5***	***C17***	***C130***	***Train***	***FSS***	***T***
1	C	1	2,234	20	40	40	100	2	2	20
	H	2	18,348	25						
2	C	1	5,784	20	80	80	100	2	2	30
	C	2	3,570	25						
	H	3	8,365	25						
	H	4	8,342	35						
	H	5	10,154	40						
3	C	1	3,797	20	60	60	100	2	2	60
	C	2	8,254	25						
	H	3	45,668	25						
	H	4	19,652	30						
4	C	1	14,912	25	90	90	100	2	2	90
	C	2	12,076	35						
	C	3	5,222	40						
	H	4	45,668	35						
	H	5	24,690	45						
	H	6	27,019	55						

Table 6.19 Model C LP Information

				# Iterations			
Scenario	***# variables***	***# constraints***	***Runtime***	***(1)***	***(2)***	***(3)***	***(4)***
1	2,991	3,658	0:00:02	1	1	1	1
2	12,069	9,521	0:00:21	1	15	3	2
3	20,702	17,784	0:00:55	1	1	1	1
4	53,136	42,809	0:06:56	1	3	1	1

The results listed in Table 6.19 highlight an interesting difference between Model C and the previous two deployment models. For both Models A and B, regardless of the version of *COMFLOW* used, the first iteration *COMFLOW* analysis established the LB(L_{max}) estimate. For Model C scenarios, the port throughput rate (different in each *COMFLOW* version) appeared to effect the number of iterations of *COMFLOW* analysis required to establish the LB(L_{max}) estimate. In most cases, the number of iterations required was small. The results for the different models are shown in Table 6.20.

Table 6.20 Model C DSAT, DANTE and COMFLOW Results

	DSAT		DANTE	COMFLOW(1)		COMFLOW(2)		COMFLOW(3)	
Scenario	*Closure*	*Lateness*	*Closure*	*Closure*	*Lateness*	*Closure*	*Lateness*	*Closure*	*Lateness*
1	22.72	-2.28	8.75	12.75	-12.25	18.00	-7.00	21.25	-3.75
2	32.38	-2.28	13.25	20.50	-14.00	22.75	-9.25	23.75	-5.00
3	59.30	29.30	23.25	27.50	-1.00	30.75	1.75	33.00	3.50
4	99.60	44.60	31.25	36.75	-8.25	41.50	-5.25	43.75	-3.50

Note that there are now significant differences between DANTE and *COMFLOW*(1) deployment closure time results. This is due, at least in part, to the differences in the network structures for the different tools. If theater routes (in addition to theater air lift) are included in the DANTE input parameters, cargo flows using strategic air lift can use any of the theater lift routes (rail, highway, waterway or air) to reach the “Fight”. In the model developed for the *COMFLOW* analysis, strategic air lift flows are restricted to theater air lift assets to reach the sink node. The different results are graphically displayed in Figures 6.13 and 6.14. Figure 6.13 shows the deployment closure time results. Figure 6.14 shows the maximum lateness results.

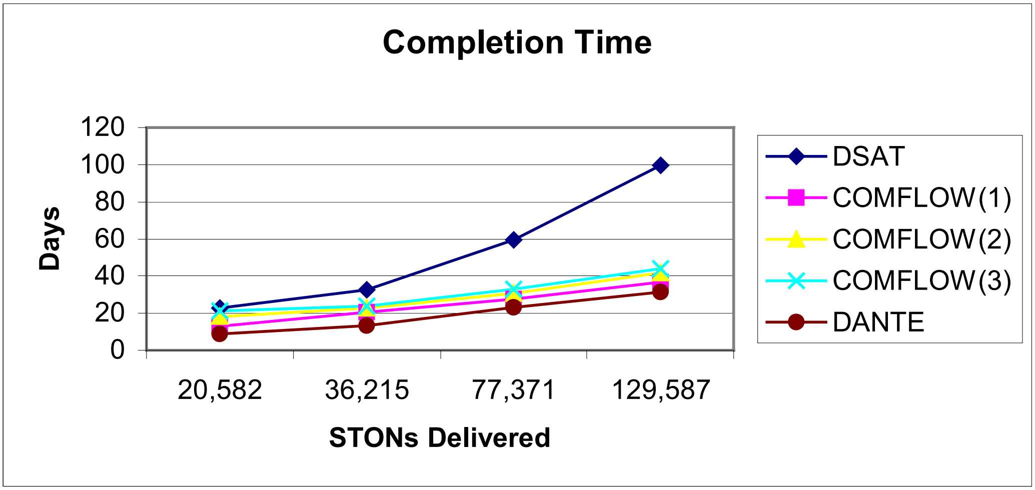


Figure 6.13 Model C Deployment Closure Time Results

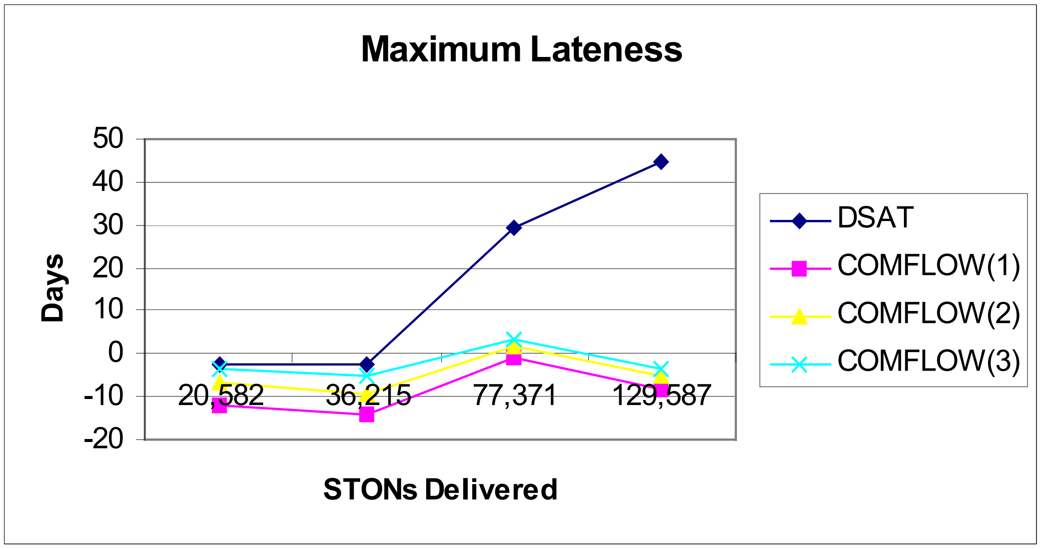


Figure 6.14 Model C Maximum Lateness Results

Although the different port throughput rate computation methods did improve the lower bound estimations for C_{max} and L_{max} for these particular scenarios, the best demonstrated estimate for both is still weak. The weakness of the lower bound estimates is due to the inclusion of ships into the deployment model. One FSS can carry the same weight carried by approximately 465 C5 aircraft (the C5 has the largest weight carrying capacity of all aircraft). Ships, as mentioned previously, rarely carry their capacity in weight. Instead, they carry their capacity in area. An assumption used in both DANTE and *COMFLOW* is that all assets can utilize 100% of their weight carrying capacity. According to historical data gathered by TEA, this is a valid assumption if considering movement by air, rail or highway (trucks, trailer, etc) assets. This is not a valid assumption when considering movement by ship. The final version of *COMFLOW* presented, *COMFLOW*(4), uses the same port throughput computation method as *COMFLOW*(3) with the following two assumptions. First, all rail and air assets utilize 100% of their specific weight carrying capacity. Second, ships, in this case FSS, use only 50% of their weight carrying capacity. The 50% figure was derived from information presented in MTMC_TEA's Deployment Planning Guide. The strategic sealift mission requirements to deploy all identified units, with specified weight requirements, was analyzed to determine the average percentage of FSS weight capacity used for all types of unit movements.

The *COMFLOW*(4) results for the LB(C_{max}) and LB(L_{max}) estimates are identified in Table 6.21. These results are included in the graphical representations in Figures 6.15 and 6.16.

Table 6.21 COMFLOW(4) Results

Scenario	*Closure*	*Lateness*
1	22.00	-3.00
2	26.50	-5.00
3	42.00	12.00
4	60.00	5.25

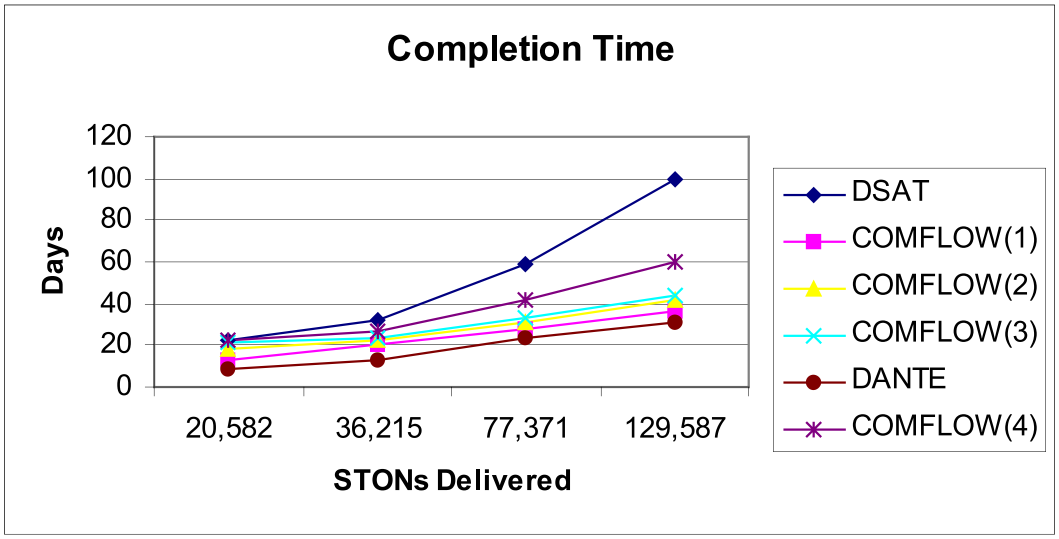


Figure 6.15 Model C Deployment Closure Time Results

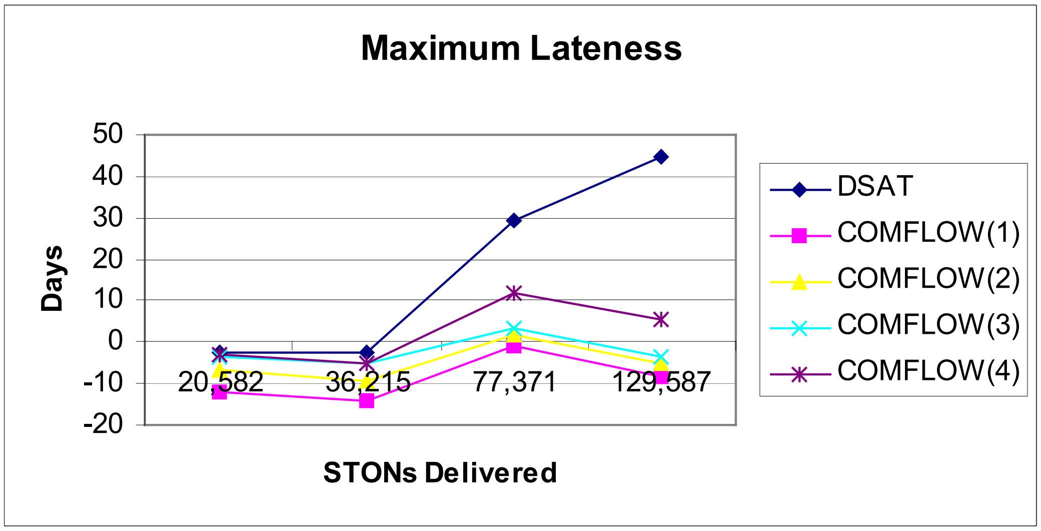


Figure 6.16 Model C Maximum Lateness Results

This last version of COMFLOW does provide a much better estimate for the lower bounds for both C_{max} and L_{max}. The effectiveness of this estimate appears to vary between scenarios. In the first two scenarios, all tools (DSAT, COMFLOW and DANTE) scheduled the deployment utilizing mostly air assets. In the last two scenarios, with their significantly larger packages, all tools used more rail and shipping in the deployment schedule.

In summary, using the weighted average port throughput rate computation method with the assumptions concerning the percentage of asset weight capacity utilized yields the best estimated lower bound values for C_{max} and L_{max}. Although this method does not guarantee a lower bound for L_{max} and a corresponding conditional lower bound for C_{max}, it will provide a very effective estimate for the lower bound values. In all models developed for this research, the actual flow schedule of the optimal solutions followed similar patterns of behavior. In all models, the first period in which flow could be delivered to the final destination was period 4. This flow was delivered by the direct C17 route from Campbell to Dhahran. Dhahran's port throughput rate, based on active routes and the optimal schedule, was 300 STONs in this period. This rate is larger than any value computed for the Dhahran port in Table 6.11. In the next period, period 5 and many subsequent periods, the optimal solution had flow arriving to Dhahran from all possible air routes. The actual port throughput rate, based on active routes and the optimal solution, was the values as identified in Table 6.11. Finally, at some point in time, the amount of equipment at one (eventually both) origin(s) was depleted. The actual Dhahran port throughput rate for this time span, based on active routes and the optimal solution was a value less than the rate shown in Table 6.11. In model C, all equipment sent by rail and ship went to the final destination seaport of Dammam. This port throughput rate was based on the weighted average arrival rate of the FSS ship. Since no other ship with different capacities was used, the Dammam port throughput rate was a constant as represented in Table 6.11. If ships with different capacities were used, the Dammam port throughput rate would most likely change over time as a function of the active routes in a given period of time. The weighted average port throughput rate computation method, assuming all routes are active, appears to establish a good estimate of the average port throughput rate for the duration of the deployment and therefore should provide a good estimate for the lower bound for maximum lateness and deployment closure time. These estimated values appear to be very good for deployments involving only air movement. These estimates might yet be weak when dealing with ship movements.

DSAT generates deployment schedules which quite efficiently utilize all transportation assets, surpassing MTMC_TEA's established asset utilization planning factors. This effective utilization of assets holds true for every model and scenario evaluated. DSAT appears to create very good deployment schedules, in terms of deployment closure time and maximum lateness, for deployments involving air movement. DSAT may or may not perform as well for deployments involving sea movement.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

This research encompassed several distinct objectives. A much-used deployment planning tool, DANTE, was presented in great detail. DANTE models all deployments using a generalized deployment network. A proof, presented in the third chapter, showed that DANTE's minimum cost objective function also minimized the completion time of the deployment, C_{max}.

DANTE's concept of deployment modeling was expanded to include due date information. *COMFLOW* was developed as a multi-commodity variation of DANTE. The commodities are uniquely identified by an origin-due date pair of defined equipment characteristics. An iterative application of *COMFLOW* established the maximum lateness of any commodity in a deployment scenario. Both DANTE and *COMFLOW* are relaxations of the original military deployment problem and therefore can be used to establish lower bounds for C_{max} and L_{max}, respectively.

Finally, this research briefly introduced a new deployment planning tool, DSAT. A key internal aspect of DSAT, the scheduling routine developed from the Virtual Factory, was presented in greater detail. DSAT allows a deployment planner to rapidly model any deployment scenario. DSAT feeds the user constructed deployment scenario to the scheduler routine and creates a schedule designed to minimize L_{max}. Work done in this research yielded measurable improvement to DSAT in terms of reduced deployment closure time and maximum lateness. Results from DSAT's schedules were compared to various standards to evaluate the tool's effectiveness in developing deployment schedules. This analysis showed that DSAT's schedules very effectively utilized all transportation assets throughout the deployment process. DSAT's asset utilization, in terms of the average percentage of weight and area or volume capacity utilized per asset mission, was significantly higher than established and accepted asset utilization planning factors published by MTMC_TEA. DSAT's schedule results for C_{max} and L_{max} were compared to the corresponding lowerbound values established by DANTE and *COMFLOW*. The gap between DSAT's results and *COMFLOW*'s results (for both LB(L_{max}) and LB(C_{max})) was small for deployments involving air assets. This indicated that DSAT generated near optimal deployment schedules for these deployment scenarios. The gap between DSAT and *COMFLOW* is significantly larger for deployment scenarios involving rail and ship movement. This gap could be due to a weak lowerbound or poor DSAT schedule results or a combination of both. Further research needs to focus on reducing the gap between DSAT and *COMFLOW* for deployment scenarios involving rail and ship movement.

7.2 Future Research

Several objectives are planned in furthering this research. A current version of DSAT, compatible with Windows 2000, is on site at MTMC_TEA. Issues and recommendations concerning software usability and scenario storage and retrieval have been identified for resolution. Some additional short term objectives include additional improvements to DSAT which should further reduce the gap between DSAT's and *COMFLOW*'s results for deployment closure time and maximum lateness. Two additional long term objectives include the addition of highway movement scheduling and time phased transportation asset – route reallocation into DSAT.

7.2.1 Short Term Improvements to DSAT

During a recent demonstration of DSAT to members of MTMC_TEA, several issues were presented for inclusion into future versions of DSAT. One requires expansion of the unit database information inherent in DSAT. This database should be expanded to include information concerning prepositioned equipment. This equipment, although strategically located about the world, if included in the deployment package, would eventually compete for space on some of the transportation assets in the deployment network. Another issue was the capability to include intermediate refuel stops. Currently, all selected hubs act as transshipment nodes. Arriving assets simulate the unloading of their cargo and maintenance operations prior to departure for its next cargo load pickup. Assets moving cargo forward from that location simulate the upload of cargo prior to departure for the next location. The addition of intermediate refuel stops would simulate the necessary refueling operations required for longer routes without transloading cargo.

Some additional improvements are specifically planned for the scheduling routine included in the DSAT program. The first involves the creation of additional fake jobs for the rail and sea routes in the deployment network. A single train and ship can both carry much more equipment than multiple planes. Fake jobs serve to create additional asset movement in the schedule. These scheduled asset movements become choices or opportunities for the job-route reallocation heuristic that recreates job routings based on actual asset arrival times. More fake jobs on the rail and ship routes would increase the number of asset movements scheduled and could serve to further reduce the deployment closure time and maximum lateness.

The second improvement concerns the accuracy of the deployment simulation itself. This particular issue is concerned with only the intermediate or transshipment nodes at which vehicle loading and unloading operations could occur simultaneously at the same port. An example is the Ramstein Airbase described in the three deployment models used in this paper. At this node, planes arrive from other locations and unload their cargo before returning to their origins. Other planes load cargo at Ramstein before departing for their destination. As the simulation starts, all planes operating at the ISB are hovering in the air so that loaded planes arriving at the ISB from other locations can land and unload their cargo before returning to their origins. Empty assets always "hover" to deconflict the competition for space at the ports. This ensures that loaded assets always have priority to "land" at the ports and unload their cargo. The newly arrived cargo is immediately loaded onto the waiting hovering planes. Once these hovering planes are given access to the ISB to land, they then simulate landing, maintenance and loading operations prior to departing for their next destination. The proposed change would ensure that these hovering planes were not loaded until they had actually landed and undergone maintenance. This change would improve the quality of the simulation and could affect the results regarding deployment closure time and maximum lateness. If the planes operating out of the ISB were loaded during the simulated load time, then it could be more fully utilized since more equipment would have previously been received at the ISB.

7.2.2 Highway Movement Scheduling

Not all units or equipment deploy by rail to a designated seaport of embarkation. Some units or portions of units self deploy or are delivered to the SPOE by tractor-trailer movement over the highways. For example, the 101st Air Assault Division would self deploy or fly all helicopters to the seaport. Military transportation units would carry other unit equipment as cargo in their trucks while they deployed by highway convoy to the SPOE.

If the seaport is relatively close to the base, highway movement is usually much less time consuming and costly than rail movement to the same location. The addition of highway movement scheduling into DSAT could reduce the amount of time needed to deploy units and equipment to the SPOE.

7.2.3 Transportation Asset – Route Reallocation

Currently, DSAT schedules deployment over a static deployment network. As mentioned previously, a static deployment network is one in which the transportation asset to route allocation is fixed for the duration of the deployment. This fixed allocation does not accurately depict the usual deployment process. Transportation assets are rigidly managed and tend to be used where they are most needed. A surplus of transportation assets on a route with little or nothing to transport would be reassigned to alternate routes with fewer assets and more equipment to move.

Modifying DSAT to allow transportation asset – route reallocation over time would provide the capability to simulate deployment utilizing a dynamic deployment network. This would result in a more accurate estimation of the deployment closure time and the maximum lateness. These “dynamic” DSAT results could be compared to results from the Dynamic Transportation Asset Allocation Model developed by Trainor [34]. This model determines the optimal allocation over time of available transportation assets to routes in a deployment network. The dynamic allocation was designed to minimize the deployment closure time of a deployment package.

In conclusion, several areas within the DSAT program could be modified. These modifications could enhance the effectiveness and possibly the accuracy of DSAT. Further experimentation is needed to determine the effects of the proposed modifications.

List of References

[1] Aronson, J.E., and Chen B.D., “A Forward Network Simplex Algorithm for Solving Multiperiod Network Flow Problems,” *Naval Research Logistics Quarterly*, 33, pp. 445 - 467, 1986.

[2] Aronson, J.E., “The Multiperiod Assignment Problem: A multicommodity Network Flow Model and Specialized Branch and Bound Algorithm”, *European Journal of Operations Research*, 23, pp. 367 - 381, 1986.

[3] Assad, A.A., “Multicommodity Network Flows – A Survey”, *Networks*, 8, pp. 37 – 91, 1978.

[4] Baker, K.R., and Scudder, G.D., “Sequencing with Earliness and Tardiness Penalties: A Review”, *Operations Research*, 38, pp. 22 – 35, 1990.

[5] Baker, K.R., and Su, Z., “Sequencing with Due-Dates and Early Start Times to Minimize Maximum Tardiness,” *Naval Research Logistics Quarterly,* 21(1), pp. 171 - 176, 1974.

[6] Bratley, P., Florian, M., and Robillard, P., “Scheduling with Earliest Start and Due Date Constraints on Multiple Machines,” *Naval Research Logistics Quarterly,* 22(1), pp. 165 - 173, 1975.

[7] Bratley, P., Florian, M., and Robillard, P., “On Sequencing with Earliest Start and Due Date with Application to Computing Bounds for the ($n/m/G/F_{max}$) Problem,” *Naval Research Logistics Quarterly,* 20(1), pp. 57 - 67, 1973.

[8] Brucker, P., Jurisch, B., “A New Lower Bound for the Job-Shop Scheduling Problem”, *European Journal of Operational Research*, 64, pp. 156 – 167, 1993.

[9] Cho, Y. and Sahni, S, “Preemptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops”, *Opns. Res*, 29(3), pp. 511 – 522, 1981.

[10] Conway, R.W., “Priority Dispatching and Job Lateness in a Job Shop,” *The Journal of Industrial Engineering*, 16, pp.228 – 237, 1965.

[11] Crainic, T.G., Gendreau, M. Soriano, P., and Toulouse, M., “A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements”, *Annals of Operations Research*, 41, pp. 359 – 383, 1993.

[12] Crainic, T.G., and Rousseau, J., “Multicommodity, Multimode Freight Transportation: A General Modeling and Algorithmic Framework for the Service Network Design Problem,” *Transpn Res.* –B, 20B, pp. 205 - 242, 1986.

[13] Crainic, T.G., Delorme, L., and Dejax, Pierre, “A Branch-and-bound Method for Multicommodity Location with Balancing Requirements”, *European Journal of Operations Research*, 65, pp. 368 – 382, 1993.

[14] Fisher, M.L., Tang, B., and Zheng, Z., “A Network Flow Based Heuristic for Bulk Pickup and Delivery Routing,” *Transpn Sci.*, 29(1), pp. 45 - 55, 1995.

[15] Hodgson, T.J., Cormier, D., Weintraub, A.J., and Zozom, Jr., A., “Satisfying Due Dates in Large Job Shops,” *Management Sci.* 44(10), pp.1442 - 1446, 1998.

[16] Hodgson, T.J., King, R.E., Thoney, K., Stanislaw, N., Weintraub, A.J., and Zozom, Jr., A., “On satisfying due-dates in large job shops: idle time insertion,” *IIE Transactions,* 32, pp. 177 - 180, 2000.

[17] Hodgson, T.J., Spivey, O., Trainor, T., Williams, M.K., “The Deployment Analysis Network Tool Extended (DANTE),” Military Traffic Management Command, Transportation Engineering Agency.

[18] Horn, W.A., “Some Simple Scheduling Algorithms,” *Naval Research Logistics Quarterly,* 22(1), pp. 177 - 185, 1974.

[19] Iakovou, El, Christos, D., Li, H., Ip, C., and Yudhbir, L., “A Maritime Global Route Planning Model for Hazardous Materials Transportation,” *Transpn Sci.*, 33(1), pp. 34 - 48, 1999.

[20] Kim, D., Barnhart, C., Ware, K. and Reinhardt, G., “Multimodal Express Package Delivery: A Service Network Design Application,” *Transpn Sci*, 33(2), pp. 391 - 407, 1999.

[21] Lamar, B.W., Sheffi, Y., and Powell, W.B., “A Capacity Improvement Lower Bound for Fixed Charge Network Design Problems”, *Opns Res*, 38(4), pp. 704 – 710, 1990.

[22] McMahon, G., and Florian, M., “On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness,” *Opns Res.,* 23(3), pp. 475 - 482, 1975.

[23] Military Traffic Management Command, Transportation Engineering Agency (1997). “Deployment Planning Guide: Transportation Assets Required for Deployment”, *MTMCTEA Reference 97-700-5*, 2.

[24] MTMCTEA. (2000). Modeling and Simulation [Online]. Available: http:\\www.tea.army.mil\dpe\modeling.html.

[25] Oak Ridge National Laboratories, (2000). AMC Deployment Analysis System (ADANS) [Online]. Available: http:\\www-cta.ornl.gov\Research\dtl\hist7.html.

[26] Oak Ridge National Laboratories, (2000). Defense Transportation Programs [Online]. Available: http:\\www.ornl.gov\~jkm\ORTRAN\defense.html.

[27] Pinedo, M., *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, New Jersey, pp. 32 – 37, 1995.

[28] Rappoport, H.K., Levy, L.Sl, Golden, B.L., and Feshbach, D.S., "Estimating Loads of Aircraft in Planning for the Military Airlift Command," *Interfaces*, 21(4), pp. 63-78, 1991.

[29] Rathi, A.K., Church, R.L., and Solanki, R.S., "A Macro Level Analysis of the Airlift Deployment Problem," *Computers Ops Res.*, 19(8), pp.731-742, 1992.

[30] Schutten, J.M.J., "Practical Job Shop Scheduling", *Annals of Operations Research*, 83, pp. 161-177, 1998.

[31] Sklar, M. G., Armstrong, R. D., and Samn, S., "Heuristics for Scheduling Aircraft and Crew during Airlift Operations," *Transpn Sci.,* 24(1), pp.63-76, 1990.

[32] Solanki, R.S. and Southworth, F., "An Execution Planning Algorithm for Military Airlift," *Interfaces,* 21(4), pp. 121-131, 1991.

[33] Thoney, K.A, "Simultaneous Plant and Supply Chain Scheduling," Unpublished Dissertation, Department of Industrial Engineering, North Carolina State University, 2000.

[34] Trainor, T., "On Scheduling Military Deployments" Unpublished Dissertation, Department of Industrial Engineering, North Carolina State University, 2001.

[35] Weintraub, A., Cormier, D., Hodgson, T., King, R., Wilson, J., and Zozom, A., "Scheduling with alternatives: a link between process planning and scheduling," *IIE Transactions,* 31, pp.1093-1102, 1999.