The Dissertation Committee for August G. Roesener Certifies that this is the approved version of the following dissertation:

AN ADVANCED TABU SEARCH APPROACH TO THE AIRLIFT LOADING PROBLEM

Committee:

J. Wesley Barnes, Supervisor

John J. Hasenbein

Erhan Kutanoglu

James T. Moore

David Van Veldhuizen

AN ADVANCED TABU SEARCH APPROACH TO THE AIRLIFT LOADING PROBLEM

by

August G. Roesener, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2006

Dedication

To Marsha. Thanks for everything.

Acknowledgements

I would like to thank my advisor, Dr. J. Wesley Barnes, for his support, encouragement, wisdom, and most importantly patience. Without him, this research would never have achieved anything of value. Thanks to Lt. Col. Dave Van Veldhuizen who, among many other things, provided invaluable information about AMC which greatly strengthened the credibility of this research. I would like to thank Dr. Jim Moore for assisting me with finding this research topic and for his help throughout this research. To Dr. John Hasenbein and Dr. Erhan Kutanoglu, thanks for taking time out of your busy schedules to serve on my committee.

To the incredible folks who are or were at Eglin AFB, you are awesome! Thanks to Col. Pete Vanden Bosch who saw something in me that I did not (and still do not) see. Without your advice and recommendations, I would not be here today. Greg Hutto, thanks for your mentorship and guidance. You arm people with the knowledge and confidence required to *properly* get the job done.

Finally, I would like to thank my wife who put up with me for these 3 years. I am sorry that I have not been able to give you the attention you deserve. Thanks for the encouragement and support!

AN ADVANCED TABU SEARCH APPROACH TO THE AIRLIFT LOADING PROBLEM

Publication No._____________

August G. Roesener, Ph.D.

The University of Texas at Austin, 2006

Supervisor: J. Wesley Barnes

This dissertation details an algorithm to solve the Airlift Loading Problem (ALP). Given a set of cargo to be transported from an aerial port of embarkation to one or more aerial ports of debarkation, the ALP seeks to *pack* the cargo items onto pallets (if necessary), *partition* the set of cargo items into aircraft loads, *select* an efficient and effective set of aircraft from available aircraft, and to *place* the cargo in allowable positions on those aircraft. The ALP differs from most partitioning and packing problems described in the literature because, in addition to spatial constraints, factors such as allowable cabin load, balance, and temporal restrictions on cargo loading availability and cargo delivery requirements must be considered. While classical methods would be forced to attack such problems in a hierarchical fashion by solving a sequence of related subproblems, this research develops an algorithm to simultaneously solve the combined problem by employing an advanced tabu search approach.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Force projection comprises a large portion of current United States (US) military activities. According to Wikipedia.com, *force projection* is:

> The ability of a nation to apply all or some of its elements of national power - political, economic, informational, or military - to rapidly and effectively deploy and sustain forces in and from multiple dispersed locations to respond to crises, to contribute to deterrence, and to enhance regional stability.
>
> (http://en.wikipedia.org/wiki/Force_projection)

The transportation of the personnel and equipment involved in force projection lies within the responsibility of the United States Transportation Command (USTRANSCOM). USTRANSCOM is composed of three transportation component commands: Air Mobility Command (AMC), Military Sealift Command (MSC), and Surface Deployment and Distribution Command (SDDC).

AMC has the responsibility of providing airlift to US armed forces wherever they are needed. AMC's air fleet provides swift response as an element of US global reach. During an average week, USTRANSCOM and its component commands operate in 75 percent of the world's countries and conduct more than 1,900 air missions (http://www.transcom.mil/organization.cfm).

The importance of airlift is demonstrated by its definition in the Air Force Doctrine Document 1: *Air Force Basic Doctrine*:

> Airlift is the transportation of personnel and material through the air, which can be applied across the entire range of military operations to achieve or support objectives and can achieve tactical through strategic effects. Airlift provides rapid and flexible mobility options that allow military forces as well as national and international governmental agencies

> to respond to and operate in a wider variety of circumstances and time frames. It provides US military forces the global reach capability to quickly apply strategic global power to various crisis situations worldwide by delivering necessary forces. The power projection capability that airlift supplies is vital since it provides the flexibility to get rapid-reaction forces to the point of a crisis with minimum delay. Airlift can serve as American presence worldwide, demonstrating our resolve, as well as serve as a constructive force during times of humanitarian crisis or natural disaster.

There are many aspects of airlift, some of which have previously been addressed: the Aerial Fleet Refueling Problem (Wiley, 2001), the Theater Distribution Vehicle Routing and Scheduling Problem (Crino, 2002), the Strategic Airlift Problem (Lambert, 2004), and the Strategic Mobility Mode Selection Problem (McKinzie, 2005). These are explained in further detail in Section 2.1.1.2. An additional area of the airlift process that has not yet been addressed involves (1) *packing* the cargo items onto pallets (if necessary), (2) *partitioning* the set of cargo into aircraft loads, (3) *selecting* an efficient and effective set of aircraft from an available pool of aircraft, and (4) feasibly *placing* the cargo in the best allowable positions. These four tasks are interdependent and their holistic combination is the Airlift Loading Problem (ALP). In addition, the ALP differs from most partitioning and packing problems described in the literature because, in addition to spatial constraints, factors such as weight, balance and temporal restrictions on cargo loading availability and cargo delivery requirements must be considered.

Because they lack the capability to attack an ALP in its totality, explicitly considering the aforementioned interdependencies, classical methods of optimization are reduced to attacking these sub-problems one at a time, in a hierarchical fashion. Tabu search (TS) *has* the capability to solve ALP without resorting to such decomposition.

1.1 A BRIEF OVERVIEW OF THE HISTORY OF AIRLIFT

A complete discussion of the history of airlift is beyond the scope of this research; however, a brief overview demonstrates the rapidity with which this method of transportation can adapt to new challenges and adopt new technologies. According to Bruce Callander, "Airlift tends to get a low priority in peacetime, but that changes when conflict begins" (Callander, 1998). The result of the low peacetime priority is that when a conflict erupts, airlift is required to play "catch-up" to meet the demand.

The roots of airlift and air mobility can be traced to 1915. In this year, the 1st Aero Squadron was established with only eight aircraft. During World War I, the vast majority of all cargo was hauled by surface vehicles (trucks, trains, and ships). In the 1920s, commercial aviation began to flourish which led to an increased interest by the US Army in airlift. The first commercial carriers purchased by the US Army were only designed for passenger transportation. In the 1930s, the US Army began searching for aircraft specifically designed to transport cargo (Callander, 1998).

World War II triggered numerous technological advances in many areas, including transportation aircraft. The demand for airlift generated by World War II far exceeded the predicted amount. This led to the development of new aircraft with larger capacity and greater range. Airlift also proved to be effective in resupplying armies in the field, as was demonstrated by the airdrops to troops pinned down at Bastogne in the Battle of the Bulge (Callander, 1998).

After World War II, the Cold War emerged, and with it one of the most famous and largest airlifts conducted by the US—the Berlin Airlift. After the Soviet Union blockaded West Berlin from all land based transport, an "air bridge" was established to deliver essential foods and medicines to West Berlin. The Berlin Airlift continued over a 15 month period. For a complete history of this specific airlift, the reader is referred to *Airbridge to Berlin—The Berlin Crisis of 1948, its Origins and Aftermath* (Giangreco and Griffin, 1988).

The Korean War emphasized the need for specialized airlifters (Callander, 1998). The US Air Force (USAF) was well equipped to deliver troops and cargo from the US to the theater (intertheater airlift), but it did not have planes designed to transport goods and cargo within the theater (intratheater airlift).

The United States' efforts in Vietnam further demonstrated the need for specialized airlifters as the US forces required both tactical and strategic airlifters to support their actions (Callander, 1998). The US government also realized that military airlift capacity would continue to be insufficient for the demand of strategic airlift. As a result, the Civil Reserve Air Fleet (CRAF) was created. The CRAF is composed of civilian aircraft used for strategic airlift (passenger and/or cargo) in national emergencies or wartime situations.

The buildup for the first Gulf War, Operation Desert Shield, was the "most massive airlift in the history of airpower" (Callander, 1998). USAF and CRAF aircraft flew more ton-miles (a ton of cargo transported one mile) in six weeks than during the entire Berlin Airlift. Numerous unaddressed complications arose for airlift in this operation. For example, the USAF did not have a staging base on the Arabian Peninsula which prevented incoming crews from resting prior to flying in-theater. This required the assignment of extra pilots to airlift aircraft, as well as pools of available pilots at other bases to relieve the incoming pilots. According to a Rand Corp. study, the strategic airlift capability was reduced by 20 to 25 percent due to the lack of a staging base (Callander, 1998).

Airlift is not restricted to use in wartime efforts. For example, in Bosnia, Somalia, and Haiti, USAF airlift "supported multinational forces and non-government organizations such as the Red Cross and CARE" (Callander, 1998). In 2005, in the aftermath of Hurricane Katrina, the USAF flew over 750 sorties, rescued over 1,350 people, evacuated over 13,000 people, and delivered more than 4,000 tons of cargo and supplies (http://www.af.mil/pressreleases/release.asp?storyID=123011597).

Airlift has a history spanning over 80 years. From its humble beginnings in World War I to the massive airlifts conducted in the 21st century, the airlift capabilities of the USAF and CRAF are insufficient to meet the demand placed upon them. While many advances have been made in both aircraft capabilities and airlift procedures, there are still many areas that can be greatly improved. Some of the shortfalls of current airlift practices are addressed in this dissertation. Sections 3.3 and 4.3 provide a more detailed discussion of this topic.

1.2 THE CURRENT AIRLIFT PROCESS

As defined in AFDD 2-6.1: *Airlift Operations*, the airlift system is "an integrated system that incorporated all aspects of intertheater, intratheater, and Joint Task Force-dedicated airlift" (1999). The Air Force (AF) airlift forces are drawn from active duty, Air Force Reserve Command (AFRC), Air National Guard (ANG), and the CRAF components. The airlift system "delivers personnel, patients, and/or cargo when and where they are needed" (AFDD 2-6.1, 1999).

Three classifications of airlift operations exist in the airlift system: Intertheater Airlift, Intratheater Airlift, and Operational Support Airlift (OSA). Intertheater airlift primarily provides "airlift to and from the supported Commander's in Chief (CINC) theater" (AFDD 2-6.1, 1999). Intertheater airlift is available to the Military Services, the combatant commands, other Department of Defense (DOD) components, other US Government agencies, and, in certain cases, foreign governments. Intratheater airlift, in contrast, provides transportation for "personnel and material within a geographic CINCs area of responsibility (AOR)" (AFDD 2-6.1, 1999). OSA provides time-sensitive movement of personnel and small amounts of cargo, and, as such, are typically smaller business-style aircraft.

Airland and airdrop are the two types of delivery methods used in the airlift process. Airland refers to airlift in which "an aircraft lands at the objective air terminal and unloads its cargo; offloading personnel and cargo is done entirely on the ground" (AFDD 2-6.1, 1999). Airdrop refers to the "delivery of personnel and material from an aircraft in flight to a drop zone" (AFDD 2-6.1, 1999).

According to Air Force Doctrine Document 1,

> Air Force airlift missions encompass passenger and cargo movement, combat employment and sustainment, aeromedical evacuation, special operations support, and operational support airlift. These missions can be tasked in a variety of ways: Channel, Air Mobility Express (AMX—a special category of Channel), special assignment airlift missions (SAAM), special air missions (SAM), joint airborne/air transportability training (JA/ATT), or exercise and contingency missions.

Channel missions are split into two categories: requirements- and frequency-based. Requirements-based channel missions are established when a specified amount of passengers or cargo destined for one location warrants movement, while frequency-based channel missions serve locations at regularly scheduled intervals.

SAAMs are those airlift missions flown to locations outside the approval channel structure. SAMs specifically support the White House and other executive branches of the government. An AMX is established to transport critically needed items rapidly to an AOR. JA/ATT missions provide training to and aid in honing proficiency of members of the airlift process. Exercise and contingency missions entail deployment, sustainment, and redeployment by intertheater or intratheater airlift (AFDD 2-6.1, 1999).

This research focuses on the transportation of palletized cargo items that are to be transported for a deployment setting. This research is limited in scope to the passenger and cargo movement mission of the airlift process; specifically requirements-based channel and the exercise and contingency missions tasking categories.

1.2.1 The Pallet Loading Process

Prior to being loaded into an aircraft at an aerial port of embarkation (APOE), cargo and personnel must first be transported to the location. For cargo, this occurs by either rail or troop convoy; passengers are sent on commercial air flights or troop convoy. Prior to shipping, smaller cargo items (called *bulk* cargo) must be loaded onto pallets and secured with cargo netting. This process is called *palletization*.

The DOD uses standard sized pallets (labeled 463L) for transportation of bulk cargo. The 463L pallet has a balsa wood core and is covered with corrosion-resistant aluminum. It is framed on all sides by aluminum rails which have 22 tie-down rings attached with six rings on each of the long sides and five rings on each of the short sides. The rails also have indents (notches) which can accept rail locks when the pallet is placed on an aircraft (http://www.inetres.com/gp/military/ar/misc/463L.html).

The dimensions of the pallets are 88” x 108” x 2¼”. Due to the restriction that each pallet must be secured using cargo netting, 2” on all four sides of the pallet are reserved for tie-down eyelet rings. As pictured in Figure 1.1, this creates a usable space of 84” x 104” for each pallet.

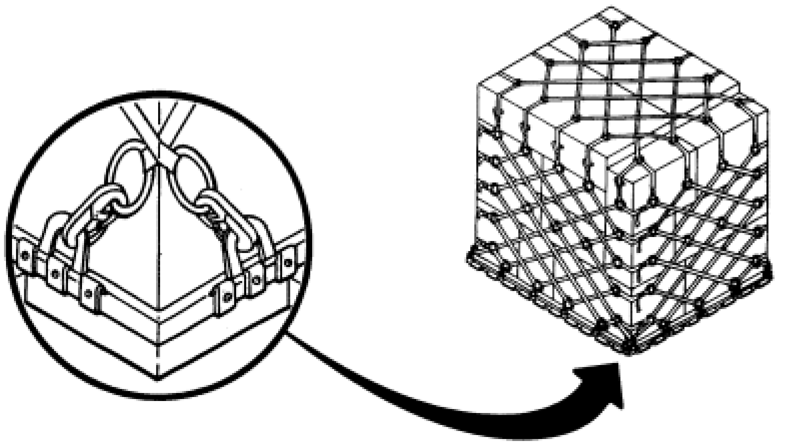

Figure 1.1 463L Tie-Down System

The 463L pallet has a maximum weight capacity of 10,000 lbs. The 463L pallet weighs 290 lbs by itself, and the cargo-netting and hooks weigh 65 1bs. Thus, the maximum gross weight of a packed 463L pallet is 10,355 lbs. Additionally, the 463L is restricted to less than 250 lbs per square inch (psi) of weight. To limit the range of pallet weights to reasonable and non-trivial weights, all pallets in this research are assumed to have a total weight greater than or equal to 2500 lbs but less than or equal to 10,000 lbs. The cargo items are assumed to be palletized prior to consideration for loading onto aircraft. Each loaded pallet is assumed to be “properly” loaded which implies that the center of gravity (CG) of the loaded pallet is located in the lengthwise and widthwise center of the pallet. None of the US armed forces currently attempt to actually verify that this CG is correct or determine the precise location of the CG; it is simply assumed to be in the center of the pallet.

Each aircraft has pre-specified pallet positions. The number and location of these pallet positions are constant for all aircraft that are of the same type (airframe). For example, as indicated in Figure 1.2, all C-17 aircraft in the logistics configuration can carry 18 pallets, with each position having an unchangable location in the aircraft. The rail locks on the aircraft require a spacing of 2 inches between pallets.

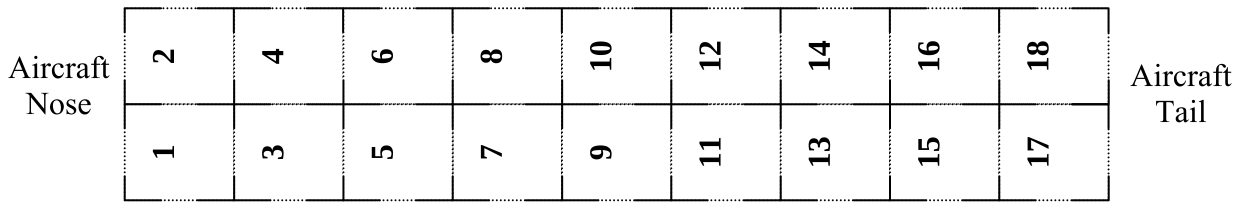

Figure 1.2 C-17 Pallet Positioning

1.3 MOTIVATION

In September 2005, the US Government Accounting Office (GAO) released a report to the US Secretary of Defense detailing the inadequacies with which AMC tracks and analyzes aircraft utility (GAO, 2005). This report specified that analysis of 14,692 strategic airlift missions demonstrated that

> … more than 86 percent flew with payloads that were lighter than established planning factors; nearly 19 percent did not meet the minimum requirements of 15 short tons or 100 passengers needed to qualify for use of strategic airlift (GAO, 2005).

Many factors contribute to aircraft underutilization, but proper planning and detailed aircraft loads can help prevent it. Additionally, aircraft utilization is implicitly increased by reducing the number of aircraft required to transport a set number of items.

Headquarters AMC's Analyses Foundation and Integration Division (AMC/A9I) has expressed a need for a software package that can quickly and effectively generate excellent solutions to tasks 2, 3, and 4 associated with the ALP. AMC/A9I has, at least initially, excluded the palletization task because it is out of its immediate jurisdiction and, in deployment situations, is usually performed by elements of the US Army. As a first step in the integrated solution of the ALP, this dissertation applies advanced tabu search (TS) techniques to the ALP in order to produce a proof of concept demonstrating that this methodology can be utilized to produce a superior product for AMC/A9I.

1.4 RESEARCH OBJECTIVES

The objective of this research is the development and computerized implementation of a TS solution methodology that effectively and efficiently solves static and dynamic instances of the ALP. Effectively solving the ALP requires that the solution produced by this methodology be superior (in terms of objective function value) to those produced by current methods. Efficiently solving the ALP implies that the methodology will produce results in a timely manner when compared to current methods. Valid solutions assign loaded pallets to aircraft at specific locations within the aircraft to ensure the center of balance (CB) of the aircraft is within specified tolerances.

Two types ALPs are addressed in this research. The first is the Static ALP (SALP). The static restriction implies that all cargo and aircraft are leaving from the same APOE and arriving at the same aerial port of debarkation (APOD) without temporal restrictions. Missions flown for a SALP correspond to channel airlift missions. The more general second type of problem, the Dynamic ALP (DALP), allows pallets and aircraft to leave from a single APOE with a single APOD destination as well as temporal restrictions on loading and arriving dates. Missions flown for a DALP correspond to contingency airlift missions.

Additional aspects of the ALP not included in this research could easily be addressed using the same ALP solution representation. For example, wheeled and tracked vehicles could replace a predetermined number of pallets (corresponding to the vehicle's footprint in the aircraft). This research could also be extended to include item incompatibilities due to hazardous material (HAZMAT) classifications.

The previous chapter presented a brief overview of the history of airlift as well as the motivation for and objectives of this research. The following chapter presents a review of the relevant literature associated with this research.

Chapter 2: Literature and Application Software Review

Research conducted on the ALP has been largely limited to military students or personnel and companies conducting research for military agencies. As a result, the ALP has not been extensively defined. The researchers who have explored the ALP have approached it from different perspectives, thereby preventing meaningful side-by-side algorithmic comparison. The following sections present research which has been conducted on the ALP or similar type problems.

2.1 AIRLIFT LOADING LITERATURE

Heidelberg et al. developed a "heuristic algorithm for determining efficient 2-dimensional packings in cargo aircraft where cargo placement constraints are critically important in determining the feasibility of packing locations" (Heidelberg et al., 1998). They note that in standard air load planning, the task of loading is approached as a 2-dimensional bin packing problem using length and width of the cargo items and of the aircraft's cargo hold (Heidelberg et al., 1998). Since cargo items are typically not stacked, height of an item is not significant except to ensure that it will fit into the aircraft.

Heidelberg et al. point out that classical methods of bin-packing (such as Best Fit Level or Best Fit Level Decreasing) are inadequate in aircraft loading (Heidelberg et al., 1998). These methods place the larger (and typically heavier) items toward the aft and left portions of the aircraft. This procedure would frequently result in an intolerable CB for the aircraft. To overcome this situation, load planners employ pyramid loading—cargo is "sorted based on factors including weight, priority, and bulkiness, and is loaded on the aircraft from the desired CB toward the fore and aft of the cargo hold" (Heidelberg et al., 1998).

The algorithm developed by Heidelberg et al. attempts to overcome requirements typically placed on classical bin-packing strategies, such as the level algorithm (Heidelberg et al., 1998). For example, after packing an item into the bin, new, straight line barriers (which divide the bins) are created. The authors utilize nine different barrier types, as necessary. These barriers are shown in Figure 2.1 (Heidelberg et al., 1998).

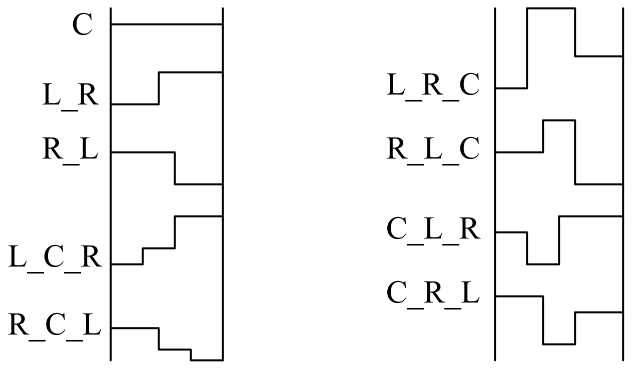

Figure 2.1 Heidelberg et al. Barrier Types

An obvious limitation to this type of barrier method is that it only considers up to three possible levels for the barrier. While not common, it is possible to have more than three levels of loading (i.e. three adjacently packed items) in large cargo aircraft.

Heidelberg et al. evaluated the performance of their algorithm versus two other bin-packing algorithms—Constrained Local Search (CLS) and Best Fit

Level Decreasing (BFLD) (Heidelberg et al., 1998). Two types of data sets were generated and tested on all three algorithms. The first type of data set had orthogonal items of random size (Heidelberg et al., 1998). The second type still had orthogonal items of random size but it also had duplicates of each item (Heidelberg et al., 1998). This type of data set is more realistic for aircraft loading than a complete set of random sized items; in deployment settings, duplicates of several items are often loaded.

The algorithm developed by Heidelberg, et al. reduced the amount of wasted space in all of the data sets of the first type but was out-performed on 1 and matched by 2 instances of the second data set. Additionally, the Heidelberg et al. algorithm required more computation time than the other two (Heidelberg et al., 1998).

Although their algorithm has applications in an academic environment, the real-world application is limited. While they mention CB in their discussion, they do not present any results on the effectiveness of their packing algorithm with respect to CB. They also do not consider the weight of items loaded into the aircraft.

Heidelberg et al. note that their analysis only accounts for the "fundamental packing capability of the algorithms and does not consider their applicability to cargo conveyance systems by considering cargo constraints" (Heidelberg et al., 1998). This limits the real-world applicability of this algorithm—most non-palletized cargo items require tie-down restraints to ensure their stability in flight. This prevents items from being positioned immediately adjacent to another item. In a pallet-only scenario for an airlift problem, all items have the same dimension. The pallets have specific available positions where they may be placed in the aircraft. Each pallet position also has a specific orientation in the aircraft; it cannot be rotated 90° to allow more items to be loaded. The algorithm developed by Heidelberg et al. does not allow for pre-specified locations of items, but rather attempts to minimize the wasted space in the aircraft.

In summary, the Heidelberg et al. algorithm does not really have any real-world applications. They exclude too many important factors which hinders applicability. In the academic environment, this algorithm would most likely be outperformed by Harwig's (2003) algorithm (see Section 2.1.1.2), but no comparisons between the two algorithms have been performed.

In 2003, Gueret et al., proposed a method for loading military aircraft (specifically, the French military) for airlift operations (Gueret et al., 2003). They label the problem a bi-dimensional bin-packing problem, with several bins and several additional constraints. They used a two-phased solution method. The first phase consists of two possible heuristics that quickly compute "good" initial solutions; the second phase is a local search algorithm to improve upon the initial solution. Their algorithm does not compute or attempt to optimize the aircraft CB.

2.1.1 Airlift Loading as a 2-Dimensional Bin Packing Problem

The Bin Packing Problem (BPP) is a classical combinatorial optimization problem and is NP-hard in the strong sense (Garey and Johnson, 1979). The BPP can be described as follows: Given a set of M bins of capacity C and a set of N items of sizes $D_1, \ldots, D_N$, minimize the number of bins required to pack all N items. For a BPP mathematical formulation, the reader is referred to Elhedhli (2003). The BPP is a well posed problem in its own right, and is also "often encountered as a subproblem when solving large-scale optimization problems through decomposition/relaxation approaches" (Elhedhli, 2003). BPPs can be of several dimensions: 1-, 2-, and 3-dimensional BPP (1-D BPP, 2-D BPP, and 3-D BPP respectively).

The ALP can easily be reduced to the 2-D BPP and is therefore also NP-hard in the strong sense. In the ALP, the floor of the aircraft is equivalent to a bin in the BPP. It has a set capacity—the amount of cargo which can be placed in the aircraft. This capacity is expressed in two forms: space and weight. An additional temporal constraint is included in the ALP: items cannot be loaded in aircraft prior to the aircraft's availability and should not be loaded such that they will arrive prior to or after the scheduled item delivery date.

Extensive research has been conducted on the 2-D BPP. Lodi, Martello, and Vigo (2002) present a survey of 2-D BPPs. They describe 5 general methods that have been used to solve the 2-D BPP: Exact Algorithms, Upper Bounds, Lower Bounds, Meta-heuristics, and Variants of the others. Lodi, Martello, and Monaci (2002) also survey advances in the 2-D BPP. They describe Models, Approximation Algorithms, Lower Bounds, and Exact Algorithms. The following sections detail the main advances in 2-D BPP literature with emphasis given to the work relevant to this research.

2.1.1.1 *Classical Approaches*

Exact Algorithms

Integer programming formulations (branch-and-bound) and enumeration techniques account for the majority of the exact algorithms presented in these survey papers. While exact algorithms can prove optimality on smaller problems, the sheer size of the majority of BPPs preclude their use in any environment other than an academic one. Lodi, Martello, and Vigo (2002) present results from an exact algorithm solving problems with 20 to 100 items to be packed into bins. In 500 test cases, only 356 were solved to optimality within the allotted computer time of 300 CPU seconds. The sizes of these problems are too small to be considered for real-world application.

Upper Bounds

Lodi, Martello, and Vigo (2002) survey methods which produce an upper bound for the 2-D BPP. Specifically, they examine two-phase algorithms, one-phase algorithms, and non-level algorithms and compare them to classical approaches used to generate an upper bound. Their comparison demonstrated that each algorithm type can outperform classical approaches.

Lower Bounds

Lodi, Martello, and Vigo (2002) and Lodi, Martello, and Monaci (2002) present surveys of lower bound (LB) methods used to solve the 2-D BPP. Elhedhli ranks lower bounds for the BPP (2003). He discusses and ranks linear programming (LP) bounds, two Lagrangean relaxation bounds, and two Lagrangean decomposition bounds. He further compares these classical bounds with a bound developed by Martello and Toth (1990) and shows that this bound is equivalent to a "Lagrangean bound evaluated at a finite and well-chosen number of Lagrange multipliers" (2003).

Bourjolly and Rebetez (2005) analyze the lower bound algorithms presented by Eilon and Christofides (1971), Martello and Toth (1990) and Labbe, Laporte and Mercure (1991), respectively LB1, LB2 and LB3. LB1 is a "branch-and-bound scheme for solving small instances of the BPP ($n \leq 50$)" (Bourjolly and Rebetez, 2005). LB2 follows from the observation that multiple heavy items cannot be placed into a bin with a set weight capacity. Bourjolly and Rebetez prove that the LB2 ≤ LB3 (2005), which implies that the number of bins required by LB2 is always ≤ the number required by LB3.

2.1.1.2 *Metaheuristics*

Heuristic search methods are approaches that attempt to produce a good (though not necessarily optimal) solution for a combinatorial optimization problem (COP) within an acceptable amount of computation time. Many heuristic search methods, such as Simulated Annealing (SA), Genetic Algorithms (GAs), and TS, have been successfully applied to a wide variety of COPs. SA employs the cooling or annealing process of solids as a model for search in COPs. GAs utilize genetic inheritance as a method of searching COP possible solutions. TS uses memory structures to effectively and efficiently guide the search through the solution space of the COP. The following paragraphs highlight some of the important components of these heuristic search methods and detail previous research that has been conducted on 2-D BPPs using these methods.

Simulated Annealing

According to Eglese, the motivation for the SA algorithm “comes from an analogy between the physical annealing of solids and combinatorial optimization problems” (1990). Physical annealing refers to the “process of finding low energy states of a solid by initially melting the substance, and then lowering the temperature slowly, spending a long time at temperatures close to the freezing point” (Eglese, 1990). The Boltzmann probability distribution is presented in Equation 2.1.

$$\text{Prob}\ (E) \sim \exp\ (-E/kt)$$

Equation 2.1: Boltzmann Probability Distribution

Equation 2.1 expresses the idea that a system in thermal equilibrium at temperature t has its energy probabilistically distributed among all different energy states E. The k in Equation 2.1 refers to the Boltzmann constant. Even at low temperature, there is a small probability that the system is in a high energy state (Press et al., 1992). In physical annealing, a substance is allowed to pass from different physical states to determine its characteristics. The annealing or cooling schedule is composed of the initial temperature, rate at which the temperature is reduced, number of iterations at each temperature and criterion used for stopping (Eglese, 1990). When applied to COPs, SA simulates this annealing schedule which is basically a form of steepest descent with the ability to escape local optima. For COPs, the annealing schedule is composed of the initial solution, other feasible solutions, method to produce solutions, and stopping criterion.

Koulmas, Antony and Jaen (1994) provide a survey of SA applications to operations research problems, including BPPs. Brusco et al. (1997) applied a morph-based SA to a modified BPP (MBPP). The MBPP differs from the BPP in that the number of bins is known and each has infinite capacity. The objective of the MBPP is to minimize the maximum workload assignment (across all bins). They demonstrated that algorithm was superior to previously developed SA algorithms. Abramson and Randall develop a simulated annealing code which can be applied to general integer linear problems (1999). They apply their code to BPPs with a weight capacity (but not a space capacity). Their algorithm is tested on problems with 120, 250, and 500 items. Their algorithm does not prove optimality of the solutions.

SA relies on the temperature and the annealing schedule to perform the local search. SA does not utilize or exploit memory—no information is passed from iteration to iteration.

Genetic Algorithms

GAs are based on the mechanics of natural selection and natural genetics. They combine survival of the fittest with a randomized information exchange to form a search algorithm. In every generation, a new set of creatures is created from the old. GAs follow these general steps in searching the solution space: (1) Create an initial population, (2) Evaluate each chromosome (member of the population), (3) Create new chromosomes through recombination and mutation, (4) Delete old chromosomes from population, add new chromosomes, and (5) Stop if a termination criteria is satisfied; else, go to 3. Additional strategies exist (e.g. Goldberg (1989) and Haupt and Haupt (2004)) which can be applied to make GAs more efficient and effective.

Raidl and Kodydek (1998) propose two variants of GAs to solve the Multiple Container Packing Problem (MCPP). The MCPP differs from a strict BPP in that there is a value or utility associated with loading each item. They allow the GA to run until no improvements are encountered within the previous 200,000 evaluations—this large number is to ensure that the GA has enough time to converge. They note that they are "primarily interested in finding high-quality solutions and only secondarily in the needed CPU-time" (Raidl and Kodydek, 1998). Their algorithm found superior solutions but required much more computational time. Effective solutions are important and desirable, but efficient methods are also desirable, so the importance of CPU-time should not be diminished.

Pimpawat and Chaiyaratana (2004) present a cooperative co-evolutionary GA (CCGA) to solve three-dimensional container loading. Their method differs from previous methods of finding an optimal sequence of packages to be loaded into containers in that they partition the entire loading sequence into a number of shorter sequences. Each of these partitions is represented by a member (chromosome) in the CCGA search. CCGA was tested against problems with a classification of three groups: large-sized, medium-sized, and small-sized package groups. CCGA proved to be more effective when tested against other forms of GAs. Unfortunately, the algorithm was not tested against any other heuristics or classical methods.

Bhatia and Basu (2004) propose an algorithm to solve the BPP using a multi-chromosomal genetic representation and better-fit heuristic. In the better-fit heuristic, a "left-out object replaces an existing object from a bin if it can fill the bin better" (Bhatia and Basu, 2004). They test their algorithm on the BPP "uniform" data sets available from Beasley's OR-Library (1990), but they only make comparisons with other GAs.

Lewis et al. (2005) describe a distributed chromosome GA for BPPs. To accomplish this, the authors distribute the workload of the bin packing across several CPUs. Unlike most other algorithms to solve BPPs, this algorithm can be applied to non-orthogonal objects. The computational results show that distributing the workload will reduce the required time to reach a solution compared to a sequential GA.

GAs do not produce the same results every time the algorithm is completed. In order to properly compare a GA's performance with other methodologies, the GA must be run several times, and the results must be averaged over those runs.

Tabu Search

In TS, an *initial* or *incumbent* solution is generated by a predetermined methodology. A *move* or modification to the incumbent solution is defined. The *neighborhood* is composed of the set of all solutions which can be reached from the current incumbent solution using this type of move. The next incumbent solution is chosen from the neighborhood based on a merit function and the details of the algorithm. The *move value* is defined as the change in the objective function due to this move. The solution with the best objective function value achieved in the search is labeled the *best solution found* and the associated objective function value is the *best value*. The best solution found is updated when a move results in an objective function value better than the best value. This procedure repeats until a *stopping criteria* has been met. Typical stopping criteria are completion of a predetermined number of iterations, a predetermined amount of time, or a predetermined number of iterations without an improvement of the best solution found. The best solution or an ensemble of best solutions is recorded for later use.

In TS, a *tabu memory structure* is utilized to track attributes of solutions that have been recently visited. These solutions are “forbidden” or tabu for *tabu tenure* future iterations. This prevents the search from returning to solutions recently visited. The tabu tenure can be constant or variable, depending on the desired type of search. The search iteratively selects the best neighbor solution which may be a better solution or the least disimproving solution. In this way, the search is able to escape from local optima and continue the search. Additionally, an optional *aspiration criteria* can allow a move to a tabu neighbor if, for example, it results in the best solution found to that point in the search. A general TS pseudo-code is presented in Figure 2.2.

As a simple example of TS, consider an *N* city Traveling Salesman Problem (TSP) with a single salesman where the travel distance between each city is known. The objective is to create a tour for the salesman to visit each city once while minimizing the travel distance, the tour length. A solution representation for this problem is an ordered set of cities or a “tour”. For example, one possible solution is given by: (1,2,3,4,…,*N*). A common move type for this type of problem is an *insertion* move. In an insertion move, a city is removed from its location in the tour and inserted between two other cities. Returning the city to the same location in the tour is made tabu for tabu tenure iterations.

```
Initialize Tabu Memory Structure
Generate & Evaluate Initial Solution
Best Value = ∞
Set Incumbent = Initial Solution
Set Incumbent Value = Initial Solution Value
While (Stopping Criteria Not Met)
{
    Select Desired Neighborhood
    Best Move Value = ∞
    For (all candidate moves)
    {
        Evaluate Move Value
        If ((Move NOT Tabu) ∪
                (Incumbent Value + Move Value < Best Value))
        {
              If (Move Value < Best Move Value)
              {
                  Best Move Value = Move Value
                  Best Move = Move
              }    // END If (Current Move Value < Best Move Value)
        }      // END If ((Current Move NOT Tabu) ∪
                          (Incumbent Value + Move Value < Best Value))
    }   // END For (all candidate moves)
    Perform Best Move
    Incumbent Value += Best Move Value
    Update Tabu Memory Structure
    If (Incumbent Value < Best Value)
    {
        Best Value = Incumbent Value
        Best Solution = Incumbent Solution
    }   // END If (Incumbent Value < Best Value)
}   // END While (Stopping Criteria Not Met)
Output Best Solution and Best Value
```

Figure 2.2 General TS Pseudo-Code

Following the pseudo-code presented in Figure 2.2, the tabu memory structure is initialized (no moves are tabu), and the initial solution or tour is generated and evaluated. Assume that the initial solution is given by (1,2,3, …,*N*). The best value achieved thus far is set to the tour length associated with the initial solution. The initial solution is the starting incumbent solution, and the initial solution value is the starting incumbent value.

Once the initialization process has been completed, the algorithm enters the search loop. Several stopping criteria are possible for this problem; suppose, for this problem, that 15 adjacent disimproving moves result in loop termination. The neighborhood under consideration is the insert neighborhood; all non-tabu insertions are considered. Additionally, since an aspiration criteria is in use, any tabu move which results in a new best value is considered.

The best available insertion move is performed, the incumbent solution and incumbent value are updated, and the insertion of the moving city into its previous location is made tabu for tabu tenure iterations. If this is a disimproving move, the disimproving move count is incremented; otherwise, it is set to zero. If the incumbent value is superior to (in this case less than) the best value achieved thus far, a new best tour has been located. The process repeats until the stopping criteria has been reached. The final solution and value are presented and the algorithm terminates.

This simple example illustrates the general components of TS. The following section details variations which can be used in TS to increase effectiveness and efficiency.

Tabu Search Variations

Several additional strategies have been developed to enhance the effectiveness of TS. *Intensification* strategies concentrate the search in areas of better solutions, and *diversification* strategies allow the search to escape from areas of poor solutions. Adaptive TS (ATS) varies the tabu tenure, bounded between defined upper and lower limits, using a myopic rule that dictates that the tabu tenure is decremented (incremented) if the current iteration improves (disimproves) the objective function relative to the previous iteration. Reactive TS (RTS) uses hashing methods to provide fine-gauge differentiation between individual solutions. RTS stresses the use of routines that automatically adjust the search parameters based on the quality of the search (Battiti and Techiolli, 1994). In 1999, Colletti developed Group Theoretic TS (GTTS) which exploits the inherently powerful framework afforded by abstract algebra terms to greatly enhance basic TS methodologies. GTTS has been applied to numerous ordering as well as partitioning and ordering (P|O) COPs. GTTS recently received its first application to strict partitioning problems (Kinney, 2005).

Tabu Search Applications to Air Mobility Problems

TS has been very successfully applied to a wide variety of large-scale COPs involving airlift. This review is limited to literature addressing applications associated with the air mobility process.

Wiley (2001) applied GTTS methods to the Aerial Fleet Refueling Problem (AFRP), which is concerned with in-flight refueling of a fleet of US Air Force aircraft. Wiley’s algorithm dramatically reduced the planning time and required resources to support refueling operations. He was able to generate *excellent solutions* in a matter of hours to problems which typically required weeks or months for a team of analysts to generate a *usable* (not necessarily good) solution.

Combs (2002) applied GTTS methods to the Aerial Fleet Crew Scheduling Problem (AFCSP), which is concerned with scheduling the crews to operate the aircraft in the AFRP. Combs utilized the natural partitioning structure of GTTS to place flights into a disjoint cycle which represents a crew rotation. The disjoint cycles also represent partial solutions to the crew scheduling problem. His algorithm uses a classical set partitioning model of the disjoint cycles, within a GTTS methodology, to improve the search process through vocabulary building (McKinzie, 2005).

Crino (2002) used GTTS and RTS to address the Theater Distribution Vehicle Routing and Scheduling Problem (TDVRSP). The TDVRSP is concerned with scheduling and routing the delivery of cargo items in a theater of operations. After the cargo and/or passengers arrive at the port of debarkation, they must be transported to the final destination in theater. Crino utilized GTTS to effectively and efficiently divide the solution space into “good” and “bad” partitions. After partitioning the solution space, Crino used RTS to exhaustively search partitions which contained the “good” solutions. The algorithm performs well on the thirty-nine benchmark problems in which it was tested.

Lambert (2004) utilized ATS to address the Strategic Airlift Problem (SAP). The SAP is concerned with efficiently delivering cargo through the airlift network. In this algorithm, cargo is initially assigned to aircraft (based solely on weight). Because the cargo is required to arrive within pre-specified time windows, excellent routings of the aircraft from origin to destination are determined such that the time constraints' (as well as other constraints') violations are minimized. Lambert's method dominates the Air Mobility Operations Simulation (AMOS), the method for solving the SAP.

McKinzie (2005) used ATS to address the Strategic Mobility Mode Selection Problem (SMMSP), which determines the best method (airlift or sealift) for transportation of deployment cargo. The SAP is actually a subset of the SMMSP—after cargo is assigned to airlift, the SAP routes the cargo through the network. The Time-Phased Force Deployment Document (TPFDD) contains a list of all of the items requiring transportation in support of a pre-existing military operational plan, as well as their various time windows (e.g. departure and arrival dates) and preferred mode of transportation. In some TPFDDs, a large number of the items may contain errors (McKinzie, 2005). In the current process, these errors could cause the transportation requirement to be completely ignored. McKinzie (2005) used pre-processing to screen the TPFDD for errors—the result on the Tunisia TPFFD was that 40% more items are recognized for transportation than under current procedures. After correcting the input files, McKinzie first used ATS to solve the SMMSP directly (McKinzie, 2005). Next, she relaxed the TPFFD port stipulations both for POEs and PODs while maintaining feasibility with regard to required delivery constraints. Finally, the transportation mode constraints (allowing cargo to switch from sealift to airlift or vice-versa) were relaxed within specific limits. McKinzie's algorithm generates deployment plans which allow for transportation of more cargo and passengers and results in *markedly* less items arriving late than the current methodology, the Joint Flow and Analysis System for Transportation (JFAST) (McKinzie, 2005).

Tabu Search Applications to the 2-D BPP

Lodi, Martello, and Vigo (1999) describe heuristic and metaheuristic methods to solve 2-D BPP. They first describe variations to BPPs which can arise: (a) *Orientation*. The items may either have a fixed orientation, or they can be rotated (by 90°), and (b) *Guillotine cuts*. It may or may not be imposed that the items are obtained through a sequence of edge-to-edge cuts parallel to the edges of the bin.

From these variations, the authors consider four problems:

1. 2BP|O|G: the items are oriented (O), i.e. they cannot be rotated, and guillotine cutting (G) is required.
2. 2BP|R|G: the items may be rotated by 90° (R) and guillotine cutting is required.
3. 2BP|O|F: the items are oriented and cutting is free (F).
4. 2BP|R|F: the items may be rotated by 90° and cutting is free.

Lodi, Martello, and Vigo (1999) compared their TS algorithm against heuristic algorithms developed by Berkey and Wang (1987). The algorithm developed by Lodi, Martello, and Vigo was applicable to all four variations of the BPP described above, while the heuristic algorithms of Berkey and Wang could not accommodate guillotine cuts. The algorithm developed by Lodi, Martello, and Vigo matched or outperformed the heuristic algorithms in both solution quality and CPU-time for the majority of the cases tested.

Harwig (2003) developed an Adaptive TS algorithm to solve 2-D BPPs. Harwig utilized a dynamic neighborhood selection process to guide the search through the solution space. This process allows the algorithm to select the type of move which will produce the most desirable change for a given situation. Harwig also developed the concept of the "Big Bin"—a large bin of infinite capacity (Harwig, 2003). Any item that is not actually loaded into a bin resides in the Big Bin. His technique was shown to be markedly superior to previous methods, improving the duality gap for a standard set of benchmark problems by an average of 25%.

Tabu Search Approach to the ALP

Chocolaad used TS to solve the ALP, a problem he defined as a geometric knapsack problem (Chocolaad, 1998). He does not assume that all items must be transported but rather that there is a utility associated with each item and a limited number of aircraft (Chocolaad, 1998). His algorithm hierarchically solves two subproblems, a knapsack problem (KP) and a packing problem (PP) (Chocolaad, 1998). The KP selected the items to potentially pack (based on the utility of the item), and the PP selected the location for the items (Chocolaad, 1998). The KP must select items before the PP can place them. The utility of an item is set equal to the weight of the object. This implies that heavier objects are more important than lighter objects, which is not necessarily the case in real-world applications.

Chocolaad's algorithm did consider the CB of the aircraft; however, it only considered the CB aspects for the longitudinal axis and ignored the vertical and lateral axes because changes in these axes "are small and flight controls can compensate for any effect on the stability of the aircraft" (Chocolaad, 1998). Additionally, his algorithm only considered one type of aircraft, the C-17A Globemaster III (C-17), of which only one aircraft was loaded. Finally, this algorithm does not consider passengers, only cargo.

Romaine (1999) expanded upon the work begun by Chocolaad. He expanded the algorithm to consider two types of aircraft: the C-17 and the C-5B Galaxy (C-5). Additionally, his algorithm can load up to seven C-17 and C-5 aircraft of different combinations. As with Chocolaad, his algorithm sets the utility of an item equal to its weight. There are also other similar limitations as those from Chocolaad's work: the algorithm does not consider passengers and the CB considerations are only for the longitudinal axis.

2.2 AIRLIFT LOADING APPLICATION SOFTWARE

Research in the form of application software is much more prevalent than ALP research published in the literature. The definitions of the problem being solved are quite different from one software package to another. This prevents meaningful side-by-side software comparisons. The following sections detail application software which has been used or is currently in use by the USAF.

2.2.1 Deployable Mobility Execution System and Computer Aided Load Manifesting

Cochard and Yost (1985) were among the first to attempt to increase the efficiency of loading cargo onto an aircraft. In 1982, they developed a computer system, called Deployable Mobility Execution System (DMES), for use by the USAF. Their model used a modified cutting stock heuristic which only generated *feasible* loads for the aircraft. These feasible loads could then be modified as necessary by the load planner using an interactive user interface. DMES was later revised, and in 1985 it was released as the USAF standard under the name Computer Aided Load Manifesting (CALM). While numerous upgrades have been made to the overall software program, no real modifications have been made to the actual loading heuristic. Neither of these software packages have the ability to generate solutions to a large scale ALP. As a result, they were later replaced.

2.2.2 Airlift Loading Model

The Airlift Loading Model (ALM) was developed for the Air Force Studies and Analysis Agency (AFSAA) which was recently renamed AF/A9. ALM is AF/A9's research and evaluation tool for analysis of loadability of military combat and support units on airlift aircraft. The model is used to evaluate capabilities and requirements of current and future airlift fleets. ALM uses aircraft characteristic parameters and loading algorithms to determine the number of sorties required to move military units including vehicles, equipment, supplies and combat troops. This model is a basic building block for analysis of airlift capabilities and requirements relating to military campaigns (http://afmsrr.afams.af.mil/index.cfm). Three of the algorithms used by this model are the *fill-gap*, the *top-down*, and the *floor-utilization*. The fill-gap algorithm finds a gap in a cargo load and selects a cargo item to fill it. The top-down algorithm loads cargo based on the user defined sorting sequence. It selects the next cargo item to be loaded, and then finds an available gap which will support the item. The floor-utilization algorithm allows the user to specify a floor space to Allowable Cabin Load (ACL) ratio (Computer Sciences Corporation, 1997). While ALM is a very useful model, it is more concerned with the transportation of vehicles and other outsize and oversize cargo than pallets and passengers (both of which have pre-determined available locations).

2.2.3 Automated Air Load Planning System

In 1998, all four military branches—Army, Navy, Air Force, and Marines—were mandated by the DOD to utilize the same aircraft loading system, called the Automated Air Load Planning System (AALPS). It applies to all military aircraft used in airlift missions. It is a "knowledge-based expert system" that assists users in planning and executing aircraft loads for all types of deployments (http://www.tis.army.mil/AALPS/default.htm). AALPS ensures feasible aircraft loads—it checks both two- and three-dimensional spatial clearance, axle weights, and overall cargo load weight. It has the capability to adapt the loads as necessary to adjust for different equipment and aircraft configurations, as well as different mission-specific constraints such as cross-loading and center of balance constraints (Computer Sciences Corporation, 1997). AALPS is an extremely user friendly computer program, but it is very limited in its optimization capability. AALPS allows the user to select cargo (from a database containing all known oversized and outsized cargo) and pallets of various weights and then positions these items onto the aircraft.

AALPS has two different loading methods—by priority and by ratio. When AALPS loads by priority, it first places the items with the highest priority on the available aircraft and continues until either all items have been loaded or all available aircraft are full (by either space or weight). This is similar to a multi-knapsack problem—many aircraft (knapsacks) are available, but they are not necessarily sufficient for all cargo items. Additionally, the cargo items each have an associated utility (their priority) which makes loading some items more desirable than others.

When loading by ratio, a set of available aircraft is pre-specified, and AALPS loads a multiple of this set until all cargo items are loaded. For example, if a C-5 and a C-17 were in the available ratio set (in that order), AALPS would load a C-5, a C-17, a C-5, a C-17, etc. until all items were loaded. In this manner, AALPS produces an upper bound on the number of aircraft needed to transport a set of cargo items.

The loading algorithm used by AALPS is extremely efficient, which implies that AALPS can quickly generate solutions. As is further discussed in Section 3.3, these solutions may require more aircraft than are actually needed. To accomplish the loading, AALPS first sorts the cargo items by decreasing priority then by decreasing weight. If all items have the same priority, then they are simply sorted by decreasing weight. AALPS loads the heaviest item with the highest priority first, followed by the next heaviest item with the highest priority, etc. When the cargo load in the aircraft reaches either the aircraft ACL or maximizes the available space, the algorithm continues to the next aircraft. AALPS only loads until one of the two constraints (ACL or space) is reached—it does not try to maximize both of them simultaneously.

2.3 SUMMARY

This chapter provided a review of the literature associated with the ALP. Literature relating to airlift loading as a 2-D BPP was also presented, followed by the presentation of solution approaches, both classical and metaheuristic. Airlift loading application software which was used or is in use by the USAF was also presented. The following chapter provides a detailed description of the static ALP as well as the methods used to solve it.

Chapter 3: A Tabu Search Approach to the Static ALP

This chapter provides a detailed description of the Static ALP (SALP), the solution representation used in this research, and the SALP-TS algorithm used to solve the problem.

3.1 Detailed Problem Statement

As was previously described, given a set of cargo items to be transported and a set of aircraft available for transportation, the goal of the ALP is to transport all of the cargo items using the fewest number of aircraft possible while still satisfying the space, weight, and CB restrictions imposed upon each aircraft. The ALP can be decomposed into four sub-problems: (1) *packing* the cargo items onto pallets (if necessary), (2) *partitioning* the set of cargo into aircraft loads, (3) *selecting* an efficient and effective set of aircraft from an available pool of aircraft, and (4) feasibly *placing* the cargo in the best allowable positions. The SALP is composed of sub-problems 2, 3, and 4. The SALP considers only items which have already been palletized and have the same destination with no temporal constraints, i.e., all of the pallets must be transported from a given APOE to the same APOD using a set of available aircraft. Each aircraft is allowed a single use (trip) in this formulation of the problem.

The SALP may be viewed as dealing with channel missions or the sustainment missions of a deployment. Channel missions can be either requirements- or frequency-based channel missions. Sustainment missions of a deployment include those that occur *after* the items on a TPFDD have been transported.

3.1.1 SALP-TS Inputs

Two tab delimited text files containing the aircraft and pallet information, easily generated using Microsoft Excel, are required for SALP-TS. Sample SALP-TS aircraft and pallet input files are presented in Appendix A and B, respectively.

3.1.1.1 *Aircraft Input File*

For each aircraft, the aircraft input file contains: (1) a *unique* integer aircraft identification (ID) number, (2) aircraft type (integer category), and (3) allowable cabin load (ACL). The location and number of the available pallet positions within a designated aircraft type will not change. Pallets may be placed only in an available pallet position on an aircraft. For example, as demonstrated in Figure 1.2, all C-17 aircraft in the logistics configuration have 18 defined pallet positions in specified locations.

The ACL is the maximum combined weight of items loaded into the aircraft. The ACL depends upon the flight length and available refueling aircraft. According to DOD 4500.9-R Part III, Mobility, Defense Transportation Regulation "flight route segments less than critical leg distances may allow for more or less ACL, depending on wind factors" (2004). Accurate ACL information can be derived only from known operating conditions and is normally established at the deployment planning conference or at the time of mission execution. A deploying unit's Tanker Airlift Control Element (TALCE) provides the specific ACL for each aircraft. This ACL is given to unit load planners for each operation. Frequently, the load planning must be accomplished before the actual mission ACL has been computed. As a result, DOD 4500.9-R Part III, Mobility, Defense Transportation Regulation provides planning ACLs based upon average wind factors throughout the world.

The planning ACL gives only a guideline for the weight which can be loaded into a particular aircraft. The aircraft types used in this research, along with their descriptions and planning and maximum ACLs are shown in Table 3.1. The aircraft type corresponds to the coding used by SALP-TS; this is not an AF designation. Although some of these aircraft types may not be frequently used as airlift aircraft, they are all included (as of November 2005) in AMC's Airload Planners Course which necessitates their inclusion in this research.

Aircraft Type #:	Description	Planning ACL (lbs)	Maximum ACL (lbs)
0	C-130	25,000	40,000
1	C-17 (Logistics System--18 Pallets)	90,000	175,000
2	C-17 (Air Drop System--11 Pallets)	90,000	175,000
3	C-5	150,000	291,000
4	KC-10 (17 Pallets)	80,000	150,000
5	KC-10 (23 Pallets)	80,000	150,000
6	C-141	46,000	70,000
7	KC-135E	30,000	40,000
8	KC-135R	30,000	40,000

Table 3.1 Aircraft type with Planning and Maximum ACLs

There is a large discrepancy between the planning ACL and the maximum ACL. If the unit load planners do not have the actual ACL, they must use the planning ACL. This will either produce equivalent loads (if the actual ACL is equivalent to the planning ACL), inferior loads (if the actual ACL is greater than the planning ACL), or infeasible loads (if the actual ACL is less than the planning ACL). Since the planning ACL is only a guideline, minor violations in this area are exploitable.

There are three distinct possible combinations of available aircraft: (1) The aircraft are of the same type with the same ACL, (2) The aircraft are of the same type, but at least one has a different ACL, and (3) The aircraft are of different types. In the first two cases, the number of available pallet positions is constant among all aircraft. In the third case, this number varies among the available aircraft. The three combinations allow for different calculations of the Lower Bound on required aircraft which is further explained in Section 3.2.4.

This discussion assumes that the available aircraft will allow a feasible solution. To ensure this, a solution to the problem is first generated by a USAF certified air load planner using AALPS version 4.3.3.1. This solution presents an upper bound on the number of aircraft required for a solution. The number of planes in the aircraft input file for SALP-TS is equivalent to the number of planes required by AALPS.

3.1.1.2 *Pallet Input File*

The pallets are sorted by decreasing weight to allow for better initial solution generation (see Section 3.2.5). The pallet input file contains: (1) pallet ID number, (2) loaded pallet weight, (3) loaded pallet height, (4) loaded pallet width, and (5) loaded pallet length.

3.2 METHODOLOGY

The previous section presented the input files required by SALP-TS. This section details the SALP-TS methodology. The complete pseudo-code for this algorithm is in Appendix C.

3.2.1 SALP-TS Data Structures

Table 3.2 details the contents of three data structures used by SALP-TS.

Aircraft	ID Number, Index, Type, Number of Pallet Positions, ACL, Loaded Cargo Weight, Total Longitudinal Moments, Total Lateral Moments, Longitudinal CB, Lateral CB, Number of Loaded Pallets, % Space Full, % Weight Full, CB Lower Bound, CB Upper Bound, Optimal CB, OF Weight Usage, OF Longitudinal CB, OF Lateral CB, OF Aircraft Usage
Pallet	ID Number, Index, Weight, Height, Width, Length, Group Number
Solution	OF Value, Number of Available Aircraft, Number of Aircraft Used, Total Number of Pallets, Solution Array, Big Bin List

Table 3.2 SALP-TS Data Structures with attributes

An *aircraft* object is assigned to each available aircraft. As detailed in Section 3.2.3, the aircraft index is used by the solution object and the tabu memory structure. The aircraft index need not be the same as the user defined ID Number (such as aircraft tail number).

The remaining fields, which can be altered by SALP-TS, are initialized at zero. The objective function (OF) fields (OF Weight Usage, OF Longitudinal CB, OF Lateral CB, and OF Aircraft Usage) combine to form the contribution of an aircraft to the overall OF value. The remaining fields are used to calculate the OF fields and are described in more detail in Section 3.2.6.

All fields for the *pallet* object, except the Index and Group Number, are specified by the input file. The unique index number, not necessarily the same as the ID number, is used by the solution object and the tabu memory structure. Each pallet is assigned to a group which allows SALP-TS to generate better initial solutions. The use of the groups is further explained in Section 3.2.5. All of the fields in a pallet object are constant; they cannot be altered by SALP-TS.

A SALP-TS *solution* object contains the constant Total Number of Aircraft and Total Number of Pallets fields. The Number of Aircraft Used is a count of the aircraft used in a particular solution. The OF Value is the combination of the individual OF fields of each aircraft as well as a usage fee for pallets not loaded in any aircraft. The Solution Array and Big Bin List detail the actual placement of pallets into specific aircraft. As detailed in Section 3.2.2, the Solution Array is initialized as empty while the Big Bin List initially contains all of the pallets which require transportation.

3.2.2 SALP-TS Solution Representation

The SALP-TS solution representation is composed of an assignment of pallets to specific positions in available aircraft. By AF designation, each pallet position in an aircraft has a specific reference number. In aircraft with a single row of pallet positions, the numbering is sequential from nose to tail. For example, in the C-130, the six pallet positions are numbered 1, 2, …, 6 from nose to tail. In aircraft with two rows of pallet positions, the pallets are labeled 1L, 1R, 2L, and so on from the nose to the tail. For example, in a C-17 aircraft in the logistics configuration, there are 18 pallet positions. The pallet position closest to the nose of the aircraft on the left (port) side is designated 1L, and the pallet adjacent to it is 1R. The pallet positions at the tail of the aircraft are 9L and 9R.

The structure used in the SALP-TS solution representation with a single row of pallet positions is the same as the AF designation (i.e. 1, 2, …, 6). For aircraft with two rows of pallet positions, the designation is changed from 1L, 1R, 2L, …, 9R to 1, 2, 3, …, 18. Figure 1.2 clearly presents the pallet position designation of a C-17 aircraft in the logistics configuration.

The SALP-TS solution representation is simply a set of assignments of pallets to specific positions within an aircraft. For example, consider four C-130 aircraft available to transport fifteen pallets. A possible solution for this problem could be given by the following representation:

(1, 4, 5, 6, 1, 2, 3)(2, 7, 8, 9, 0, 10, 11)(3, 12 13, 0, 0, 14, 15)(4, 0, 0, 0, 0, 0, 0)(0)

The first number in each set is the aircraft index. The remaining numbers are loaded pallet indices. An index of zero corresponds to an empty pallet position. In this example problem, aircraft 1 has pallet 4 in position 1, pallet 5 in position 2, and so on. Aircraft 4 has no pallets loaded. The last set corresponds to Harwig's Big Bin; it is "aircraft" 0 with infinite capacity for weight and space capacity equal to the total number of pallets (Harwig, 2003). In reality, the Big Bin is a warehouse or the tarmac at the APOE. Any unloaded pallet resides in the Big Bin, and an empty Big Bin is represented by a single zero in the set.

3.2.3 Tabu Memory Structure

A tabu memory structure is utilized by TS algorithms to track attributes of solutions which have been recently visited. Such solutions may not be revisited for *tabu tenure* future iterations.

The SALP-TS tabu memory structure is a three-dimensional array of integers. The size of the tabu memory structure is given by:

(Number of Pallets) x (Max Number of Pallet Positions) x (Number of Aircraft+1)

Equation 3.1: Size of Tabu Memory Structure

The Max Number of Pallet Positions is the largest number of available pallet positions in any of the available aircraft. An additional "aircraft" (which corresponds to the Big Bin (Harwig, 2003)) is included in the tabu memory structure, yielding the "+1" on the Number of Aircraft. A position exists in the tabu memory structure for each pallet in each available position on each available aircraft. Every pallet also has an available position in the Big Bin. The positions in the tabu memory structure are accessed using the pallet index, pallet position index, and aircraft index.

As an example, consider a problem of transporting 25 pallets using two C-130 aircraft and one C-141 aircraft. The C-130s each have 6 pallet positions, and the C-141 has 13 pallet positions. The tabu memory structure would be a three-dimensional array of size (25 x 13 x 4).

If the available aircraft are of different types, the tabu memory structure will include unnecessary positions. In the example above, the tabu memory structure has allocated positions for 13 pallets in each aircraft, but the C-130s only have 6 available positions. While they do require additional computer memory, the extra positions do not significantly increase the computation time or change the quality of the SALP-TS results.

After a move is performed, a return to the previous incumbent solution is made tabu for tabu tenure additional iterations. SALP-TS places an upper and lower bound on the tabu tenure. The tabu tenure is not allowed to drop below 4% and cannot exceed 20% of the total number of pallets. The tabu tenure is initialized at 12% of the total number of pallets. These values were determined to be acceptable through the performance of experiments early in the algorithmic development.

Including the ATS mechanism enhances the effectiveness of SALP-TS by allowing the tabu tenure to dynamically change in response to the search. While respecting the stated bounds, an improving (disimproving) move causes a decrement (an increment) in the tabu tenure. This has the effect of intensifying the search in areas of improvement and diversifying when the solution has become worse.

As detailed in Section 3.2.7, there are four different SALP-TS move types which may be viewed as either fine or broad gauge moves. Fine gauge moves occur when a pallet is moved *within* a specific aircraft. In this case, a moved pallet is not allowed to return to the prior *position* for tabu tenure additional iterations. A broad gauge move occurs when one or more pallets are removed from an aircraft. In this case, the pallet is prevented from returning to the prior *aircraft* for tabu tenure additional iterations.

The tabu memory structure guides the search into areas with excellent solutions and helps escape areas of poor solutions. It is an essential part in the efficiency and effectiveness of SALP-TS.

3.2.4 SALP Lower Bound

AALPS generates an upper bound on the number of aircraft required for a feasible solution, and a lower bound is easily calculated for the minimum number of aircraft required. This is computed by simultaneously considering both the total available weight capacity and total number of available pallet positions. Figure 3.1 provides the SALP-TS pseudo-code for calculating this lower bound. It may not be possible to achieve the computed lower bound since a specific SALP instance may not have any feasible solutions using only that number of aircraft. The lower bound is important because it provides a baseline for measuring the quality of solutions.

The three distinct aircraft combinations, as detailed in Section 3.1.1.1, require three separate methods for calculating the lower bound. When aircraft are of the same type with the same ACL, the lower bound is determined by calculating:

$$\max\left(\left\lceil \frac{\text{total number of pallets}}{\text{number of pallet positions in a aircraft}} \right\rceil, \left\lceil \frac{\text{total pallet weight}}{\text{aircraft ACL}} \right\rceil\right)$$

Equation 3.2: SALP-TS Lower Bound

When the aircraft are of the same type with differing ACLs, the lower bound is computed by sequentially removing aircraft by ascending ACL, starting with the smallest ACL, until a removal would cause either total pallet positions or total ACL to be insufficient. When the aircraft are of different types with different ACLs, SALP-TS sequentially removes, in ascending order, the still available aircraft with the smallest ratio of ACL to number of available pallet positions until any additional removals would cause either the total pallet positions or total ACL to become insufficient.

```
Calculate the Lower Bound:
  {
Three possible combinations of aircraft:
1.  The aircraft are the SAME type (each aircraft has the same number of pallet positions) and have the SAME ACL.
2.  The aircraft are the SAME type, but have DIFFERENT ACLs.
3.  The aircraft are DIFFERENT types with DIFFERENT ACLs.
        If (Combination 1)          // Determine lower bound
        {   # Aircraft Lower Bound = max ( ⌈total number pallets / number pallet positions in aircraft⌉,
                                           ⌈total pallet weight / aircraft ACL⌉          )
        }  // END If (Combination 1)
        Else if (Combination 2)     // Determine lower bound
        {   Lower Bound = Total number of aircraft
            While (can still remove aircraft)
            {  Get useable aircraft with smallest ACL
                 If  (can remove this aircraft and still meet total ACL AND number pallet position restrictions)
                 {   Decrement Lower Bound
                     Make Aircraft Unusable
                 }   // END if
                 Else {   cannot remove any more aircraft     }  // END Else
            }  // END While (can still remove aircraft)
        }  // END Else if  (Combination 2)
        Else  (Combination 3) // Determine lower bound
        {   While (can still remove aircraft)
            {    Get useable aircraft with smallest ratio of ACL to Number of Pallet Positions
                 If  (can remove this aircraft and still meet total ACL AND number pallet position restrictions)
                     {    Decrement Lower Bound
                          Make Aircraft Unusable
                     }    // END if
                     Else {    cannot remove any more aircraft           }  // END Else
            }  // End While (can still remove aircraft)
        }  // END Else (Combination 3)
        For each aircraft:          // All unusable aircraft need to be usable to get initial solution.
        {   If (aircraft is unusable)
            {    Make aircraft usable   }     // END If (aircraft is unusable)
        }  // END For each aircraft:
}  // END Calculate Lower Bound
```

Figure 3.1   Pseudo-Code for Calculation of Lower Bound

3.2.5   Initial Solution Generator

Since SALP-TS does not even *require* an initial feasible solution, SALP-TS attempts to produce a quality initial solution without too much computational effort. When generating an initial solution, SALP-TS attempts, for each aircraft used, to simultaneously maximize both the number of pallets loaded and the total weight of the loaded cargo. Number of pallets loaded is deemed maximized when all positions are occupied. The total weight is deemed maximized when adding any additional available pallet would exceed the aircraft ACL. The SALP-TS pseudo-code to generate an initial solution is presented in Figure 3.2.

```
Generate Initial Solution:
{ For 1 to total number of aircraft:
  {   Get useable aircraft with largest ratio of ACL to Number of Pallet Positions
      While ( (BigBin NOT empty) ∩ (All Aircraft NOT in State 4) )
      {    Increment iteration count (For TS memory structure use)
Four descriptive aircraft states during initial solution generation:
1. Not maximum for either weight or space.          2.  Maximized for weight but not space.
3. Maximized for space but not weight.              4.  Maximized for space and weight.
                If (State 1)
            {    For (each group)
                 {    If (can add pallet without violating space and weight)
                      {    Add pallet to aircraft  }  // END If
                 }  // END For (each group)
            }  // END If (State 1)
            If (State 2)
            {    Get heaviest pallet in aircraft, Remove pallet to BigBin
                 Update Tabu Memory Structure (Make return move tabu)
                 Get head pallet of lightest non-empty group
                 While ((aircraft NOT maxed weight) ∩ (aircraft NOT maxed space) ∩ (NOT at END of BigBin list))
                 {  If ((aircraft can hold pallet) ∩ (NOT Tabu move) ){Add pallet to aircraft }// END If
                 }  // END While
            }  // END If (State 2)
            If (State 3)
            {    Get lightest pallet in aircraft, Remove pallet to BigBin,
                 Update Tabu Memory Structure (Make return move tabu)
                 Get head pallet of heaviest non-empty group
                 While ( (aircraft NOT maxed weight) ∩ (aircraft NOT maxed space) ∩ (NOT at END of BigBin list) )
                 {    If ( (aircraft can hold pallet) ∩ (NOT Tabu Move) ) {  Add pallet to aircraft   }  // END If
                      Get next pallet
                 } // END While
            }  // END If (State 3)
            If ((number loaded pallets unchanged) ∩ (cargo weight changed) )
            {    Increment CycleCount      }     // END If
            Else{ CycleCount = 0 }     // END If
            If  (( CycleCount > 10)  ∪ OR (Case 4))
            {    EXIT While (BigBin NOT empty) loop }           // END If
      }  // END While ( (BigBin NOT empty) ∩ (All Aircraft NOT in State 4) )
  }  // END For 1 to total number of aircraft
} // END Generate Initial Solution
```

Figure 3.2 Pseudo-Code for Initial Solution Generator

To accomplish the goal of simultaneously maximizing aircraft pallets and weight, SALP-TS initially partitions the pallets into groups. The number of groups is user defined and can range between 1 and 10. Based on initial experiments, 4 groups were used in the generation of the computational results discussed in Section 3.3. The pallets are sorted by decreasing weight with the first group containing the heaviest pallets. SALP-TS attempts to equally divide the number of pallets between groups. If the total number of pallets precludes equal division, then the last (lightest) group will have any additional pallets.

To begin the process of assigning pallets to aircraft, SALP-TS selects the unfilled aircraft with the largest ratio of ACL to number of pallet positions, i.e., the aircraft that can carry the most weight per pallet position. If there are multiple aircraft of the same type with the same ACL, SALP-TS selects the one with the smallest index. All pallets are currently in the Big Bin. At any stage in the initial solution generator, there are four possible descriptive states of the aircraft being loaded: (1) not maximized for either weight or space, (2) maximized for weight but not for space, (3) maximized for space but not for weight, and (4) maximized for both space and weight.

If the aircraft is in state 1, the initial solution generator begins with the heaviest non-empty group. A group's *head pallet* is the first available pallet that has not been given prior consideration for the current aircraft. The group's head pallet will be loaded if it will not cause the combined pallet weight to exceed the aircraft ACL. If the head pallet in any group cannot be loaded onto the aircraft, SALP-TS advances to the next group.

If a state 1 iteration is completed in which no pallets have been added, the aircraft is in state 2 and SALP-TS chooses the *heaviest* pallet in the aircraft, removes it from the aircraft, and replaces it into its original group. A pallet which is removed from the aircraft is not allowed to return for at least 10 iterations. Next, SALP-TS selects the non-empty group with the *lightest* pallets. If loading the head pallet will not violate the aircraft ACL and the move is not tabu, it is loaded. If the head pallet is not loaded, it is marked as unavailable for the current aircraft. Regardless of whether the pallet is loaded, the group's new head pallet is considered. The algorithm does not stop at the end of a group, but rather continues to the next group. This process is repeated until the aircraft is maximized according to weight, space, or both, or the end of the list of available pallets is reached.

When an aircraft is in state 3, SALP-TS chooses the *lightest* pallet in the aircraft, removes it from the aircraft, and replaces it into its original group. A pallet which is removed from the aircraft is not allowed to return for at least 10 iterations. SALP-TS selects the heaviest available (non-tabu) pallet that can be loaded without causing violation of the aircraft ACL. If no available pallet can be added without causing a violation, the pallet which was removed at the start of the iteration is reinserted and the aircraft is declared to be in state 4.

If an aircraft is in state 4, SALP-TS selects the next unloaded aircraft with the largest ratio of ACL to number of pallet positions and repeats the entire process. This continues until either all pallets have been loaded or no additional aircraft are available for loading.

It is possible for SALP-TS to exhibit cyclic behavior while loading an aircraft. To prevent cycling, the initial solution generator has an additional exit criteria. If an entire iteration is completed with no change in the number of pallets loaded onto the aircraft (i.e. a single pallet is unloaded and a single pallet is loaded), a variable labeled CycleCount is iterated. If there is a change in the number of loaded pallets, this variable is set to zero. If the CycleCount variable reaches a value of 10, SALP-TS declares the aircraft to be in state 4.

The SALP-TS initial solution generator ignores such things as CB requirements and will usually produce sub-optimal and/or infeasible solutions with respect to these constraints.

3.2.6 Objective Function

The SALP-TS objective function to be minimized is a combination of penalties and usage fees. The components of the objective function are Unloaded Pallet Penalty, Aircraft Usage Fee, Percent Weight Full Penalty, Lateral CB Penalty, and Longitudinal CB Penalty. The Unloaded Pallet Penalty applies to the overall solution. The remaining components apply to each aircraft and are combined additively to form the OF Value.

3.2.6.1 *Unloaded Pallet Penalty*

To be feasible, a solution must have *all* pallets loaded onto an aircraft. As a result, the penalty multiplier for the each unloaded pallet must be relatively large. The large penalty drives SALP-TS to load all pallets onto available aircraft. The Unloaded Pallet Penalty is given by

$$\lambda_1 \sum_{i=1}^{N} B_i$$
$$B_i \in \{0,1\} \forall i = 1,\ldots,N$$

Equation 3.3: Unloaded Pallet Penalty

where B_i = 1 if pallet i is in the Big Bin and 0 otherwise (it is loaded on an aircraft,), N = total number of pallets to be transported, λ_1 = the penalty factor associated with the unloaded pallets.

3.2.6.2 *Aircraft Usage Fee*

The objective function is increased by a user specified amount, C_j, for each aircraft j used. This enables the user to adapt SALP-TS to various scenarios. The aircraft usage fee is given by

$$\sum_{j=1}^{M} C_j A_j$$
$$A_j \in \{0,1\} \forall j = 1,\ldots,M$$

Equation 3.4: Aircraft Usage Fee

where C_j = the usage fee (cost) associated with aircraft j; A_j = 1 if aircraft j is used and 0 otherwise; M = the number of aircraft available.

3.2.6.3 *Percent Weight Full Penalty*

The goal is to maximally load the aircraft. The squared deviation percent weight full penalty is calculated for each aircraft and multiplied by a penalty factor. The sum of these values is added to the objective function value.

The percent weight full penalty for underloading is

$$\lambda_2 \sum_{j=1}^{M} \left(100 - \%WF_j\right)^2 A_j X_j$$
$$A_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$X_j \in \{0,1\} \forall j = 1,\ldots,M$$

Equation 3.5: Percent Weight Full Penalty (underloading)

where $\%WF_j$ = percent aircraft ACL loaded for aircraft j; $X_j = 1$ if $\%WF_j$ of aircraft $j \leq 100$, 0 otherwise; and λ_2 = the penalty factor associated with percentage weight full ≤ 100.

As was mentioned in Section 3.1.1.1, the planning ACL is not a hard constraint and, under certain circumstances, it can be exceeded. SALP-TS allows the aircraft's ACL to be violated but imposes a heavier penalty on such violations. The percent overloading is squared and then multiplied by an additional penalty factor. This additional factor must not be less than one, otherwise overloading would be preferred to under-loading. The percent weight full penalty for overloading is

$$\lambda_2 \lambda_3 \sum_{j=1}^{M} \left(100 - \%WF_j\right)^2 A_j \left(1 - X_j\right)$$
$$A_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$X_j \in \{0,1\} \forall j = 1,\ldots,M$$

Equation 3.6: Percent Weight Full Penalty (overloading)

where λ_3 = penalty multiplier associated with percentage weight full > 100. It is not possible to simultaneously be over- and under-loaded; the decision variable (X_j) in Equations 3.5 and 3.6 ensures that only one penalty is non-zero for a given aircraft load.

3.2.6.4 *CB Penalty: Lateral and Longitudinal CB Computation*

The CB lateral and longitudinal aircraft penalties are also computed for each aircraft. The CB is the distance (measured in inches) of the cargo load CB from the associated reference line. The CB is calculated by summing the moments of all pallets and dividing by the total weight of the cargo. The moment of each pallet is the product of the pallet weight and the distance from the associated reference line. In equation form, an aircraft's CB (either lateral or longitudinal) is computed as

$$CB_j = \frac{\sum_{k \in K} (W_k * D_k)}{\sum_{k \in K} W_k}$$

Equation 3.7: CB Calculation

where CB_j = Center of Balance of aircraft j; W_k = Weight of pallet k; D_k = Distance of pallet k's CG from the reference line; and K = subset of pallets currently loaded on aircraft j.

Lateral CB

The lateral CB is only calculated for aircraft with two rows of pallets: the C-17, C-5, and KC-10. The reference line for the lateral CB is at the exact aircraft center from nose to tail. The two rows of pallet positions are placed symmetrically about the reference line. As was stated in Section 1.2.1, aircraft rail locks require two inches of separation between pallets. As a result, an extra inch separates each pallet's edge from the reference line. For a C-5, the pallets' long (108") sides are parallel to the fuselage yielding pallet CG distances of 55"

(54” from pallet’s edge to its CG plus 1” for spacing between rail locks) from the reference line.

In the C-17 and KC-10, the orthogonal pallet orientation yields CG distances from the reference line of 43” (42” from pallet’s edge to its CG plus 1” for spacing between rail locks). Left side pallets have negative distances, while right side pallets have positive distances. The target for the lateral CB is 0 inches from the reference line. Figure 3.3 illustrates the reference line and distance to pallet CG of a C-17 aircraft in the logistics configuration.

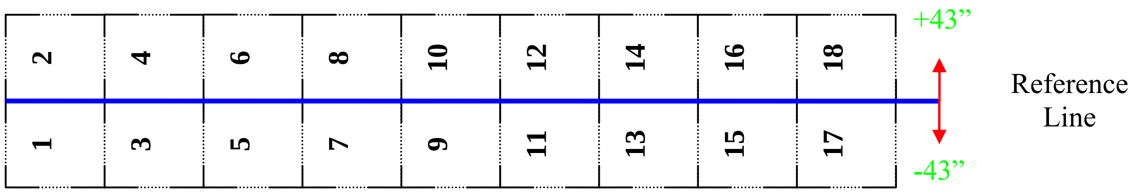

Figure 3.3 C-17 Lateral CB Reference Line

The lateral CB contribution to the OF value is the sum of the squares of the CB locations for each aircraft multiplied by a penalty factor and is given as

$$\lambda_4 \sum_{j=1}^{M} \left(Lat\_CB_j\right)^2 A_j$$
$$A_j \in \{0,1\} \forall\, j = 1,\ldots,M$$

Equation 3.8: Lateral CB Penalty

where *Lat_CB*$_j$ = Actual Lateral CB of aircraft j and λ_4 = penalty multiplier associated with lateral CB violations.

In most load plans (including those produced by certified load planners), the lateral CB is assumed to be insignificant and thus is ignored. The distance to the CG of an item for the lateral CB is small relative to the length of an aircraft wing. The lateral CB impact on the aircraft flight characteristic is very small; pilots can utilize the aircraft trim feature to enable the aircraft to fly straight and level even without required pilot inputs. Complications to this assumption of insignificant flight contribution of the lateral CB arise when turbulence or aircraft trim malfunctions occur. In these scenarios, the pilots will be required to physically maintain control of the aircraft at all times. Any aid which can be rendered to the pilots in these times is invaluable.

Properly loaded cargo (with a lateral CB near the aircraft center) allows pilots to more easily control the roll of the aircraft. As a result, lateral CB calculations are included in this research. The inclusion of the lateral CB results in insignificant additional computation time. The potential improvements in the load quality (and hence the ease of control of the aircraft) require its inclusion.

Longitudinal CB

The reference line for longitudinal CB is the *reference datum line* (RDL). It is a line at or near the nose of the aircraft and is identical for all aircraft of the same type. The *fuselage stations* (FS) are measurements (in inches) from the RDL to a specific point within the aircraft. Since the pallet positions are fixed and their center positions correspond precisely to the associated FS, the distance to pallet CG or the associated moment arm is simply the FS location of the center of the pallet. Appendix D contains a complete listing of FS locations for the CG of pallet positions for all aircraft considered in this research. Figure 3.4 illustrates the FS locations for each pallet position for a C-17 aircraft in the logistics configuration. Note that the CG of the pallet positions 1 and 2 are 444 inches from the RDL which corresponds to the FS of 444.

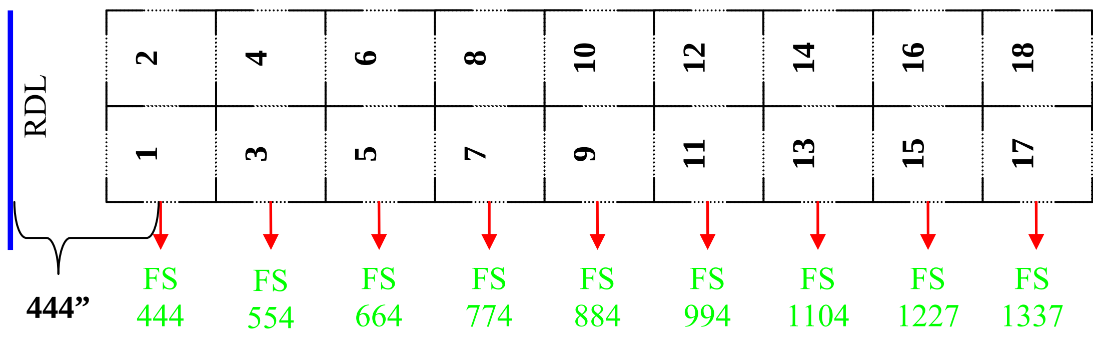

Figure 3.4 C-17 FS Locations

The longitudinal CB is computed analogously to the lateral CB. The pallet weights are multiplied by their FS locations and these products are summed for all pallets and divided by the total weight of all pallets.

Each aircraft has a target longitudinal CB location. This is the FS *at the current cargo weight* where the aircraft obtains the best fuel consumption rate. Unlike the lateral CB, the longitudinal CB has an upper bound and a lower bound for the CB location of specific total loaded cargo weights. If the aircraft attempts to fly with a longitudinal CB located outside this window, the aircraft can depart from normal flight. Appendix E contains the complete listing of allowable CBs for each aircraft at specific combined cargo weights as well as the target CB location for these weights.

To compute the penalty associated with the longitudinal CB location, steps very similar to that of the lateral CB penalty are followed. The squared deviation from the target value is computed for each aircraft and these values are summed and multiplied by a factor. If the longitudinal CB location is outside of the allowable window, the previous amount is multiplied by another factor to ensure the solution is *much* less desirable. The penalty associated with the longitudinal CB location is given by

$$\begin{aligned}
&\lambda_5 \sum_{j=1}^{M} \left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_j Y_j \\
&+ \lambda_5 \lambda_6 \sum_{j=1}^{M} \left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_j \left(1 - Y_j\right) \\
&A_j \in \{0,1\} \forall\, j = 1,\dots,M \\
&Y_j \in \{0,1\} \forall\, j = 1,\dots,M
\end{aligned}$$

Equation 3.9: Longitudinal CB Penalty

where Y_j =1 if the Longitudinal CB of aircraft *j* is within its window, 0 otherwise; *Target_Long_CB*$_j$ = Target FS location for Lateral CB of aircraft *j*; *Long_CB*$_j$ = Actual Longitudinal CB of aircraft *j*; λ_5 = penalty factor for longitudinal CB violations from target value; and λ_6 = penalty multiplier associated with longitudinal CB location outside of specified window

CB Calculation Example

As an example of CB calculation, consider a C-17 aircraft in the logistics configuration which has been assigned 18 pallets that have a combined weight of 90,000 lbs. The pallet ID number and weights of each pallet are given in Table 3.3.

ID Number	Weight
1	7500
2	7500
3	7500
4	7500
5	7500
6	7500
7	7500
8	4500
9	4500
10	4500
11	4500
12	4500
13	2500
14	2500
15	2500
16	2500
17	2500
18	2500

Table 3.3 Pallet ID number and Weights for CB calculation example.

If pallets are considered to be unique (those with identical weights are not interchangeable), then there are $18! = 6.40 \times 10^{15}$ possible different loading configurations for this aircraft. If pallets with identical weights are considered to be inter-changeable, then there are $7!*5!*6! = 4.35 \times 10^{8}$ different loading configurations. Since only weight and distance to CG are considered in CB calculation, the second number of configurations is applicable for this research.

One possible loading configuration for this example is shown in Figure 3.5. The pallet position indices are shown above the aircraft diagram, and the pallet ID indices are indicated in **bold**.

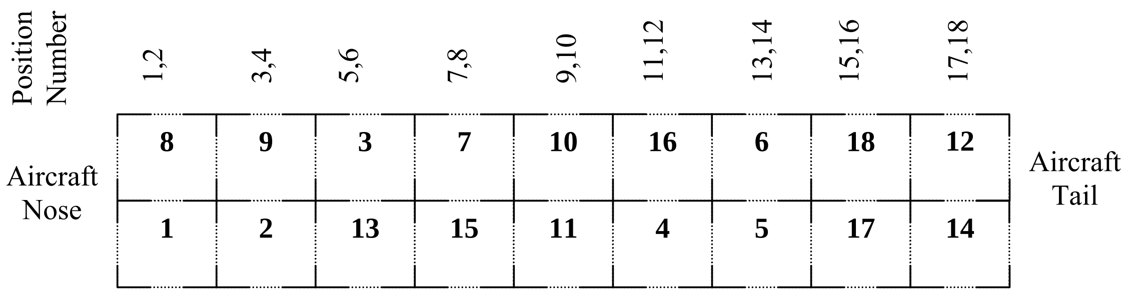

Figure 3.5 Possible loading configuration for CB calculation example.

For a C-17 with 90,000 lbs of cargo, the longitudinal CB window is from FS 760 to 935 with a target of 847.5. For the Figure 3.3 example, the lateral CB is computed using Equation 3.7. Since this is a C-17 aircraft, the D_k (distance from the aircraft centerline to the CG of the pallet or distance of the moment arm) for Equation 3.6 is +43" for all pallets on the right-side of the aircraft and -43" for all pallets on the left-side of the aircraft. The W_k from Equation 3.6 is the pallet weight listed in Table 3.3. The actual computation for the lateral CB is given by:

$$\frac{((7500+7500+2500+2500+4500+7500+7500+2500+2500)*43)}{90000}+$$
$$\frac{((4500+4500+7500+7500+4500+2500+7500+2500+2500)*(-43))}{90000}=$$

$$\frac{44500*43}{90000}-\frac{45500*43}{90000}=$$

$$21.2611-21.7389=-0.4778$$

The longitudinal CB for this example problem is also calculated using Equation 3.7. The longitudinal CB uses the FS location of the pallet centers as the distance to CG. For longitudinal CB, the D_k from Equation 3.7 for each pallet corresponds to the FS location of the pallet's CG. Appendix D presents the pallet CG FS location for each aircraft used in this research.

The actual CB computation is shown below. There are 9 different FS locations for the 18 pallet positions on a C-17. To aid the understanding of this calculation, each pallet is listed separately (rather than combined as they were for the lateral CB).

$$\frac{7500*444}{90000}+\frac{4500*444}{90000}+\frac{7500*554}{90000}+\frac{4500*554}{90000}+\frac{2500*664}{90000}+\frac{7500*664}{90000}+$$
$$\frac{2500*774}{90000}+\frac{7500*774}{90000}+\frac{4500*884}{90000}+\frac{4500*884}{90000}+\frac{7500*994}{90000}+\frac{2500*994}{90000}+$$
$$\frac{7500*1104}{90000}+\frac{7500*1104}{90000}+\frac{2500*1227}{90000}+\frac{2500*1227}{90000}+\frac{2500*1337}{90000}+\frac{4500*1337}{90000}=$$

$$37+22.2+46.1667+27.7+18.444+55.333+$$
$$21.5+64.5+44.2+44.2+82.833+27.6111+$$
$$92+92+34.0833+34.0833+37.1389+66.85=847.844$$

For this example, the lateral CB is located -0.4778 inches from the aircraft centerline or 0.4778 inches to the left of the centerline. The longitudinal CB is located at FS 847.844. Both of these calculations are for the *CB only*; the resulting values are used in Equations 3.8 and 3.9 to compute the penalties associated with these CB locations. Using these values, the lateral CB contribution (Equation 3.8) to the objective function is:

$$\lambda_4(-0.46667)^2 = \boldsymbol{\lambda_4(0.2181)}$$

The FS location for the longitudinal CB is within the allowable window of 760 – 935, which implies the second half of Equation 3.9 is 0. The longitudinal CB contribution (Equation 3.9) to the objective function is:

$$\lambda_5\,(847.5 - 847.844)^2 + \lambda_5\lambda_6\,(847.5 - 847.844)^2{*}0 = \boldsymbol{\lambda_5(0.118) + 0}$$

3.2.6.5 *Objective Function Summary*

The SALP objective function additively combines Equations 3.3, 3.4, 3.5, 3.6, 3.8, and 3.9 described in this section and is given by

$$\begin{aligned}
&\lambda_1\sum_{i=1}^{N}B_i + \sum_{j=1}^{M}C_jA_j + \lambda_2\sum_{j=1}^{M}\left(100-\%WF_j\right)^2 A_jX_j + \lambda_2\lambda_3\sum_{j=1}^{M}\left(100-\%WF_j\right)^2 A_j\left(1-X_j\right)\\
&+\lambda_4\sum_{j=1}^{M}\left(Lat\_CB_j\right)^2 A_j + \lambda_5\sum_{j=1}^{M}\left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_jY_j\\
&+\lambda_5\lambda_6\sum_{j=1}^{M}\left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_j\left(1-Y_j\right)\\
&A_j \in \{0,1\}\,\forall\, j=1,\ldots,M\\
&B_i \in \{0,1\}\,\forall\, i=1,\ldots,N\\
&X_j \in \{0,1\}\,\forall\, j=1,\ldots,M\\
&Y_j \in \{0,1\}\,\forall\, j=1,\ldots,M
\end{aligned}$$

Equation 3.10: SALP-TS Objective Function

Minimizing the objective function allows SALP-TS to search for solutions which use a minimal number of aircraft and have pallets positioned ideally.

3.2.7 Move Neighborhoods

SALP-TS performs four different types of pallet move neighborhoods. Each associated neighborhood serves a specific purpose in guiding the search to quality solutions. *Swap moves* cause two pallets to change positions and only involve pallets with different weights. *Insert moves* cause a pallet to change its current location by moving to an empty pallet position. The four neighborhoods are: (1) Big Bin to Aircraft Insert, (2) Unload Entire Aircraft, (3) Intra-Aircraft Insert/Swap, and (4) Inter-Aircraft Insert/Swap. Each move is called under specific strategic circumstances.

3.2.7.1 ***Big Bin to Aircraft Insert Neighborhood***

The Big Bin to Aircraft Insert neighborhood removes pallets from the Big Bin and inserts them *only* into an *empty* position in an aircraft. Every pallet in the Big Bin will be removed when this neighborhood is invoked.

To perform moves with this neighborhood, SALP-TS first selects the heaviest pallet in the Big Bin. SALP-TS then chooses the non-empty aircraft with at least one pallet position empty which has the greatest unsatisfied ACL. If every aircraft has already met or exceeded its ACL, the pallet is inserted into the aircraft with the smallest violation. The pallet is inserted into the selected aircraft's lowest indexed empty pallet position. The process repeats until the Big Bin is emptied.

As was detailed in Section 3.1.1.1, the initial number of aircraft available for SALP-TS is the same as that required in the feasible AALPS solution. As a result, sufficient pallet positions will always be available for this neighborhood. The assignment of pallets to empty pallet positions may cause ACL violations, but this neighborhood ignores both the ACL and CB constraints. SALP-TS will use other move neighborhoods to produce feasible solutions.

3.2.7.2 *Unload Entire Aircraft Neighborhood*

One of the goals of SALP-TS is to minimize the number of aircraft required to produce a feasible solution. To achieve this goal, an entire aircraft may occasionally need to be emptied. This can occur when there is one pallet on the aircraft or the aircraft has every pallet position occupied.

This move is very similar to a Big Bin to Aircraft move. The only difference is that an aircraft is emptied instead of the Big Bin. The selected aircraft is emptied of pallets in order of decreasing weight and the pallets are inserted into non-empty aircraft with at least one empty pallet position and the greatest weight capacity remaining. If every aircraft has already met or exceeded its ACL, the pallet is inserted into the aircraft with the smallest violation.

If every pallet position on every usable aircraft (excluding the aircraft being unloaded) is occupied, unloading ceases. The aircraft will remain usable for the remainder of the algorithm. In this scenario, SALP-TS has found a lower bound which, unlike the aircraft ACL, cannot be violated. Removing additional aircraft would result in insufficient pallet positions available to transport all pallets.

3.2.7.3 *Intra-Aircraft Insert/Swap Neighborhood*

An Intra-Aircraft Insert/Swap neighborhood involves a single aircraft. Every possible two-tuple (swapping two pallets or inserting a pallet into an empty position) *within* the aircraft is evaluated. Swaps of pallets with identical weights are disallowed because of the null effect on the aircraft load. The best non-tabu move is selected and performed. The lowest indexed pallet moved may not return to its previous position for tabu tenure future iterations.

3.2.7.4 *Inter-Aircraft Insert/Swap Neighborhood*

The Inter-Aircraft Insert/Swap neighborhood involves two non-empty aircraft. Every possible two-tuple (swapping two pallets or inserting a pallet into an empty position) *between* two aircraft is evaluated. Swaps of identical pallets are disallowed. The best non-tabu move is selected and performed. Returning a pallet to its donor aircraft is not allowed for tabu tenure future iterations.

3.2.8 Local Search Procedure

The four neighborhoods described in Section 3.2.7 are strategically employed during the search. SALP-TS incorporates a dynamic neighborhood selection process to select the most appropriate neighborhood at each point in the search.

The local search procedure begins immediately after the generation of the initial solution. The pseudo-code for the dynamic selection process is shown in Figure 3.6.

```
Commence Local Search
{     Create and Initialize Tabu Memory Structure, Initialize Variables
      Adjust CB:
      { For (each aircraft)
            {     While ((Number of Trial Improvements < 5) AND (Number of Disimproving Moves = 0) )
                  { Perform Intra Aircraft Swap-Select best possible swap between pallets on same aircraft.  }  // END While
            }  // END For (each aircraft)
      }  // END Adjust CB
      Set Iterate Count = 0
      Set Fix CB Count = 0
      Dynamic Neighborhood Search:
      {  While ( (Sequential Disimproving Move Count<=20) AND (Sequential Trivial Move Count<=20) AND (Iterate Count<=500) )
      {     // Increment Iterate Count
Four possible solution states:
1.  BigBin is NOT empty                   2.  (Aircraft CB is out of window) ∩ (Fix CB Count <= 5)
3.  {Weight Usage of Aircraft ? 25% }
∪  { [Number of Non-Empty Aircraft ? Lower Bound on Number of Aircraft]  ∩ [15 Adjacent Trivial Improvements ] }
∪  { [Number of Non-Empty Aircraft ? Lower Bound on Number of Aircraft]  ∩ [15 Sequential Disimproving Moves ] }
4.  Neither State 1, 2 or 3
            If (State 1)
            {     Set Fix CB Count = 0
                  While (BigBin NOT empty)
                  {     If (still empty pallet positions on *any* usable aircraft)
                        {  Get aircraft with ( (maximum weight capacity remaining) AND (one or more pallet positions empty) )        }
                        Else
                        {  Get *unusable* aircraft with *largest* ratio of ACL to Number of Pallet Positions.  Make aircraft usable.  }
                        Get heaviest pallet in BigBin. Add pallet to Aircraft.
                        Update Tabu Memory Structure (pallet cannot return to BigBin for Tabu Tenure iterations)
                  }  // END While (BigBin NOT empty)
            }  // END If (State 1)
            Else if (State 2)
            {     Increment Fix CB Count
                  Perform Intra-Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert within an aircraft)
                  If (Disimproving Move)
                  {     Undo move.  Fix CB Count = 6  }     // END If (Disimproving Move)
                  Else
                  {     Update Tabu Memory Structure (pallet may not return to its *previous position* for Tabu Tenure iterations)
                  }     // END Else
            }  // END Else If (State 2)
            Else If (State 3)
            {     Set Fix CB Count = 0
                  Get lightest loaded, useable aircraft.  Make aircraft unusable.
                  While (aircraft not empty)
                  {     Remove heaviest pallet, Insert into lightest loaded, useable aircraft
                        Update Tabu Memory Structure (pallet may not return to *any position* in losing aircraft for Tabu Tenure iterations)
                  } // End While (aircraft not empty)
            }  // END Else If (State 3)
            Else (State 4)
            {     Set Fix CB Count = 0
                  Perform an Inter Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert between two aircraft)
                  Update Tabu Memory Structure (pallet may not return to *any position* in losing aircraft for Tabu Tenure iterations)
            }  // END Else (State 4)
            If ( (New Solution OF Value*1.025) < Previous Solution OF Value)
            { Increment Improving Move Count, Disimproving Move Count = 0 }       // END If
            Else If ( New Solution OF Value < Previous Solution OF Value)
            { Increment Trivial Move Count, Disimproving Move Count = 0 }         // END Else If
            Else { Increment Disimproving Move Count, Improving Move Count = 0}  // END Else
       }  // END While
      } // END Dynamic Neighborhood Search
```

Figure 3.6   Pseudo-Code for Dynamic Neighborhood Selection

The first step in the SALP-TS search is to create the tabu memory structure, as described in Section 3.2.3, and to create and initialize the variables required in the search. Next, SALP-TS attempts to adjust all CBs for the aircraft.

2.1.1.1 CB Adjustment

Since the initial solution generator ignores CB, SALP-TS performs a series of Intra-Aircraft Insert/Swap moves on each aircraft to attempt to correct any CB violations without changing the total pallet weight or number on an aircraft.

The Intra-Aircraft Insert/Swap moves are categorized as improving, trivially improving, or disimproving. An improving move decreases the overall OF value by at least 2.5%. A trivially improving move decreases the overall OF value by 0-2.5%. A disimproving move increases the OF value. The value of 2.5% was selected based upon initial experimentation directed at avoiding search entrapment in a region which would allow only very small improving iterations.

At this point, since SALP-TS is attempting to ensure that no aircraft has a longitudinal CB outside of the CB window, each aircraft is limited to five sequential trivial improving moves or one disimproving move. Large improvements ($\geq$ 2.5%) may initially occur if an aircraft has a longitudinal CB violation, but these will quickly reduce as the CB violations are corrected.

3.2.8.2 *Stopping Criteria for the Dynamic Neighborhood Selection Process*

After the longitudinal CB adjustments are performed, SALP-TS initiates the dynamic neighborhood selection portion of the search. For purposes of clarity, we consider the stopping criteria first. The dynamic neighborhood selection is repeated until there have been (1) 20 adjacent disimproving moves; (2) 20 adjacent trivially improving moves, or (3) 500 total iterations.

The stopping criteria counts are initialized to zero. Every time a disimproving move is performed, the disimproving move count increments. If either type of improving move is performed, the disimproving move count is reset to zero. If a trivial improving move is performed, the trivial improving move count is incremented. If an improving move is performed, the trivial improving move count is reset to zero. If a disimproving move is performed, the trivial move count *does not change*. This logic prevents SALP-TS from indefinitely cycling between disimproving and trivially improving moves.

3.2.8.3 *Dynamic Neighborhood Selection Process*

Recalling the SALP-TS move neighborhoods previously described in Section 3.2.7, four states describe the status of the current SALP solution and each state invokes a different move neighborhood. These states are:

1. Big Bin is not empty
2. (Aircraft CB Allowable Window Violated) $\cap$ (Fix CB Count $\leq$ 5)
3. {Weight Usage of any Aircraft $\leq$ 25%}
   $\cup$ {[Number of Non-Empty Aircraft $\geq$ Lower Bound of Aircraft]
   $\cap$ [15 Adjacent Trivially Improving Moves] }
   $\cup$ {[Number of Non-Empty Aircraft $\geq$ Lower Bound of Aircraft]
   $\cap$ [[15 Adjacent Disimproving Moves] }.
4. Neither case 1, 2, or 3 applies.

State 1

The Big Bin can only contain pallets immediately after the initial solution is generated. Despite the fact that a feasible solution is known to exist, it is rare, but possible, for SALP-TS to produce an initial solution which uses all available aircraft but is unable to load all pallets. For state 1, SALP-TS utilizes the *Big Bin to Aircraft Insert* which will load all remaining pallets. SALP-TS uses the other neighborhoods to correct any resulting infeasibilities. This move helps drive the solution toward feasibility and results in relatively large changes to the OF value.

State 2

An aircraft longitudinal CB window violation requires a "quick fix" with the *Intra-Aircraft Insert/Swap* neighborhood. There are three ways to escape state 2: (1) the CB window is satisfied, (2), the CB window is unsatisfied for 5 iterations (timely progress is not present), or (3) the CB window violation worsens. These moves result in relatively small changes to the OF value by changing the CB of a single aircraft.

State 3

Three situations render state 3 which invokes the *unload entire aircraft* neighborhood:

(1) Any aircraft with percent weight utilization less than 25% is considered trivially loaded. These aircraft will be unloaded; SALP-TS uses the other neighborhoods to minimize the violations due to using one fewer aircraft. Table 3.4 demonstrates that (given the requirement that no pallet may weigh less than 2500 pounds) an aircraft with every pallet position filled cannot be trivially loaded.

	Number	Trivial	Total	Planning	Percent
Aicraft	**Positions**	**Pallet Weight**	**Trivial Weight**	**ACL**	**Loaded**
C-130 E/H	6	2500	15000	25,000	**60.00%**
C-141	13	2500	32500	46,000	**70.65%**
C-17: Logistics	18	2500	45000	90,000	**50.00%**
C-17: Airdrop	11	2500	27500	90,000	**30.56%**
C-5	36	2500	90000	150,000	**60.00%**
KC-10A: 17 Pallets	16	2500	40000	80,000	**50.00%**
KC-10A: 23 Pallets	22	2500	55000	80,000	**68.75%**
KC-135E	6	2500	15000	30,000	**50.00%**
KC-135R	6	2500	15000	30,000	**50.00%**

Table 3.4 Every Pallet Position Occupied Using Trivially Loaded Pallets

(2) If 15 adjacent iterations result in trivially improving moves *and* the number of aircraft in use is not less than the lower bound, the algorithm has stabilized on an unproductive plateau. Unloading an entire aircraft constitutes a diversification strategy which can lead to better solutions.

(3) If 15 disimproving moves have been performed *and* the number of aircraft in use is not less than the lower bound, diversification is indicated to escape from a nonproductive region of the solution space. Moves of this type result in large changes to the OF value by removing an aircraft from consideration.

State 4

If none of the previous states apply, the *Inter-Aircraft Insert/Swap* neighborhood will be used. These moves result in mid-sized objective function changes and are frequently used throughout SALP-TS.

3.2.9 Solution Output

Upon completion of the algorithm, up to nine solutions are reported. This supports the philosophy that SALP-TS is a *decision making aid* and, as such, should allow the decision maker to select a preferred solution for a given situation by utilizing his/her experience and intuition and information not necessarily embodied in the SALP-TS model. The solutions are categorized by three group types: (1) Feasible, (2) Trivially Infeasible and (3) Marginally Infeasible.

The best (in terms of OF value) three solutions from each group type are presented. The first group of solutions satisfies all constraints. The next two groups of solutions are infeasible, but the benefits of these violations may enable the decision makers to choose them over a strictly feasible solution. Trivially infeasible solutions allow the total loaded pallet weight of one or more aircraft to exceed the aircraft ACL by no more than 1.5%, while marginally infeasible solutions can exceed the aircraft ACL by no more than 2.5%. Solutions with larger violations of the ACL are not presented. The level of allowable violations are not constant; they can be changed to meet the requirements set forth by the decision maker.

If solutions of these types are encountered during the search, the best three of each type will be presented as output. It is possible during the search process that SALP-TS may not encounter one or more type of solutions. For example, the pallet weights in a data set may be such that it is not possible to have ACL violations less than 1.5%. In this case, no trivially infeasible solutions will be presented as output.

3.3 COMPUTATIONAL RESULTS

To demonstrate its efficiency and effectiveness, SALP-TS was directly compared with AALPS on twelve different scenarios. The scenario descriptions and the comparative results are presented in the following sections.

3.3.1 Scenario Description

Each scenario has three factors which are allowed to vary: Pallet Set, Number of Pallets, and Type of Aircraft. There are two types of pallet sets. The first has an equal number of pallets weighing 2500, 4500, 7500, and 10,000 pounds. These were selected because AALPS is configured to allow an AALPS user to quickly select pallets with these weights. Additionally, PPs with multiple items of identical dimensions can be more difficult to solve than those with unique dimensions, and this data set tests SALP-TS's ability to successfully solve this problems. The second pallet set has pallets of random weights between 2500 and 10,000 pounds. The pallets sets contain either 500 or 1000 pallets. Two types of aircraft (the C-17 and C-5) were used in these scenarios. Although SALP-TS can accommodate nine different configurations of six different military airlift aircraft, only the principle military *strategic* airlifters were incorporated into the scenarios generated here; the other aircraft are more frequently used as tactical airlifters or air refuelers. The ACL for each of these aircraft was set to the planning ACL value. The twelve scenarios are shown in Table 3.5.

Pallet Description	Number of Pallets	Type of Aircraft	Label Name
Equal Distribution	500	C-17	Scenario 1
Random Weights	500	C-17	Scenario 2
Equal Distribution	1000	C-17	Scenario 3
Random Weights	1000	C-17	Scenario 4
Equal Distribution	500	C-5	Scenario 5
Random Weights	500	C-5	Scenario 6
Equal Distribution	1000	C-5	Scenario 7
Random Weights	1000	C-5	Scenario 8
Equal Distribution	500	C-17 & C-5	Scenario 9
Random Weights	500	C-17 & C-5	Scenario 10
Equal Distribution	1000	C-17 & C-5	Scenario 11
Random Weights	1000	C-17 & C-5	Scenario 12

Table 3.5 Scenarios for SALP-TS Testing

3.3.2 Results

The single AALPS solution and nine SALP-TS solutions are presented for each scenario. Since AALPS requires negligible time (less than 1 second), no time is presented for it. The required time to locate a solution for SALP-TS are presented. Each scenario result section contains a fourteen row chart, with row descriptions as follows:

1. Solution Type—Feasible, Trivially Infeasible, or Marginally Infeasible.
2. # Aircraft in LB— Lower bound on number of aircraft required.
3. # Aircraft in AALPS—Number of aircraft required by AALPS for feasible solution.
4. AALPS OF Value—The OF value of the AALPS solution.
5. # Aircraft in Initial—Number of aircraft required by SALP-TS in initial solution.
6. # Aircraft in SALP-TS—Number of aircraft required by SALP-TS solution.
7. Achieve LB?—“No” if SALP-TS solution required more aircraft than LB, “Yes” if SALP-TS solution required the same number of aircraft as LB, and “Below” if SALP-TS solution used fewer aircraft than LB.

8. # Aircraft Reduction—Reduction in aircraft for SALP-TS solution when compared to the AALPS solution.
9. % Aircraft Reduction—The percent reduction in the number of aircraft required per solution when the SALP-TS solution is compared to the AALPS solution. It is given by:

$$\text{\% Aircaft Reduction} = \frac{\text{\# Aircraft Reduction}}{\text{\# Aircraft Required in AALPS}}$$

Equation 3.11: SALP-TS % Aircraft Reduction

10. $ Saved by Reduction = The amount of money saved by reducing the required number of planes and is given by:

$$\sum_{j \in M} A_j C_j$$

Equation 3.12: SALP-TS $ Saved by Reduction

11. # Aircraft Infeasible—Number of aircraft exceeding ACL.
12. SALP-TS OF Value—The OF value of the SALP-TS solution.
13. % OF Reduction— The percent reduction in the OF value when the SALP-TS solution is compared to the AALPS solution. It is given by:

$$\text{\% OF Reduction} = \frac{(\text{AALPS OFValue - [SALP - TS OF Value]})}{\text{AALPS OF Value}}$$

Equation 3.13: SALP-TS % OF Reduction

14. CPU Time (sec)—The time (in seconds) required for SALP-TS to find this solution.

Row 10 contains the amount (in dollars) saved by using the SALP-TS solution over the AALPS solution. In this calculation, A_j is 1 if aircraft j is used by AALPS but not by SALP-TS and 0 otherwise, and C_j is the cost of flying aircraft j on an average overseas transportation mission. The value for C_j is computed using estimates provided by AMC. The average cost per flight hour of an AMC C-17 is \$17,445, while a C-5 is \$29,106. The average flying time required for a channel mission is not known, but for the purposes of this dissertation, the AMC estimate of 15 hours one-way (30 hours round trip) is used. Using these values, the cost of each round-trip channel mission is \$523,350 for the C-17 and \$873,180 for the C-5.

As was described in Section 3.2.9, SALP-TS saves the three best solutions encountered in each of three different solution types. In some scenarios, the three solutions of a particular time may be nearly identical. This results from small changes (1 or two pallets) within a solution. For brevity, only the best solution found in each solution type is presented in the following sections.

The penalty values used in the objective function in each of the scenarios are listed in Table 3.6. These values are based upon knowledge of the problem and experimentation.

Penalty Description:	Value:
Big Bin =	10,000
Aircraft % Weight Full =	1
Over Weight =	30
Longitudinal CB Location =	0.25
Latitudinal CB Location =	0.25
CB Out of Window =	200
C-17 Aircraft Usage =	5000
C-5 Aircraft Usage =	5000

Table 3.6 Penalty Multiplier Values for SALP Scenarios

3.3.2.2 *Scenario 1*

The results for scenario 1 are presented in Table 3.7

Sol'n Type	Feasible	Marginal
# Aircraft in LB	35	35
# Aircraft in AALPS	39	39
AALPS OF Value	246545.80	246545.80
# Aircraft in Initial	35	35
# Aircraft SALP-TS	35	34
Achieve LB?	Yes	Below
# Aircraft Reduction	4	5
% Aircraft Reduction	10.26%	12.82%
$ Saved by Reduction	$2,093,400	$2,616,750
# Aircraft Infeasible	0	3
SALP-TS OF Value	183675.31	170230.20
% OF Reduction	25.50%	30.95%
CPU Time (sec)	0.922	2.172

Table 3.7 SALP-TS Scenario 1 Results

AALPS required 39 aircraft. SALP-TS produced feasible solutions which achieved the LB by reducing the required number of aircraft by 10.26% and the cost by $2,093,400. SALP-TS produced no trivially infeasible solutions, but did produce marginally infeasible solutions requiring 34 aircraft, *less than* the LB, by allowing 3 aircraft to exceed their ACL by no more than 2.5%, yielding an aircraft reduction 12.82% and a potential cost reduction of $2,616,750.

The reduction in the OF values of the SALP-TS solutions over the AALPS solutions is even more impressive than the simple plane reduction. The OF value considers not only the number of planes used in a solution (the Aircraft Usage Fee), but also the CB location penalties and percent weight full penalty. The OF value reductions for this scenario (25.50% and 30.95%) are due to the corrections in the lateral and longitudinal CB, the reduction in the number of required aircraft, and the increase (across all aircraft) in the percent weight full. Since all pallets are loaded in all the solutions presented by SALP-TS, the unloaded pallet penalty is 0 in every case. The CPU times required were 0.922 and 2.172 seconds for feasible and marginally infeasible solutions, respectively.

3.3.2.3 *Scenario 2*

The results for scenario 2 are presented in Table 3.8.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	35	35	35
# Aircraft in AALPS	37	37	37
AALPS OF Value	217382.79	217382.79	217382.79
# Aircraft in Initial	35	35	35
# Aircraft SALP-TS	35	34	34
Achieve LB?	Yes	Below	Below
# Aircraft Reduction	2	3	3
% Aircraft Reduction	5.41%	8.11%	8.11%
$ Saved by Reduction	$1,046,700	$1,570,050	$1,570,050
# Aircraft Infeasible	0	15	15
SALP-TS OF Value	175454.36	170546.91	170551.63
% OF Reduction	19.29%	21.55%	21.54%
CPU Time (sec)	11.203	19.093	18.875

Table 3.8 SALP-TS Scenario 2 Results

AALPS required 37 aircraft. SALP-TS produced feasible loads achieving the LB and reducing the required aircraft by 5.41% and potentially reducing the cost by $1,046,700. SALP-TS also further reduced the required number of aircraft to 34 (1 aircraft *below* the lower bound) by allowing 15 aircraft to slightly exceed their ACL. This is a total reduction in aircraft of 8.11% and potential cost reduction of $1,570,050. Additionally, SALP-TS reduced the overall OF value by 19.29%, 21.55% and 21.54%, respectively, for the feasible, trivially infeasible, and marginally infeasible solutions. The time required to produce feasible solutions was 11.203 seconds, while 19.093 seconds were required for the trivially infeasible solutions. The best marginally infeasible solution was found in 18.875 seconds.

3.3.2.4 *Scenario 3*

The results for scenario 3 are presented in Table 3.9.

Sol'n Type	Feasible	Marginal
# Aircraft in LB	69	69
# Aircraft in AALPS	77	77
AALPS OF Value	474011.34	474011.34
# Aircraft in Initial	69	69
# Aircraft SALP-TS	69	68
Achieve LB?	Yes	Below
# Aircraft Reduction	8	9
% Aircraft Reduction	10.39%	11.69%
$ Saved by Reduction	$4,186,800	$4,710,150
# Aircraft Infeasible	0	5
SALP-TS OF Value	355052.14	340357.91
% OF Reduction	25.10%	28.20%
CPU Time (sec)	3.406	13.047

Table 3.9 SALP-TS Scenario 3 Results

AALPS required 77 aircraft. SALP-TS produced feasible solutions achieving the LB and reducing the required number of aircraft by 10.39% and potentially reducing the cost by $4,186,800. SALP-TS did not produce any trivially infeasible solutions, but it did produce a marginally infeasible solution which only required 68 aircraft, one *below* the lower bound. By allowing 5 aircraft to exceed their ACL, SALP-TS required 9 fewer aircraft (11.69%) than AALPS, which produces a potential cost savings of $4,710,150. SALP-TS reduced the overall OF value by 25.10% and 28.20% for the feasible and marginally infeasible solutions, respectively. The CPU time required to achieve these results were 3.406 seconds for the feasible solutions and 13.047 seconds for the marginally infeasible solutions.

3.3.2.5 *Scenario 4*

The results for scenario 4 are presented in Table 3.10.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	70	70	70
# Aircraft in AALPS	75	75	75
AALPS OF Value	429803.24	429803.24	429803.24
# Aircraft in Initial	72	72	72
# Aircraft SALP-TS	71	71	70
Achieve LB?	No	No	Yes
# Aircraft Reduction	4	4	5
% Aircraft Reduction	5.33%	5.33%	6.67%
$ Saved by Reduction	$2,093,400	$2,093,400	$2,616,750
# Aircraft Infeasible	0	1	14
SALP-TS OF Value	355573.65	355573.16	350929.63
% OF Reduction	17.27%	17.27%	18.35%
CPU Time (sec)	80.390	79.125	110.437

Table 3.10 SALP-TS Scenario 4 Results

AALPS required 75 aircraft. SALP-TS produced feasible loads which reduced the required number of aircraft by 4 or 5.33% for a potential cost reduction of $2,093,400. SALP-TS did not produce any trivially infeasible solutions which required fewer aircraft than the feasible solutions, but the marginally infeasible solutions achieved the lower bound by reducing the number of required aircraft by 5 or 6.67% while allowing 14 aircraft to exceed their ACL. This creates a potential cost savings of $2,616,750. Additionally, SALP-TS reduced the overall OF value by 17.27%, 17.27% and 18.35% over the AALPS solution for the feasible, trivially infeasible, and marginally infeasible solutions.

The CPU time required to achieve best feasible and trivially infeasible solutions were 80.390 and 79.125 seconds, while 110.437 seconds were required for the best marginally infeasible solution.

3.3.2.6 *Scenario 5*

The results for scenario 5 are presented in Table 3.11.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	21	21	21
# Aircraft in AALPS	22	22	22
AALPS OF Value	172798.91	172798.91	172798.91
# Aircraft in Initial	21	21	21
# Aircraft SALP-TS	21	21	21
Achieve LB?	Yes	Yes	Yes
# Aircraft Reduction	1	1	1
% Aircraft Reduction	4.55%	4.55%	4.55%
$ Saved by Reduction	$873,180	$873,180	$873,180
# Aircraft Infeasible	0	1	1
SALP-TS OF Value	105258.56	105279.04	105362.00
% OF Reduction	39.09%	39.07%	39.03%
CPU Time (sec)	13.094	13.110	13.141

Table 3.11 SALP-TS Scenario 5 Results

AALPS required 22 aircraft. SALP-TS produced feasible loads which achieved the lower bound by reducing the number of aircraft by 1 or 4.55% for a potential cost savings of $873,180. SALP-TS did not find any trivially or marginally infeasible solutions which used fewer aircraft than the feasible solutions. The time required to achieve the best feasible, trivially infeasible, and marginally infeasible solutions was 13.094, 13.110, and 13.141 seconds, respectively. In this scenario, the feasible solutions demonstrated the greatest reduction in OF Value. Since the number of aircraft required is the same for all three solution types and the OF value is lower, the logical choices for this scenario are the feasible solutions.

3.3.2.7 *Scenario 6*

The results for scenario 6 are presented in Table 3.12.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	21	21	21
# Aircraft in AALPS	22	22	22
AALPS OF Value	461823.74	461823.74	461823.74
# Aircraft in Initial	21	21	21
# Aircraft SALP-TS	21	21	20
Achieve LB?	Yes	Yes	Below
# Aircraft Reduction	1	1	2
% Aircraft Reduction	4.55%	4.55%	9.09%
$ Saved by Reduction	$873,180	$873,180	$1,746,360
# Aircraft Infeasible	0	1	20
SALP-TS OF Value	105258.56	105229.86	101920.65
% OF Reduction	77.21%	77.21%	77.93%
CPU Time (sec)	13.078	13.265	25.187

Table 3.12 SALP-TS Scenario 6 Results

AALPS required 22 aircraft. SALP-TS produced feasible and trivially infeasible loads which achieved the lower bound by reducing the number of aircraft by 1 or 4.55%. This allows for a potential cost savings of $873,180. SALP-TS also further reduced the required number of aircraft to 20 (1 aircraft *below* the lower bound) by allowing 20 aircraft to slightly exceed their ACL. This is a total reduction of 2 aircraft or 9.09% and a potential cost reduction of $1,746,360.

SALP-TS reduced the overall OF value by a range of 77.21% to 77.93%. This is a relatively large reduction, compared to other scenarios. This scenario involves only C-5 aircraft. The C-5 aircraft differs from other AF cargo aircraft because the target longitudinal CB location is *not necessarily* in the center of the acceptable region. In fact, the target CB location could actually be the boundary of the window for a specific cargo weight (see Appendix E). While AALPS only provides feasible loads which have the longitudinal CB *within* the window, SALP-TS attempts to find solutions in which the longitudinal CB of each aircraft is close to the target value. This accounts for the relatively large reduction in the OF value over the AALPS solution. The time required to achieve feasible and trivially infeasible solutions was 13.078 and 13.265 seconds, respectively, while 25.187 seconds was required to find the best marginally infeasible solution.

3.3.2.8 *Scenario 7*

The results for scenario 7 are presented in Table 3.13.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	41	41	41
# Aircraft in AALPS	44	44	44
AALPS OF Value	343126.20	343126.20	343126.20
# Aircraft in Initial	41	41	41
# Aircraft SALP-TS	41	41	41
Achieve LB?	**Yes**	**Yes**	**Yes**
# Aircraft Reduction	**3**	**3**	**3**
% Aircraft Reduction	**6.82%**	**6.82%**	**6.82%**
$ Saved by Reduction	**$2,619,540**	**$2,619,540**	**$2,619,540**
# Aircraft Infeasible	**0**	**1**	**1**
SALP-TS OF Value	205082.49	205113.29	205169.15
% OF Reduction	**40.23%**	**40.22%**	**40.21%**
CPU Time (sec)	40.828	33.453	40.312

Table 3.13 SALP-TS Scenario 7 Results

AALPS required 44 aircraft. SALP-TS produced feasible loads which achieved the lower bound by reducing the number of aircraft by 3 or 6.82% with a potential cost savings of $2,619,540. SALP-TS did not find any trivially or marginally infeasible solutions which used fewer aircraft than the feasible solutions. The time required to achieve the best feasible, trivially infeasible, and marginally infeasible solutions was 40.828, 33.453, and 40.312 seconds, respectively. In this scenario, the feasible solutions demonstrated the greatest reduction in OF Value. Since the number of aircraft required is the same for all three solution types and the OF value is lower, the logical choices for this scenario are the feasible solutions.

3.3.2.9 *Scenario 8*

The results for scenario 8 are presented in Table 3.14.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	42	42	42
# Aircraft in AALPS	43	43	43
AALPS OF Value	328098.41	328098.41	328098.41
# Aircraft in Initial	43	43	43
# Aircraft SALP-TS	43	42	42
Achieve LB?	**No**	**Yes**	**Yes**
# Aircraft Reduction	**0**	**1**	**1**
% Aircraft Reduction	**0.00%**	**2.33%**	**2.33%**
$ Saved by Reduction	**$0**	**$873,180**	**$873,180**
# Aircraft Infeasible	**0**	**6**	**7**
SALP-TS OF Value	215543.57	210059.59	210136.82
% OF Reduction	**34.31%**	**35.98%**	**35.95%**
CPU Time (sec)	55.797	96.406	98.031

Table 3.14 SALP-TS Scenario 8 Results

AALPS required 22 aircraft. SALP-TS did not produce any feasible solutions which used fewer aircraft than AALPS. SALP-TS produced trivially and marginally infeasible solutions which achieved the lower bound (reducing the required number of aircraft by 1 or 2.33% with a potential cost savings of $873,180) by allowing 6 and 7 aircraft, respectively, to exceed their ACL. Although it did not reduce the number of required aircraft, SALP-TS produced feasible solutions which reduced the overall OF value by 34.31% by improving the CB locations. The trivially and marginally infeasible solutions reduced the OF value by 35.98% and 35.95%, respectively. The time required to achieve the best feasible solution was 55.797 seconds, while 96.406 and 98.031 were required to achieve the best trivially and marginally infeasible solutions.

3.3.2.10 *Scenario 9*

The results for scenario 9 are presented in Table 3.15.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	27	27	27
# Aircraft in AALPS	28	28	28
AALPS OF Value	190318.31	190318.31	190318.31
# Aircraft in Initial	27	27	27
# Aircraft SALP-TS	27	26	26
Achieve LB?	Yes	Below	Below
# Aircraft Reduction	1	2	2
% Aircraft Reduction	3.57%	7.14%	7.14%
$ Saved by Reduction	$873,180	$1,746,360	$1,746,360
# Aircraft Infeasible	0	2	2
SALP-TS OF Value	145954.86	130118.00	130146.93
% OF Reduction	23.31%	31.63%	31.62%
CPU Time (sec)	0.703	1.969	2.141

Table 3.15 SALP-TS Scenario 9 Results

AALPS required 28 aircraft: 14 C-17s and 14 C-5s. The lower bound requires 14 C-17s and 13 C-5s. Since this scenario involves multiple aircraft types, SALP-TS reduces the aircraft which have the smallest ratio of ACL to number of available pallet positions. For scenarios 9-12, the C-5 (with ACL = 150,000 pounds and 36 pallet positions) has the smallest ratio at *(150,000 / 36) = 4166.67 pounds per position*. This implies that all reduced planes in these scenarios will be C-5 aircraft. SALP-TS produced feasible solutions which achieved the lower bound by reducing 1 C-5 aircraft or 3.57%. This allows for a potential cost reduction of $873,180. SALP-TS also produced trivially and marginally infeasible solutions which only required 14 C-17s and 12 C-5s. This is *below* the lower bound and is a reduction of 2 C-5 aircraft or 7.14% by allowing 2 aircraft to exceed their ACL. This could potentially reduce the required cost by $1,746,360. SALP-TS reduced the overall OF value by 23.31%, 31.63%, and 31.62% for the feasible, trivially infeasible, and marginally infeasible solutions, respectively. The CPU time required to achieve the best feasible result was 0.703 seconds; 1.969 and 2.141 seconds were required to achieve the best trivially and marginally infeasible results.

3.3.2.11 *Scenario 10*

The results for scenario 10 are presented in Table 3.16.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	26	26	26
# Aircraft in AALPS	27	27	27
AALPS OF Value	177019.33	177019.33	177019.33
# Aircraft in Initial	27	27	27
# Aircraft SALP-TS	27	26	26
Achieve LB?	No	Yes	Yes
# Aircraft Reduction	0	1	1
% Aircraft Reduction	0.00%	3.70%	3.70%
$ Saved by Reduction	$0	$873,180	$873,180
# Aircraft Infeasible	0	10	10
SALP-TS OF Value	135743.43	130034.38	130104.64
% OF Reduction	23.32%	26.54%	26.50%
CPU Time (sec)	13.312	23.765	23.859

Table 3.16   SALP-TS Scenario 10 Results

AALPS required 27 aircraft: 14 C-17s and 13 C-5s. The lower bound requires 14 C-17s and 12 C-5s. SALP-TS did not produce any feasible solutions which used fewer aircraft than AALPS, but it produced trivially and marginally infeasible solutions which achieved the lower bound by allowing 12 aircraft to exceed their ACL. The reduction of 1 C-5 aircraft could potentially lower the cost by $873,180. Although it did not reduce the number of planes required in feasible solutions, SALP-TS reduced the overall OF value by 23.32%. SALP-TS reduced the OF value for the trivially and marginally feasible solutions by 26.54% and 26.50%, respectively. The time required to achieve the best feasible solution was 13.312 seconds, while 23.765 and 23.859 were required for the best trivially and marginally infeasible solutions.

3.3.2.12 *Scenario 11*

The results for scenario 11 are presented in Table 3.17.

Sol'n Type	Feasible	Marginal
# Aircraft in LB	53	53
# Aircraft in AALPS	56	56
AALPS OF Value	385202.80	385202.80
# Aircraft in Initial	53	53
# Aircraft SALP-TS	53	52
Achieve LB?	Yes	Below
# Aircraft Reduction	3	4
% Aircraft Reduction	5.36%	7.14%
$ Saved by Reduction	$2,619,540	$3,492,720
# Aircraft Infeasible	0	3
SALP-TS OF Value	275904.35	260214.24
% OF Reduction	28.37%	32.45%
CPU Time (sec)	3.578	67.781

Table 3.17 SALP-TS Scenario 11 Results

AALPS required 56 aircraft: 28 C-17s and 28 C-5s. The lower bound requires 53 aircraft: 28 C-17s and 25 C-5s. SALP-TS produced feasible solutions which achieved the lower bound by reducing the required number of aircraft by 3 or 5.36%. This creates a potential cost savings of $2,619,540. SALP-TS did not produce any trivially infeasible solutions, but it did produce marginally infeasible solutions which only required 52 (28 C-17s and 24 C-5s), which is *below* the lower bound. To achieve this, SALP-TS allowed 3 aircraft to exceed their ACL, causing a reduction of 4 aircraft or 7.14%. A reduction of 4 aircraft could potentially lower the cost by $3,492,720. SALP-TS reduced the overall OF value by 28.37% and 32.45% for the feasible and marginally infeasible solutions. The time required to achieve the best feasible solution was 3.578 seconds; 67.781 seconds were required for the best marginally infeasible solution.

3.3.2.13 *Scenario 12*

The results for scenario 12 are presented in Table 3.18.

Sol'n Type	Feasible	Trivial	Marginal
# Aircraft in LB	53	53	53
# Aircraft in AALPS	55	55	55
AALPS OF Value	386999.38	386999.38	386999.38
# Aircraft in Initial	54	54	54
# Aircraft SALP-TS	54	53	53
Achieve LB?	No	Yes	Yes
# Aircraft Reduction	1	2	2
% Aircraft Reduction	1.82%	3.64%	3.64%
$ Saved by Reduction	$873,180	$1,746,360	$1,746,360
# Aircraft Infeasible	0	19	17
SALP-TS OF Value	270460.43	265136.23	265200.80
% OF Reduction	30.11%	31.49%	31.47%
CPU Time (sec)	49.250	90.266	89.875

Table 3.18 SALP-TS Scenario 12 Results

AALPS required 55 aircraft: 28 C-17s and 27 C-5s. The lower bound requires 53 aircraft: 28 C-17s and 25 C-5s. SALP-TS did not produce any feasible solutions which achieved the lower bound, but, when compared to AALPS, the feasible solutions reduced 1 C-5 aircraft (1.82%) and the overall OF value by 30.11%. The reduction leads to a potential cost savings of $873,180. SALP-TS produced trivially and marginally infeasible solutions which achieved the lower bound by reducing the number of required aircraft by 2 C-5s or 3.64% after allowing 19 and 17 aircraft to exceed their ACL. This could potentially reduce the cost by $1,746,360. The trivially and marginally infeasible solutions reduced the overall OF value by 31.49% and 31.47%. The time required to achieve the best feasible solution was 49.250 seconds, while 90.266 and 89.875 seconds were required for the best trivially and marginally infeasible solutions.

3.3.2.14 *Results Summary*

SALP-TS and AALPS were directly compared using twelve different scenarios. In nine of the twelve scenarios, the SALP-TS initial solution generator produced solutions which required fewer aircraft than AALPS. While these initial solutions are feasible with respect to weight and spatial constraints, they are not necessarily feasible with respect to CB. The TS portion of SALP-TS is utilized to locate solutions which are CB feasible. Additionally, in ten of the twelve scenarios, SALP-TS produced trivially infeasible and/or marginally infeasible solutions which required fewer aircraft than the solution produced by the SALP-TS initial solution generator.

The comparison results with respect to number of planes reduced by SALP-TS over AALPS are presented in Figure 3.7. In eight of the scenarios, SALP-TS produced feasible results which achieved the lower bound, while AALPS required more aircraft than the lower bound in every scenario. In two of the four scenarios in which it did not achieve the lower bound, SALP-TS required fewer aircraft than AALPS. In the remaining two scenarios (8 and 10), SALP-TS required the same number of planes as AALPS. Overall, SALP-TS produced feasible solutions which reduced the number of aircraft required by 0 to 8 aircraft with an average of 2.33 aircraft.

In all scenarios in which SALP-TS found trivially infeasible solutions, the solutions required fewer planes than the AALPS solution. The problem construction prevented SALP-TS from finding any trivially infeasible solutions in three of the twelve scenarios (1, 3, and 11)—this is indicated by the asterisks (*)

on Figure 3.7. For trivially infeasible solutions, the aircraft are either feasible or have an ACL violation less than 1.5% of the total ACL. Of the nine scenarios with trivially infeasible solutions, SALP-TS produced results which achieved the lower bound in six scenarios and surpassed the lower bound in two scenarios. Although SALP-TS did not achieve or surpass the lower bound in one scenario, it did outperform AALPS. In the scenarios in which SALP-TS encountered trivially infeasible solutions, SALP-TS produced solutions which reduced the number of aircraft required by 1 to 4 aircraft with an average of 2.00 aircraft.

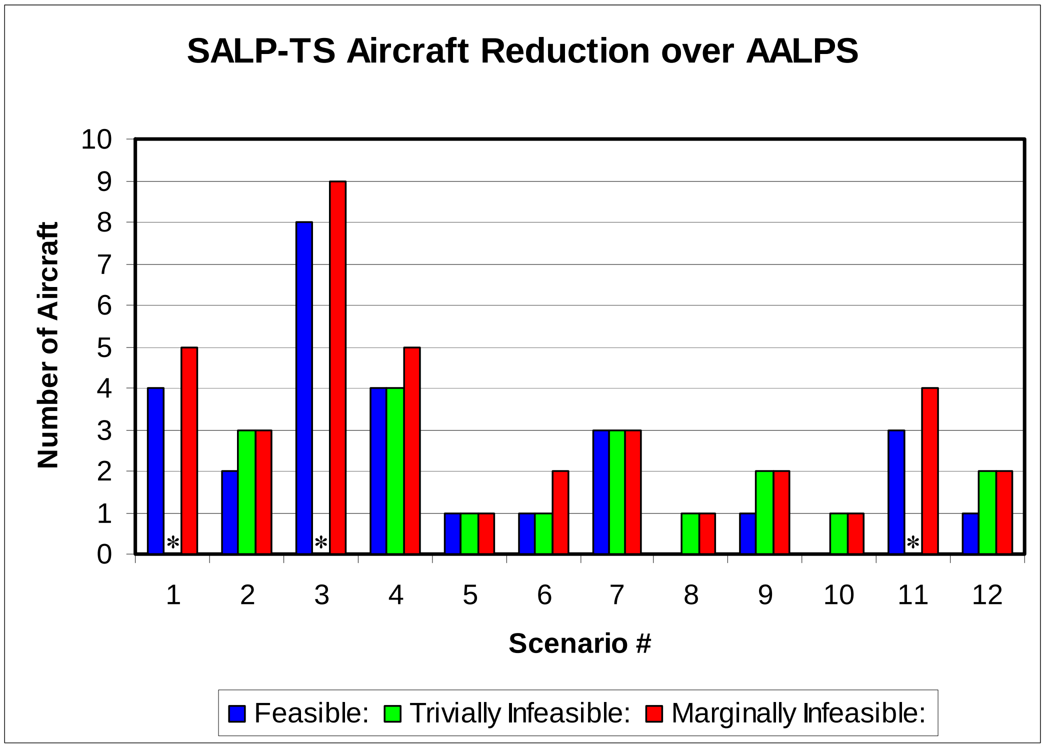

Figure 3.7 SALP-TS Aircraft Reduction over AALPS

In every scenario, SALP-TS produced marginally infeasible solutions which required fewer planes than AALPS and met or surpassed the lower bound by allowing aircraft ACL violations of less than 2.5% of the total ACL. In six of the scenarios, SALP-TS produced solutions which required fewer aircraft than the lower bound. Overall, SALP-TS produced marginally infeasible solutions which reduced the number of aircraft required by 1 to 9 aircraft with an average of 3.17 aircraft.

Another metric which demonstrates the effectiveness of SALP-TS solutions is the percentage of reduction in required aircraft of SALP-TS over AALPS. This metric is presented by scenario in Figure 3.8. SALP-TS produced feasible solutions which reduced the required aircraft by 0% to 10.39%. On average, the SALP-TS feasible solutions reduced the required number of aircraft by 4.84%.

In the scenarios in which SALP-TS encountered trivially infeasible solutions, the required number of aircraft was reduced by 2.33% to 8.11% over AALPS. On average, SALP-TS produced trivially infeasible solutions which reduced the required number of aircraft by 5.13%. SALP-TS produced marginally infeasible solutions which reduced the required number of aircraft from 2.33% to 12.82% with an average of 6.98%.

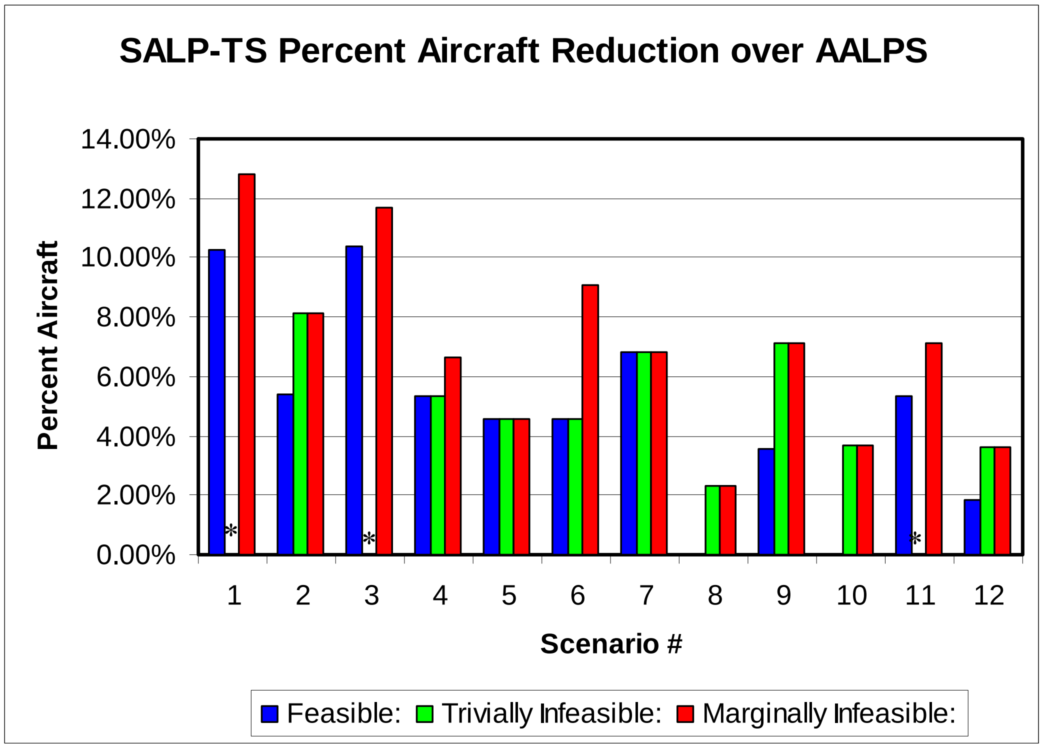

Figure 3.8 SALP-TS Percent Aircraft Reduction over AALPS

On average, SALP-TS produced *feasible* solutions which reduced the number of aircraft by approximately 5% over the AALPS solution. In fiscal year 2005, 61,796 hours of C-17 flight time were dedicated solely to channel missions. If an average round trip flight requires 30 flight hours, then (61,796 / 30) ≈ 2060 flights were planned. If the average cost of a C-17 channel mission is as previously described ($523,350), and if 5% or 103 of the 2060 C-17 flights could be eliminated without loss of throughput, a potential estimated savings of *(103 * $523,350)* = *$53,905,050* per year could be obtained without loss of effectiveness. This estimate is only for *one aircraft type*; similar savings would be anticipated for the C-5. If one were to suppose the same number of C-5 flights were saved, an additional cost savings of *(103 * $873,180) = $89,937,540* could be realized. The combined total savings from these two airframes would be approximately *$53,905,050 + $89,937,540 = $143,842,590.* This estimate also does not consider the maintenance costs or wear and tear on the aircraft. Additionally, a 5% reduction allows fewer aircraft to be purchased. The reduction in cost due to fewer aircraft purchases is also not incorporated into this cost savings estimate.

In terms of OF value, SALP-TS outperformed AALPS in every scenario. Figure 3.9 presents the percentage reduction in OF value of SALP-TS over AALPS. SALP-TS produced feasible solutions which reduced the OF value from 17.27% to 77.22 percent with an average of 31.95%. In the scenarios in which it encountered trivially infeasible solutions, SALP-TS produced solutions which reduced the OF value from 17.27% to 77.21% with an average of 35.66%. Marginally infeasible solutions were produced which reduced the OF value from 18.35% to 77.93% with an average of 34.52%.

The percentage reduction in OF value demonstrates that the SALP-TS solutions not only required fewer aircraft, but also more closely achieved the target CB location on both longitudinal and lateral CB. Significant improvements in aircraft utilization were also achieved.

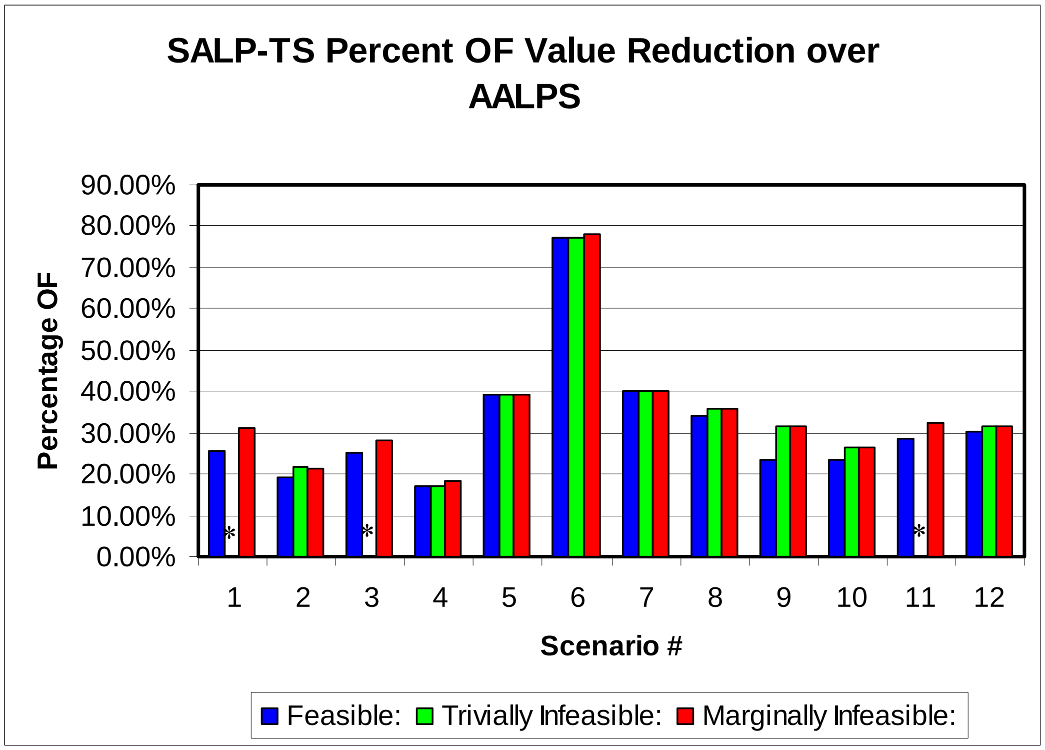

Figure 3.9 SALP-TS Percent OF Value Reduction over AALPS

3.4 SALP SUMMARY

This chapter provided a detailed description of the SALP, the solution representation utilized in this research, and the SALP-TS algorithm developed as a proof of concept to solve the problem. The solution methodology used in SALP-TS effectively and efficiently solves the SALP. In side-by-side testing, SALP-TS required fewer or the same number of aircraft as AALPS, the method currently utilized by the Air Force.

Chapter 4: A Tabu Search Approach to the Dynamic ALP

4.1 DETAILED PROBLEM STATEMENT

The DALP is an extension of the SALP and, as such, is very similar in design and construction. The DALP can be viewed as the contingency phase of military involvement in an area. The DALP considers the same portions of the ALP that are considered by the SALP: *partitioning* the set of cargo into aircraft loads, *selecting* an efficient and effective set of aircraft from an available pool of aircraft, and feasibly *placing* the cargo in the best allowable positions.

Temporal constraints are added to the SALP to generate the DALP. The temporal constraints are the acceptable "window" of days within which each pallet is supposed to reach its destination. Although it is preferable for the pallets to arrive within its specified time window, solutions are considered in which the pallet will arrive outside the desired window. By exploring and saving feasible and infeasible solutions, decision makers receive multiple solutions from which they can choose their preferred solution.

For the DALP, aircraft are not limited to one trip. The travel time from the APOE to APOD, down time at the APOD (for crew rest and refueling), and travel time from APOD to APOE are presumed to be known in advance. Each aircraft is assumed to return to its home APOE. The overall DALP goal includes minimizing not only the number of flights required to transport all pallets, but also the total number aircraft required, while at the same time minimizing aircraft ACL violations and pallet temporal violations. An aircraft that is available for a

DALP scenario but not required in the solution can be made available for other needs. Pallet temporal violations occur when a pallet's arrival date is outside of its acceptable window.

The SALP can be viewed as the sustaining or channel phase of military involvement in an area, and the DALP is the contingency or deployment phase. In the contingency phase, the military involvement in an area has just begun or a new unit is deploying to the area. To ensure that items arrive as they are required (not early or late) proper sequencing is necessary. This sequencing is detailed in a TPFDD with specified earliest and latest arrival dates for items.

4.1.1 DALP-TS Inputs

As with the SALP-TS, two tab delimited text files containing the aircraft and pallet information, easily generated using Microsoft Excel, are required for DALP-TS. Sample DALP-TS aircraft and pallet input files are presented in Appendices F and G, respectively.

4.1.1.1 *Aircraft Input File*

The aircraft input file contains: (1) a *unique* integer aircraft identification (ID) number, (2) aircraft type (integer category), (3) ACL, (4) initial ready to load date (RLD), (5) required travel time, and (6) required crew rest time at APOD. The first three fields are identical to those described in Section 3.1.1.1. The first additional field specifies the initial RLD for the aircraft—the aircraft cannot depart *prior* to this date and no pallet can be assigned to the aircraft with an available load date (ALD) *after* this date. The second additional field specifies the amount of travel time required for the aircraft to travel from the APOE to

APOD. The final additional field specifies the amount of time required for crew rest at the APOD.

4.1.1.2 *Pallet Input File*

The pallet input file contains: (1) pallet ID number, (2) loaded pallet weight, (3) loaded pallet height, (4) loaded pallet width, (5) loaded pallet length, (6) pallet ALD, (7) pallet earliest arrival date (EAD), (8) pallet latest arrival date (LAD), and (9) pallet required delivery date (RDD).

The pallet input file contains additional fields and is sorted in a different manner from the input file for SALP-TS. With SALP-TS, the pallets needed only to be sorted by decreasing weight. For DALP-TS, the pallets are first sequentially sorted by increasing ALD, EAD, LAD, and RDD (respectively), and then by decreasing weight. The order of dates corresponds to their chronological order in the TPFDD. This ensures that the heaviest pallet with the earliest ALD, EAD, LAD, and RDD is the first pallet in the input file.

4.2 METHODOLOGY

The previous section detailed the input files required for the DALP. This section details the DALP-TS methodology. The complete pseudo-code describing this algorithm is given in Appendix H.

4.2.1 DALP-TS Data Structures

The DALP-TS data structures are extensions of the SALP-TS data structures. DALP-TS uses the same three data structures which are included in SALP-TS: *aircraft*, *pallet*, and *solution*, but the aircraft object is renamed the *aircraft trip* object to account for multiple uses of the same aircraft. The additional fields required for each of the DALP-TS objects are shown in Table 4.1. These fields are *in addition to* the fields listed in Table 3.2.

Aircraft Trip	Ready to Load Date, Departure Date, Arrival Date, Trip Number, Index of Previous Trip, Index of Next Trip
Pallet	ALD, EAD, LAD, RDD
Solution	Number of Flights Flown

Table 4.1 Additional attributes for DALP-TS Data Structures

The *aircraft trip* object has six additional, changeable fields. The aircraft RLD is the *earliest* date upon which an aircraft can be loaded and depart the APOE. After the pallet load has been selected, an appropriate aircraft departure date (DD) *from the APOE* is determined. The aircraft arrival date (AD) is the date upon which the aircraft arrives *at the APOD*. It incorporates the DD and the required travel time from APOE to APOD. The aircraft DD *can* be the same as the RLD, but, for certain pallet loads, delaying the DD to a later date will reduce the amount of temporal violations.

The RLD for each aircraft's first trip is day 1. The RLD of each subsequent trip of this particular aircraft considers the previous trip's DD and the travel and down time previously described.

The trip number indexes a specific aircraft trip and details the total number of flights flown by each aircraft. The indices for the previous and next trip are used to quickly update the RLD and DD of additional aircraft trips when changes to the solution are considered.

The *pallet* object has four additional, constant fields. The ALD is strictly enforced in the algorithm. Violations of the remaining three fields are allowed but are penalized. Pallets reaching the APOD prior to the EAD may arrive before the necessary equipment and/or personnel required for cargo unloading are in place and may cause maximum-on-the-ground (MOG) violations at the APOD. The LAD and RDD are very similar and, often, are identical. The LAD is the latest date by which the pallet *should* arrive at the APOD. The RDD is the latest date by which the pallet *must* arrive at the APOD. However, for practical instances of the DALP, it is known that violations of the RDDs will occur due to insufficient resources and other unavoidable causes.

Further explanation should clarify the possible differences in these two dates. Assume that the unit requiring the pallet will depart the APOD on the RDD and suppose items on a pallet require assembly and/or other types of preparation. In this situation, arrival on or before the LAD will allow preparations to be conducted at the APOD prior to departure on the RDD.

If a pallet arrives after the RDD, receiving unit personnel must return to the APOD to collect it. This not only prevents the unit from using the items on the pallet, but also requires the APOD to store the pallet possibly causing MOG violations.

4.2.2 DALP-TS Solution Representation

In both the SALP and the DALP, the final solution is an assignment of pallets to specific positions within an aircraft. The additional constraints in the DALP are considered in the OF portion of DALP-TS. The only variation between the solution representation of DALP-TS and SALP-TS is that DALP-TS can have individual aircraft flying multiple trips.

4.2.3 Tabu Memory Structure

No changes to the SALP-TS tabu memory structure presented in Section 3.2.3 are required for use by DALP-TS.

4.2.4 Initial Solution Generator

DALP-TS attempts to efficiently produce a good initial solution by inserting pallets into aircraft with arrival dates within the pallet's acceptable window and with sufficient remaining space and weight capacity. At the time of this research, no software or methodology was used by the US military to generate solutions to a DALP. As a result, a methodology was developed in this research which is an analogous extension of the AALPS loading procedure applied to a DALP.

As detailed in Section 4.1.1.2, the pallets are sequentially sorted by increasing ALD, EAD, LAD, and RDD, and then by decreasing weight and reside in the Big Bin. These dates correspond to priorities which can be specified in AALPS. Unfortunately, AALPS can only accommodate two priority levels: the unassigned item with the highest priority and greatest weight is selected for assignment. The DALP temporal constraints represent more than two priority levels. Items are distinguished by ALD, EAD, LAD, RDD, and weight. The methodology developed in the initial solution generator is an extension of the AALPS solution process which incorporates these additional priority levels.

The DALP-TS initial solution generator first selects the pallet at the head of the BigBin. Next, the best suitable aircraft is selected. There are three possible states of *feasible* (i.e. no temporal or weight violations) aircraft availability for every pallet: 1) At least one non-empty aircraft (i.e., with one or more pallets loaded) is presently at the APOE with feasible temporal constraints and sufficient weight and space capacity remaining to support the pallet. In this case, the aircraft with the largest ratio of ACL to number of pallet positions and the smallest weight capacity remaining is selected. 2) At least one empty aircraft is available which has *already made* at least one trip and has an RLD which will allow the pallet to arrive on or before its LAD. In this case, the aircraft with the largest ratio of ACL to number of pallet positions and the largest trip number is selected. 3) No additional trip of an aircraft is present that can satisfy the pallet's temporal window requirements. For state 3, a previously unused aircraft is selected.

After the aircraft is selected, the pallet is removed from the BigBin and assigned to the first available position in the aircraft. If the current state is 1, no changes to the aircraft DD or solution object are required.

If the current state is 2 or 3, changes are required. First, the number of trips required in the solution object must be incremented. The aircraft DD and AD are updated to reflect the new temporal constraints. The aircraft AD is set to be the pallet LAD, and the aircraft DD is adjusted to ensure arrival by the AD.

An additional aircraft trip must also be inserted into the solution. The RLD of this trip incorporates the previous trip's DD and total travel time. For this additional trip, the aircraft is empty but available for future loading. If the initial solution generator is in state 3, the aircraft is being used for the first time, and the aircraft-used counter in the solution object is incremented.

The DALP-TS initial solution generation process ignores lateral or longitudinal CB and only considers weight and temporal constraints. As described in Section 4.2.7, the lateral and longitudinal CB are adjusted as the first step in the dynamic neighborhood selection process of DALP-TS.

Proper departure date selection is important for problems containing time windows. Next, two examples are presented demonstrating possible departure date selections.

4.2.4.1 *Example: Initial Solution Generator using LAD*

The benefits of having the initial solution generator set the aircraft DD (and equivalently the AD) as late as possible while avoiding pallet temporal violations (i.e. the pallet LAD) can be demonstrated by an example. Suppose Table 4.2 contains a partial listing of pallets remaining in the Big Bin. Pallets with earlier ALDs, EADs, LADs, and RDDs have already been assigned to aircraft. For this example, an aircraft trip requires one day for travel to an APOD from an APOE, one day for unloading and crew rest, and one day for travel from the APOD to the APOE. As a result, an aircraft trip RLD will be four days after the previous trip's DD.

Pallet ID Num	Weight	ALD	EAD	LDD	RDD
264	10000	12	13	15	15
265	7000	12	13	15	15
266	5500	12	13	15	15
267	5000	12	13	15	15
268	4500	12	13	15	15
269	2500	12	13	15	15
270	2500	12	13	15	15
271	10000	13	14	16	16
272	10000	13	14	16	16
273	9000	13	14	16	16
274	9000	13	14	16	16
275	8500	13	14	16	16
276	8500	13	14	16	16
277	5900	13	14	16	16
278	4500	13	14	16	16
279	4500	13	14	16	16
280	4500	13	14	16	16
281	3500	13	14	16	16
282	3500	13	14	16	16
283	3500	13	14	16	16
284	3000	13	14	16	16
285	3000	13	14	16	16
286	2500	13	14	16	16
287	2500	13	14	16	16
288	2500	13	14	16	16
289	10000	15	19	22	22
290	7500	15	19	22	22
291	7500	15	19	22	22
292	7500	15	19	22	22
…	…	…	…	…	…

Table 4.2 Unassigned Pallets

Furthermore, suppose only C-17 aircraft (with 90,000 pounds ACL and 18 pallet positions) are available for loading. Table 4.3 details the information for aircraft trips currently available in the solution. Since this example does not start at the beginning of the initial solution generation process, aircraft have already been assigned pallets. The trips with a departure date greater than 0 have been assigned pallets. For those trips which have not yet been assigned pallets, the DD, AD, Total Number of Pallets, and Total Cargo Weight are 0, and the RLD reflects the DD and travel time of the previous trip.

ID Number	Trip Number	Ready to Load Date	Departure Date	Arrival Date	Total Number Pallets	Total Cargo Weight
10001	1	1	2	3	18	78500
10002	1	1	2	3	5	12500
10003	1	1	6	7	18	61750
10004	1	1	8	9	18	62200
10005	1	1	7	8	18	90000
10006	1	1	0	0	0	0
10007	1	1	0	0	0	0
10008	1	1	0	0	0	0
10009	1	1	0	0	0	0
10010	1	1	0	0	0	0
10001	2	6	6	7	12	89800
10002	2	6	6	7	18	78100
10001	3	10	13	14	18	78500
10002	3	10	13	14	18	69300
10003	2	10	14	15	18	62000
10004	2	12	0	0	0	0
10005	2	11	0	0	0	0
10001	4	17	0	0	0	0
10002	4	17	0	0	0	0
10003	3	18	0	0	0	0

Table 4.3 Aircraft Information

The head pallet (264) in the BigBin is selected for assignment. For this pallet, the acceptable window of arrival at the APOD is day 13-15, inclusive. The initial solution generator first attempts to select a *non-empty* aircraft with sufficient space and weight capacity remaining and an acceptable arrival date for this pallet; however, no aircraft is available in this category. Next, the initial solution generator attempts to select an *empty* aircraft with the largest trip number which can satisfy this pallet's requirements. Two aircraft are available in this category: the second trip of aircraft 10004 and 10005. Both aircraft have the same ratio of ACL to number of pallet positions (90000 / 18 = 5000), and it is the second trip of both aircraft. As a result, the pallet is assigned to the first position in the first aircraft encountered (10004).

The information of this aircraft trip is updated to reflect the assignment of this pallet. The trip's AD is set to correspond with the pallet's LAD (15). The trip's DD is computed by subtracting the required travel time of one day from the AD. The total number of pallets and total cargo weight are also updated to include the pallet's assignment. Additionally, since this aircraft trip is receiving its first pallet assignment, the next aircraft trip must be established. Trip 3 of aircraft 10004 will have an RLD of 18 with the remaining fields initialized to 0.

The acceptable arrival window of pallet 265 is also day 13-15, inclusive. A non-empty aircraft with sufficient space and weight capacity remaining and an acceptable arrival date for this pallet is selected. The second trip of aircraft 10004 is now in this category. The pallet is assigned to the first available position (position 2) and the total number of pallets and total cargo weight are updated. No temporal constraints require change. This same process is applied to pallets 266-270; all are assigned to the second trip of aircraft 10004. This trip now has seven pallets and 37,000 pounds assigned to it.

For pallet 271, the acceptable arrival window at the APOE is day 14-16, inclusive. Since aircraft 10004, trip 2, has an acceptable AD (15) and sufficient weight and space capacity remaining, the pallet is assigned to this trip. In a similar fashion, pallets 272-275 are assigned to the second trip of aircraft 10004. At this point, the trip has been assigned 12 pallets and 83,500 pounds.

For pallet 276, the acceptable arrival window at the APOE is day 14-16, inclusive. Although aircraft 10004, trip 2, has an acceptable AD, it does not have sufficient weight capacity remaining to support the pallet. As a result, the initial solution generator selects an empty aircraft trip which can satisfy this pallet's requirements: aircraft 10005, trip 2. Pallet 276 is assigned to this trip, the AD is set to the pallet's LAD (16), and the DD, total number of pallets, and total cargo weight are adjusted accordingly. Additionally, since this aircraft trip is receiving its first pallet assignment, the next aircraft trip must be established. Trip 3 of aircraft 10005 will have an RLD of 19 and the remaining fields initialized to 0.

The acceptable arrival window for pallet 277 is day 14-16, inclusive. Two aircraft trips have arrival dates within this window and sufficient weight and space capacity remaining to support the pallet: trip 2 on aircraft 10004 and 10005. Both aircraft have the same ratio of ACL to number of pallet positions (90000 / 18 = 5000), and it is the second trip of both aircraft. To drive individual aircrafts to be loaded as full as possible, the aircraft trip with the *least* weight capacity remaining (trip 2 on aircraft 10004) is selected. After assigning pallet 277, this aircraft trip has 13 pallets and 89,400 pounds of cargo.

The arrival window for Pallet 278 is day 14-16, inclusive. Because aircraft 10005, trip 2, has an acceptable AD and sufficient weight and space capacity remaining, pallet 278 is assigned to it. No changes to the DD or AD are required and the total number of pallets and total cargo weight are updated. Similarly, pallets 279-288 will be assigned to aircraft 10005, trip 2. After these assignments, aircraft 1005, trip 2 has 12 pallets and 46,000 pounds of cargo assigned to it.

Pallet 289 has an arrival window of 19-22. None of the non-empty aircraft trips have an arrival date within this window, so an empty aircraft trip (aircraft 10001, trip 4) is selected. This process continues until the Big Bin is completely empty.

The assignment of pallets of 264-288 to specific aircraft is shown in Table 4.4

Aircraft 10004, Trip 2

Position Num	Pallet ID Num	Weight
1	264	10000
2	265	7000
3	266	5500
4	267	5000
5	268	4500
6	269	2500
7	270	2500
8	271	10000
9	272	10000
10	273	9000
11	274	9000
12	275	8500
13	277	5900
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
	Total Weight:	**89400**
	Aircraft RLD	**12**
	Aircraft DD	**14**
	Aircraft AD	**15**

Aircraft 10005, Trip 2

Position Num	Pallet ID Num	Weight
1	276	8500
2	278	4500
3	279	4500
4	280	4500
5	281	3500
6	282	3500
7	283	3500
8	284	3000
9	285	3000
10	286	2500
11	287	2500
12	288	2500
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
	Total Weight:	**46000**
	Aircraft RLD	**11**
	Aircraft DD	**15**
	Aircraft AD	**16**

Table 4.4 Partial Solution Using LAD: Assignment of Pallets 264-288

The aircraft information after the pallet assignment is displayed in Table 4.5. Note that the second trips of aircraft 10004 and 10005 have been changed and that a third trip for both aircraft has been added.

Aircraft Information

ID Number	Trip Number	Ready to Load Date	Departure Date	Arrival Date	Total Number Pallets	Total Cargo Weight
10001	1	1	2	3	18	78500
10002	1	1	2	3	5	12500
10003	1	1	6	7	18	61750
10004	1	1	8	9	18	62200
10005	1	1	7	8	18	90000
10006	1	1	0	0	0	0
10007	1	1	0	0	0	0
10008	1	1	0	0	0	0
10009	1	1	0	0	0	0
10010	1	1	0	0	0	0
10001	2	6	6	7	12	89800
10002	2	6	6	7	18	78100
10001	3	10	13	14	18	78500
10002	3	10	13	14	18	69300
10003	2	10	14	15	18	62000
10004	**2**	**12**	**14**	**15**	**12**	**89400**
10005	**2**	**11**	**15**	**16**	**12**	**46000**
10001	4	17	0	0	0	0
10002	4	17	0	0	0	0
10003	3	18	0	0	0	0
10004	**3**	**18**	**0**	**0**	**0**	**0**
10005	**3**	**19**	**0**	**0**	**0**	**0**

Table 4.5 Aircraft Information: After Pallet Assignment Using LAD

4.2.4.2 *Example: Initial Solution Generator using EAD*

In the previous example, the first assignment of a pallet to an aircraft trip caused the trip's AD to be set to the pallet's LAD. Instead, in this example, the trip's AD is set to the pallet's EAD and the possible implications are examined. Assume that the same initial information that was presented in Table 4.2 and Table 4.3 applies to this example.

The initial solution generator begins by selecting the proper aircraft for assignment of pallet 264. The selection process is the same that was described in the previous example; pallet 264 is assigned to aircraft 10004, trip 2. The trip AD is set to the pallet's EAD of 13, the trip DD is computed and set to 12, and the total number of pallets and total cargo weight are adjusted to reflect the pallet assignment. Because this is the first pallet assignment to this aircraft, the next trip (with an RLD of 18) is inserted into the solution. Since it has an acceptable AD and sufficient space and weight remaining, pallets 264-270 are assigned to the same aircraft trip. At this point, there are seven pallets assigned to the aircraft weighing a total of 37,000 pounds.

No non-empty aircraft trip is available with an acceptable arrival date for pallet 271, so the pallet is assigned to an empty aircraft trip: aircraft 10005, trip 2. The trip AD is set to the pallet's EAD of 14, the trip DD is computed and set to 12, and the total number of pallets and total cargo weight are adjusted to reflect the pallet assignment. Since this is the first pallet assignment to this aircraft, the next trip (with an RLD of 19) is inserted into the solution. Pallets 272-284 are assigned to the same aircraft trip, resulting in a total of 14 pallets weighing 87,900 pounds.

There is not sufficient weight remaining on aircraft 10005, trip 2, to support pallet 285, and no other non-empty trip has an acceptable AD. As a result, an empty trip must be selected. No aircraft with multiple trips have an RLD which will allow arrival by the pallet EAD. This situation requires using an aircraft which has not yet been assigned pallets. Note that it does not simply require another trip by a previously used aircraft, but rather an additional *aircraft* must be utilized. Aircraft 10006 is the first unused aircraft, so it is selected and pallet 285 assigned to it. The aircraft AD is set to the pallet EAD of 14, the trip DD is computed and set to 12, and the total number of pallets and total cargo weight are adjusted to reflect the pallet assignment. Since this is the first pallet assignment to this aircraft, the next trip of aircraft 10006 (with an RLD of 19) is inserted into the solution.

In a similar manner, the remaining three pallets (286-287) are assigned to aircraft 10006, trip 1. The aircraft trip is now assigned 4 pallets and 10,500 pounds.

Pallet 289 has an arrival window of 19-22. None of the non-empty aircraft trips have an arrival date within this window, so an empty aircraft trip (aircraft 10001, trip 4) is selected. This process continues until the Big Bin is completely empty.

The assignment of pallets of 264-288 to specific aircraft is shown in Table 4.6. The aircraft information after the pallet assignment is displayed in Table 4.7. Note that the first trip of aircraft 10006 and the second trips of aircraft 10004 and 10005 have been changed. Additionally, a third trip has been added for aircraft 10004 and 10005 as well as a second trip for aircraft 10006.

Aircraft 10004, Trip 2

Position Num	Pallet ID Num	Weight
1	264	10000
2	265	7000
3	266	5500
4	267	5000
5	268	4500
6	269	2500
7	270	2500
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
	Total Weight:	**37000**
	Aircraft RLD	**12**
	Aircraft DD	**12**
	Aircraft AD	**13**

Aircraft 10005, Trip 2

Position Num	Pallet ID Num	Weight
1	271	10000
2	272	10000
3	273	9000
4	274	9000
5	275	8500
6	277	5900
7	276	8500
8	278	4500
9	279	4500
10	280	4500
11	281	3500
12	282	3500
13	283	3500
14	284	3000
15	0	0
16	0	0
17	0	0
18	0	0
	Total Weight:	**87900**
	Aircraft RLD	**11**
	Aircraft DD	**13**
	Aircraft AD	**14**

Aircraft 10006, Trip 1

Position Num	Pallet ID Num	Weight
1	285	3000
2	286	2500
3	287	2500
4	288	2500
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
	Total Weight:	**10500**
	Aircraft RLD	**1**
	Aircraft DD	**13**
	Aircraft AD	**14**

Table 4.6 Partial Solution Using EAD: Assignment of Pallets 264-288

Aircraft Information

ID Number	Trip Number	Ready to Load Date	Departure Date	Arrival Date	Total Number Pallets	Total Cargo Weight
10001	1	1	2	3	18	78500
10002	1	1	2	3	5	12500
10003	1	1	6	7	18	61750
10004	1	1	8	9	18	62200
10005	1	1	7	8	18	90000
10006	**1**	**1**	**13**	**14**	**4**	**10500**
10007	1	1	0	0	0	0
10008	1	1	0	0	0	0
10009	1	1	0	0	0	0
10010	1	1	0	0	0	0
10001	2	6	6	7	12	89800
10002	2	6	6	7	18	78100
10001	3	10	13	14	18	78500
10002	3	10	13	14	18	69300
10003	2	10	14	15	18	62000
10004	**2**	**12**	**12**	**13**	**7**	**37000**
10005	**2**	**11**	**13**	**14**	**14**	**46000**
10001	4	17	0	0	0	0
10002	4	17	0	0	0	0
10003	3	18	0	0	0	0
10004	**3**	**17**	**0**	**0**	**0**	**0**
10005	**3**	**18**	**0**	**0**	**0**	**0**
10006	**2**	**18**	**0**	**0**	**0**	**0**

Table 4.7 Aircraft Information: After Pallet Assignment Using EAD

4.2.4.3 *Initial Solution Generator Summary*

While feasibility (i.e. no weight or temporal violations) is not an initial solution requirement for TS, the DALP-TS initial solution generator produces a feasible solution for a given set of pallets and aircraft. The proper assignment of a pallet to an aircraft trip usually yields good quality (i.e. fewer aircraft and aircraft trips required) initial solutions. Selecting the best AD (and correspondingly the DD) for an aircraft trip which is receiving its first pallet assignment is imperative in producing good initial solutions. Assigning the aircraft trip AD to be the LAD of the first pallet assigned assists the initial solution generator in attempting to avoid scenarios similar that of the second example.

4.2.5 Objective Function

The DALP-TS objective function to be minimized is a combination of the penalties and usage fees. The components of the objective function are: (1) Percent Weight Full Penalty, (2) Lateral CB Penalty, (3) Longitudinal CB Penalty, (4) Aircraft Usage Fee, and (5) Temporal Violation Penalty. The first three components are identical to those described in Section 3.2.6: Equations 3.5 and 3.6 for Percent Weight Full Penalty, Equation 3.8 for Lateral CB Penalty and Equation 3.9 for Longitudinal CB Penalty. The fourth and fifth components are described below.

4.2.5.1 *Aircraft Usage Fee*

Similar to SALP-TS, a usage fee is charged for each aircraft trip employed. The usage fee varies with the aircraft type. In contrast to SALP-TS, DALP-TS charges an *increased* usage fee for the first aircraft trip which makes it more preferable for DALP-TS to use an additional trip on an aircraft than use an aircraft which has not yet been assigned any pallets. In the DALP, the goal is to transport all pallets with both the fewest trips and the fewest aircraft employed.

The aircraft usage fee is given by:

$$\begin{aligned}&\sum_{j=1}^{M} C_j^1 A_j V_j + \sum_{j=1}^{M} C_j^2 A_j \left(1 - V_j\right)\\&A_j \in \{0,1\} \forall\, j = 1,\ldots,M\\&V_j \in \{0,1\} \forall\, j = 1,\ldots,M\end{aligned}$$

Equation 4.1: Aircraft Usage Fee

where C_j^1 = the usage fee (cost) associated with aircraft j if this is the first use of aircraft j; C_j^2 = the normal usage fee (cost) associated with aircraft j; V_j = 1 if this is the first use of aircraft j and 0 otherwise; A_j = 1 if aircraft j is used and 0 otherwise; M = the number of aircraft available.

4.2.5.2 *Temporal Violation Penalty*

Each individual pallet has an acceptable window of arrival dates and a strict lower bound on its DD. It is preferable for a pallet to arrive after its EAD and before its LAD; no specific date within the acceptable window is preferable to any other. DALP-TS allows exploration of solutions with arrival dates outside the acceptable region by applying penalties to these violations. Lambert and Barnes' (2006) and McKinzie and Barnes' (2006) approach for temporal violations involved penalizing the OF by a product of the number of days the item arrives after the RDD and the item's weight. DALP-TS utilizes a similar approach where both early and late arrivals are penalized; however, the more restrictive LAD (rather than the RDD) is used. Pallets arriving prior to their EAD incur a penalty equal to the difference between the EAD and arrival date multiplied by the pallet weight and a penalty factor. Likewise, pallets arriving after their LAD incur a penalty equal to the difference between the arrival date and LAD multiplied by the pallet weight and a penalty factor.

The temporal violation penalty is given by:

$$\lambda_7 \sum_{j=1}^{M}\sum_{i=1}^{N}(EAD - arrival\,date)A_j Z_{ij}^1 W_i + \lambda_8 \sum_{j=1}^{M}\sum_{i=1}^{N}(arrival\,date - LAD)A_j Z_{ij}^2 W_i$$
$$A_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$Z_{ij}^1 \in \{0,1\} \forall i = 1,\ldots,N; j = 1,\ldots,M$$
$$Z_{ij}^2 \in \{0,1\} \forall i = 1,\ldots,N; j = 1,\ldots,M$$

Equation 4.2 Temporal Violation Penalty

where A_j is 1 if aircraft j is used, 0 otherwise; Z_{ij}^1 = 1 if aircraft j's arrival date is before pallet *i's* EAD; Z_{ij}^2 = 1 if aircraft j's arrival date is after pallet *i's* LAD; W_i is the weight of pallet i; λ_7 = the penalty factor associated with the pallet arriving prior to the EAD; and λ_8 = the penalty factor associated with the pallet arriving after the LAD.

4.2.5.3 *Objective Function Summary*

The DALP-TS objective function additively combines Equations 3.5, 3.6, 3.9, 3.9, 4.1, and 4.2 and is given by:

$$\sum_{j=1}^{M} C_j^1 A_j V_j + \sum_{j=1}^{M} C_j^2 A_j \left(1 - V_j\right) + \lambda_2 \sum_{j=1}^{M} \left(100 - \%WF_j\right)^2 A_j X_j$$
$$+ \lambda_2 \lambda_3 \sum_{j=1}^{M} \left(100 - \%WF_j\right)^2 A_j \left(1 - X_j\right) + \lambda_4 \sum_{j=1}^{M} \left(Lat\_CB_j\right)^2 A_j$$
$$+ \lambda_5 \sum_{j=1}^{M} \left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_j Y_j$$
$$+ \lambda_5 \lambda_6 \sum_{j=1}^{M} \left(Target\_Long\_CB_j - Long\_CB_j\right)^2 A_j \left(1 - Y_j\right)$$
$$+ \lambda_7 \sum_{j=1}^{M} \sum_{i=1}^{N} \left(EAD - arrival\, date\right) A_j Z_{ij}^1 W_i$$
$$+ \lambda_8 \sum_{j=1}^{M} \sum_{i=1}^{N} \left(arrival\, date - LAD\right) A_j Z_{ij}^2 W_i$$
$$A_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$V_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$X_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$Y_j \in \{0,1\} \forall j = 1,\ldots,M$$
$$Z_j^1 \in \{0,1\} \forall j = 1,\ldots,M$$
$$Z_j^2 \in \{0,1\} \forall j = 1,\ldots,M$$

Equation 4.3: DALP-TS Objective Function

Minimizing the objective function encourages DALP-TS to search for solutions which use a minimal number of aircraft, have pallets arriving within their acceptable arrival window, and have pallets ideally positioned within the aircraft.

4.2.6 Move Neighborhoods

Three of the four move neighborhoods used by SALP-TS are also incorporated into DALP-TS. Since the initial solution will always have all pallets loaded onto aircraft, the *Big Bin to Aircraft Insert* neighborhood is not necessary in DALP-TS. The three neighborhoods are: (1) Unload Entire Aircraft, (2) Intra-

Aircraft Insert/Swap, and (3) Inter-Aircraft Insert/Swap. Each move is called under specific strategic circumstances.

4.2.6.1 *Unload Entire Aircraft Neighborhood*

The unload portion of this neighborhood does not change from SALP-TS to DALP-TS. In both cases, the selected aircraft is emptied of pallets in order of decreasing weight. Unlike SALP-TS, the pallets are inserted into non-empty aircraft with at least one empty pallet position *and* a suitable AD. If multiple aircraft have acceptable ADs, the pallet is inserted into the aircraft with the greatest weight capacity remaining. If no aircraft has an acceptable window, the pallet is inserted into the aircraft which causes the smallest amount of pallet temporal violation penalty.

4.2.6.2 *Intra-Aircraft Insert/Swap Neighborhood*

No changes from the SALP-TS intra-aircraft insert/swap neighborhood are necessary to incorporate this neighborhood into DALP-TS. Every possible two-tuple (swapping two pallets or inserting a pallet into an empty position) *within* a single aircraft is evaluated. Swaps of pallets with identical weights and temporal constraints are disallowed because of the null effect on the aircraft load. The best non-tabu move is selected and performed. The lowest indexed pallet moved may not return to its previous position for tabu tenure future iterations.

4.2.6.3 *Inter-Aircraft Insert/Swap Neighborhood*

The DALP-TS Inter-Aircraft Insert/Swap neighborhood is an extension of the SALP-TS neighborhood; two non-empty aircraft are considered. Every possible two-tuple (swapping two pallets or inserting a pallet into an empty position) *between* two aircraft is evaluated. Swaps of pallets with identical weights and temporal constraints are disallowed. The best non-tabu move is selected and performed. Returning a pallet to its donor aircraft is not allowed for tabu tenure future iterations.

Several changes to an aircraft's information can occur during a DALP-TS inter-aircraft insert/swap. The same changes from the SALP-TS inter-aircraft insert/swap neighborhood can occur: (1) weight, (2) longitudinal CB, and (3) lateral CB. In DALP-TS, the aircraft trip DD may also change to minimize pallet temporal violations.

If the current DD of a trip is later than the RLD, the DD may be decreased. An aircraft DD prior to its RLD is not permitted. Decreasing a DD will only occur if the change is beneficial to the overall OF value (i.e. the change lowers the OF value). Changing the DD to a date which causes infeasibilities in the majority of the loaded pallets is unproductive. Decreasing the DD of an aircraft trip will cause a series of RLD decreases to occur in subsequent aircraft trips. The DD of the succeeding aircraft trips will be changed if it is beneficial to the overall OF value. Note that the trip RLD is the earliest date upon which an aircraft *may* depart; it is *not necessarily* the best DD for a given pallet load.

If a trip's DD is changed to a later date, a series of updates to the subsequent aircraft trips may be necessary. This occurs when a trip's DD is altered causing the RLD of the succeeding trip to be *after* its DD. This is not a feasible state and the DD will be adjusted accordingly.

4.2.7 Local Search Procedure

The three neighborhoods described in Section 4.2.6 are strategically employed during the search. DALP-TS incorporates a dynamic neighborhood selection process to select the most appropriate neighborhood at each point in the search.

The first step in the DALP-TS search is identical to that of SALP-TS: create the tabu memory structure, as described in Section 3.2.3, and create and initialize the variables required in the search. Next, DALP-TS attempts to adjust all CBs for the aircraft.

2.1.1.2 CB Adjustment

Since the initial solution generator ignores CB, DALP-TS performs a series of Intra-Aircraft Insert/Swap moves using the same methodology as that described in Section 2.1.1.1 for the SALP-TS.

4.2.7.1 *Stopping Criteria for the Dynamic Neighborhood Selection Process*

After the longitudinal CB adjustments are performed, DALP-TS initiates the dynamic neighborhood selection portion of the search. For purposes of clarity, we consider the stopping criteria first. The dynamic neighborhood selection is repeated until there have been (1) 20 adjacent disimproving moves; (2) 20 adjacent trivially improving moves, or (3) 500 total iterations. The improving, trivially improving, and disimproving moves are initialized and updated in an identical method to that of the SALP-TS described in Section 3.2.8.2.

4.2.7.2 *Dynamic Neighborhood Selection Process*

Three states arise from the move neighborhoods previously described. These states are:

1. (Aircraft CB Allowable Window Violated) $\cap$ (Fix CB Count $\leq$ 5)
2. (15 Adjacent Trivially Improving Moves) $\cup$ (15 Adjacent Disimproving Moves)
3. Neither case 1 or 2 applies.

State 1

An aircraft longitudinal CB window violation often can be corrected, i.e., "fixed" with the *Intra-Aircraft Insert/Swap* neighborhood. There are three ways to escape state 1: (1) the CB window is satisfied, (2), the CB window is unsatisfied for 5 iterations (timely progress is not present), or (3) the CB window violation worsens. These moves result in relatively small changes to the OF value by changing the CB of a *single aircraft*.

State 2

Unlike the SALP, only two situations render state 2 in the DALP. This state invokes the *unload entire aircraft* neighborhood:

(1) If 15 adjacent iterations result in trivially improving moves *and* the number of aircraft in use is not less than the lower bound, the algorithm has stabilized on an unproductive plateau. Unloading an entire aircraft constitutes a diversification strategy which can lead to better solutions.

(2) If 15 disimproving moves have been performed *and* the number of aircraft in use is not less than the lower bound, diversification is indicated to escape from a nonproductive region of the solution space.

Because there are temporal as well as weight restrictions imposed upon aircraft, DALP-TS does not consider unloading trivially loaded aircraft. The temporal constraints may necessitate a loading which could be considered trivial. These moves result in large changes to the OF value by removing an aircraft from consideration.

State 3

If none of the previous states apply, the *Inter-Aircraft Insert/Swap* neighborhood will be used. These moves result in mid-sized objective function value changes and are frequently used throughout DALP-TS.

4.2.8 Solution Output

Upon completion of the algorithm, up to twelve solutions are reported. DALP-TS is a *decision making aid* and, as such, should allow the decision maker to select a preferred solution for a given situation by utilizing his/her experience and intuition and information not necessarily embodied in the DALP-TS model. The solutions are categorized by four group types: (1) Feasible, (2) ACL violations, (3) Temporal Violations, and (4) ACL and Temporal Violations.

If DALP-TS encounters solutions from a specific group type, then the three best solutions from the type are presented, otherwise, no solution of the specific group type are presented. The first solution type satisfies all constraints—there are no ACL violations and all pallets have an arrival date that is after the EAD but before the LAD. The next three solution types are infeasible, but the benefits of these violations may enable the decision makers to choose them over a strictly feasible solution. The second solution type allows the total loaded pallet weight of one or more aircraft to exceed the aircraft ACL by no more than 2.5%. The third solution type includes those solutions which have violations in at least one pallet arrival date (either early or late). The last solution type encompasses solutions which have no more than 2.5% ACL violations *and* have violations in at least one pallet arrival date.

If solutions of these types are encountered during the search, the best three of each type will be presented as output. It is possible during the search process that SALP-TS may not encounter one or more type of solutions. For example, the pallet weights in a data set may be such that it is not possible to have ACL violations less than 2.5% with no temporal violations. In this case, no solutions with only ACL violations will be presented as output.

4.3 COMPUTATIONAL RESULTS

DALP-TS was tested using six different scenarios. At the time of this research, no previously developed baseline method or solution methodology to the DALP could be located. The results of the DALP-TS are compared with the first solution, or baseline solution, used by the dynamic search procedure portion of DALP-TS. The baseline solution is not the original solution produced by the initial solution generator. Instead, it is the initial solution *after* it has received CB adjustment. Unlike the AALPS solution used for comparison in the SALP, the CB for this solution is adjusted even after the aircraft achieves feasible CB location. It is allowed to perform up to 5 trivially improving Intra-Aircraft Swap/Insert moves to further enhance the CB location. The scenario descriptions and the comparative results are presented in the following sections.

4.3.1 Scenario Description

Two different factors are varied to develop the test scenarios: Number of Pallets and Type of Aircraft. Two sets of pallets were used in the scenarios: one with 500 pallets and another with 1000 pallets. Pallets are assumed to have a lower and upper bound on weight of 2500 and 10,000 pounds, respectively. At the time of this research, no data sets were available at the DALP level. As a result, two pallet sets were generated from information contained in an unclassified version of the January 2005 TPFDD from the Air Expeditionary Force cycle 9/10. The 500 pallet data set has a planning timeline of day 1 – day 29; the earliest ALD is day 1 and the latest RDD is day 29. The 1000 pallet data set has a planning timeline of day 1 – day 51. The 1000 pallet data set is a continuation of the 500 pallet data set, which implies that the 500 pallet data set is contained in the 1000 pallet data set.

Since items in a TPFDD are not listed as they would appear after palletization, only the weights and temporal constraints were required. To generate the pallet data from the weight listed in a TPFDD, first upper and lower bounds on the pallet number were computed. The upper bound is the largest number of pallets that could be created from the given weight. For example, if the TPFDD specified 31.9 short tons (63,800 pounds) of cargo, the maximum number of pallets possible is given by: $\lfloor 63800/2500 \rfloor = 25$, and the minimum number is given by: $\lceil 63800/10000 \rceil = 7$. A random number between these bounds (inclusive) was generated. To generate the weights for this number of pallets, Microsoft Excel Solver was used to create weights between 2500 and

10,000 pounds which sum to the specified weight from the TPFDD using the randomly generated number of pallets. To simplify the process, weights were computed in hundred pound increments.

Two types of aircraft (the C-17 and C-5) were used in the scenarios. Although DALP-TS can accommodate nine different configurations of six different military airlift aircraft, only the principle military strategic airlifters were incorporated into the scenarios generated here. The ACL for each of these aircraft were set to be the planning ACL value. The six scenarios are shown in Table 3.5

Number of Pallets	Type of Aircraft	Label Name
500	C-17	Scenario 1
1000	C-17	Scenario 2
500	C-5	Scenario 3
1000	C-5	Scenario 4
500	C-17 & C-5	Scenario 5
1000	C-17 & C-5	Scenario 6

Table 4.8 Scenarios for DALP-TS Testing

4.3.2 Results

The *best* solution from each solution type is presented and compared to the baseline solution. Each scenario result section contains a fifteen row chart, with row descriptions as follows:

1. Solution Type—Feasible, ACL Violations, Temporal Violations, Temporal & ACL Violations
2. # Aircraft in Baseline—Number of planes required in the baseline solution.
3. # Trips in Baseline—Number of trips required in the baseline solution.
4. Baseline OF Value—The OF value of the baseline solution.

5. # Aircraft in DALP-TS—Number of planes required in the DALP-TS solution.
6. # Trips in DALP-TS—Number of trips required in the DALP-TS solution.
7. DALP-TS OF Value—The OF value of the DALP-TS solution.
8. # Aircraft Reduction—Reduction in the number of planes for the DALP-TS solution when compared to the baseline
9. % Aircraft Reduction—The percent reduction in the number of aircraft required per solution when the DALP-TS solution is compared to the baseline solution. It is given by:

$$\text{\% Aircaft Reduction} = \frac{\text{\# Aircraft Reduction}}{\text{\# Aircraft Required in Baseline}}$$

Equation 4.4: DALP-TS % Aircraft Reduction

10. # Trip Reduction—Reduction in the number of trips for the DALP-TS solution when compared to the baseline
11. % Trip Reduction—The percent reduction in the number of trips required per solution when the DALP-TS solution is compared to the baseline solution. It is given by:

$$\text{\% Trip Reduction} = \frac{\text{\# Trip Reduction}}{\text{\# Trips Required in Baseline}}$$

Equation 4.5: DALP-TS % Trip Reduction

12. $ Saved by Reduction = This is the amount of money saved by reducing the required number of trips and is given by:

$$\sum_{j \in M} A_j C_j$$

Equation 4.6: DALP-TS $ Saved by Reduction

13. # Trips Infeasible—Number of planes exceeding ACL, violating temporal constraints, or both.
14. Amount Temporal Violation—The sum of the short ton days for pallet temporal violations.
15. % OF Reduction—The percentage of the reduction in OF value when the DALP-TS is compared to the baseline solution. It is given by:

$$\%\text{ OF Reduction} = \frac{(\text{Baseline OFValue - }[\text{DALP - TS OF Value}])}{\text{Baseline OFValue}}$$

Equation 4.7: DALP-TS % OF Reduction

16. CPU Time (sec)—The time (in seconds) required for SALP-TS to find this solution.

Row 11 contains the amount (in dollars) saved by using the DALP-TS solution over the baseline solution. This row is computed identically to the method used in SALP-TS described in Section 3.3.2: the cost of each round-trip overseas mission is $523,350 for the C-17 and $873,180 for the C-5.

Row 14 contains the amount of temporal violations for all pallets. The temporal violation for a single pallet is the product of the pallet weight (in short tons) and the number of days which the pallet is in temporal violation. This in commonly referred to as ton days early/late.

The penalty values used in the objective function in each of the scenarios are listed in Table 4.9. These values are based upon knowledge of the problem and experimentation.

Penalty Description:	**Value:**
Big Bin =	10,000
Aircraft % Weight Full =	1
Over Weight =	30
Longitudinal CB Location =	0.25
Latitudinal CB Location =	0.25
CB Out of Window =	200
C-17 Aircraft Usage =	5000
C-5 Aircraft Usage =	5000
Temporal Violations =	0.25

Table 4.9 Penalty Multiplier Values for DALP Scenarios

4.3.2.2 *Scenario 1*

Scenario 1 has 500 pallets to be assigned to C-17 aircraft. The results for this scenario are presented in Table 4.10.

Solution Type	Feasible	Temporal Violations	Temporal & ACL Violations
# Aircraft in Baseline	10	10	10
# Trips in Baseline	33	33	33
Baseline OF Value	675247.97	675247.97	675247.97
# Aircraft in DALP-TS	10	10	10
# Flights in DALP-TS	32	29	29
DALP-TS OF Value	639325.35	531631.04	532126.49
# Aircraft Reduction	0	0	0
% Aircraft Reduction	0.00%	0.00%	0.00%
# Flight Reduction	1	4	4
% Flight Reduction	3.03%	12.12%	12.12%
$ Saved by Reduction	$523,350	$2,093,400	$2,093,400
# Trips with ACL Violations	0	0	1
# Trips with Temporal Violations	0	4	3
Amount Temporal Violation	0	32.3	27.9
% OF Reduction	5.32%	21.27%	21.20%
CPU Time (sec)	86.125	347.562	350.531

Table 4.10 DALP-TS Scenario 1 Results

The baseline solution required 10 aircraft and 33 trips. The best feasible solution required 10 aircraft and 32 trips. No solutions were encountered during DALP-TS with only ACL violations. The best solutions with temporal only and temporal and ACL violations both required 10 aircraft and 29 trips. The cost savings from this reduction would be approximately $2 million.

4.3.2.3 *Scenario 2*

Scenario 2 has 1000 pallets to be assigned to C-17 aircraft. The results for this scenario are presented in Table 4.11.

Solution Type	Feasible	Temporal Violations
# Aircraft in Baseline	12	12
# Trips in Baseline	64	64
Baseline OF Value	961696.20	961696.20
# Aircraft in DALP-TS	12	12
# Flights in DALP-TS	63	62
DALP-TS OF Value	919283.12	856261.80
# Aircraft Reduction	**0**	**0**
% Aircraft Reduction	**0.00%**	**0.00%**
# Flight Reduction	**1**	**2**
% Flight Reduction	**1.56%**	**3.13%**
$ Saved by Reduction	**$523,350**	**$1,046,700**
# Trips with ACL Violations	**0**	**0**
# Trips with Temporal Violations	**0**	**1**
Amount Temporal Violation	**0**	**12.5**
% OF Reduction	**4.41%**	**10.96%**
CPU Time (sec)	364.657	774.407

Table 4.11 DALP-TS Scenario 2 Results

The baseline solution required 12 aircraft and 64 trips. The best feasible solution required 12 aircraft and 63 trips. No solutions were encountered during DALP-TS with only ACL violations or with temporal and ACL violations. The best solutions with temporal only violations required 12 aircraft and 622 trips. The cost savings from this reduction would be approximately $1 million.

4.3.2.4 *Scenario 3*

Scenario 3 has 500 pallets to be assigned to C-5 aircraft. The results for this scenario are presented in Table 4.12.

Solution Type	Feasible	ACL Violations	Temporal Violations	Temporal & ACL Violations
# Aircraft in Baseline	7	7	7	7
# Trips in Baseline	20	20	20	20
Baseline OF Value	589479.10	589479.10	589479.10	589479.10
# Aircraft in DALP-TS	7	7	7	7
# Flights in DALP-TS	20	18	18	18
DALP-TS OF Value	430221.25	365640.35	380822.40	367769.40
# Aircraft Reduction	0	0	0	0
% Aircraft Reduction	0.00%	0.00%	0.00%	0.00%
# Flight Reduction	0	2	2	2
% Flight Reduction	0.00%	10.00%	10.00%	10.00%
$ Saved by Reduction	$0	$1,746,360	$1,746,360	$1,746,360
# Trips with ACL Violations	0	1	0	1
# Trips with Temporal Violations	0	0	1	1
Amount Temporal Violation	0	0	35.0	7.5
% OF Reduction	27.02%	37.97%	35.40%	37.61%
CPU Time (sec)	237.140	445.250	498.125	494.296

Table 4.12   DALP-TS Scenario 3 Results

The baseline and the best feasible solutions required 7 aircraft and 20 trips. The best solutions with ACL only, temporal only, and temporal and ACL violations required 7 aircraft and 18 trips. Unless overloading is strictly forbidden, the decision maker would prefer to use the solution with only ACL violations; no pallets arrive late in this solution. The cost savings from this reduction would be approximately $1.75 million.

4.3.2.5 *Scenario 4*

The results for scenario 4 are presented in Table 4.13.

Solution Type	Feasible	ACL Violations	Temporal Violations	Temporal & ACL Violations
# Aircraft in Baseline	7	7	7	7
# Trips in Baseline	38	38	38	38
Baseline OF Value	683863.09	683863.09	683863.09	683863.09
# Aircraft in DALP-TS	7	7	7	7
# Flights in DALP-TS	38	37	37	37
DALP-TS OF Value	520221.60	465760.37	481210.04	467820.72
# Aircraft Reduction	0	0	0	0
% Aircraft Reduction	0.00%	0.00%	0.00%	0.00%
# Flight Reduction	0	1	1	1
% Flight Reduction	0.00%	2.63%	2.63%	2.63%
$ Saved by Reduction	$0	$873,180	$873,180	$873,180
# Trips with ACL Violations	0	1	0	1
# Trips with Temporal Violations	0	0	1	1
Amount Temporal Violation	0	0	35.0	7.5
% OF Reduction	23.93%	31.89%	29.63%	31.59%
CPU Time (sec)	945.313	1201.891	1198.360	1187.079

Table 4.13 DALP-TS Scenario 4 Results

The baseline and the best feasible solutions required 7 aircraft and 38 trips. The best solutions with ACL only, temporal only, and temporal and ACL violations both required 7 aircraft and 37 trips. Unless overloading is strictly forbidden, the decision maker would prefer to use the solution with only ACL violations; no pallets arrive late in this solution. The cost savings from this reduction would be approximately $900,000.

4.3.2.6 *Scenario 5*

The results for scenario 5 are presented in Table 4.14.

Solution Type	Feasible	Temporal Violations
# Aircraft in Baseline	9	9
# Trips in Baseline	26	26
Baseline OF Value	974735.78	974735.78
# Aircraft in DALP-TS	9	9
# Flights in DALP-TS	25	23
DALP-TS OF Value	504179.81	436096.74
# Aircraft Reduction	0	0
% Aircraft Reduction	0.00%	0.00%
# Flight Reduction	1	3
% Flight Reduction	3.85%	11.54%
$ Saved by Reduction	$873,180	$2,619,540
# Trips with ACL Violations	0	0
# Trips with Temporal Violations	0	1
Amount Temporal Violation	0	6.3
% OF Reduction	48.28%	55.26%
CPU Time (sec)	182.734	488.031

Table 4.14 DALP-TS Scenario 5 Results

The baseline solution required 9 aircraft and 26 trips. The best feasible solution required 9 aircraft and 25 trips. No solutions were encountered during DALP-TS with only ACL violations or with temporal and ACL violations. The best solutions with temporal only violations required 9 aircraft and 23 trips. The cost savings from this reduction would be approximately $2.6 million.

4.3.2.7 *Scenario 6*

The results for scenario 6 are presented in Table 4.15.

Solution Type	Feasible	Temporal Violations
# Aircraft in Baseline	11	11
# Trips in Baseline	49	49
Baseline OF Value	1093922.05	1093922.05
# Aircraft in DALP-TS	11	11
# Flights in DALP-TS	48	46
DALP-TS OF Value	764373.23	693007.43
# Aircraft Reduction	0	0
% Aircraft Reduction	0.00%	0.00%
# Flight Reduction	1	3
% Flight Reduction	2.04%	6.12%
$ Saved by Reduction	$873,180	$2,619,540
# Trips with ACL Violations	0	0
# Trips with Temporal Violations	0	1
Amount Temporal Violation	0	6.3
% OF Reduction	30.13%	36.65%
CPU Time (sec)	736.313	2159.672

Table 4.15 DALP-TS Scenario 6 Results

The baseline solution required 11 aircraft and 49 trips. The best feasible solution required 11 aircraft and 48 trips. No solutions were encountered during DALP-TS with only ACL violations or with temporal and ACL violations. The best solutions with temporal only violations required 11 aircraft and 46 trips. The cost savings from this reduction would be approximately $2.6 million.

4.3.3 Results Summary

DALP-TS solutions were directly compared to an initial baseline solution in six different scenarios. These scenarios were first tested using SALP-TS; the temporal constraints were removed from the scenarios. SALP-TS demonstrated similar percentage reductions to those produced using DALP-TS.

With respect to required number of aircraft trips, DALP-TS improved upon the baseline solution. The reduction in required number of aircraft trips is presented in Figure 4.1. In four scenarios, DALP-TS produced feasible results which required fewer trips than the baseline solution. The feasible solutions reduced the required number of trips by 0 or 1 with an average of 0.67. Similar to SALP-TS, results were produced which contained ACL violations of no more than 2.5% over the aircraft ACL in two scenarios. In the remaining four scenarios, solutions with only ACL violations were not encountered. This is indicated by the asterisks (*) on Figure 4.1. The solutions with ACL only violations reduced the required number of trips by 1 or 2 with an average of 1.5.

Additionally, results were produced in which temporal violations (pallets arrive early or late) were allowed. In every scenario, solutions containing temporal violations were produced. In the solutions with only temporal violations, the required number of trips was reduced by 1 to 4 with an average of 2.5. The largest amount of short-ton days early/late in any of these solutions was 35.

In three scenarios, solutions were also produced which contained both ACL and temporal violations. In these solutions, the required number of trips was reduced by 1 to 4 with an average of 2.33. Of these scenarios, the largest violations experienced were 27.9 short ton days early/late and two aircraft over ACL. If it is not beneficial or feasible to reduce the number of trips required, DALP-TS will equally distribute (as much as possible) the loads among the aircraft with proper temporal constraints while simultaneously attempting to more closely achieved the target CB location on both longitudinal and lateral CB.

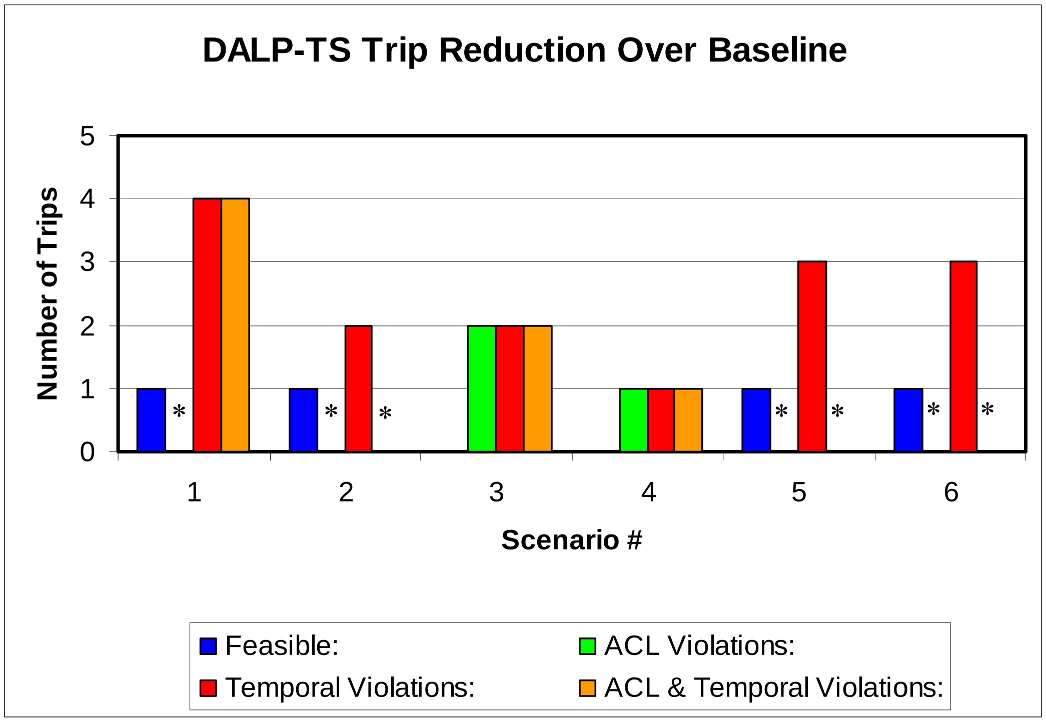

Figure 4.1 DALP-TS Trip Reduction over Baseline

Figure 4.2 presents the trip reduction as a percentage. Feasible solutions were produced which reduced the number of trips required by 0% to 3.85% with an average of 1.75%. The two scenarios which encountered ACL only violations reduced the number of trips by 2.63% and 10.00% with an average of 6.32%. Solutions with only temporal violations were produced which reduced the number of trips from 2.63% to 12.12% with an average of 7.59%. In the scenarios in which solutions with both ACL and temporal violations were encountered, DALP-TS produced results which reduced the required number of trips from 2.63% to 12.12% with an average of 8.25%.

The DALP involves those missions that are considered contingency: those missions that involve the initial deployment of military forces into an area. As a result, any reduction in the number of aircraft or trips required releases an aircraft or trip to be utilized to service APOE/APOD pairs. Cost savings which can be gained by a reduction in aircraft or trips required does not adequately account for the availability of an aircraft for other uses.

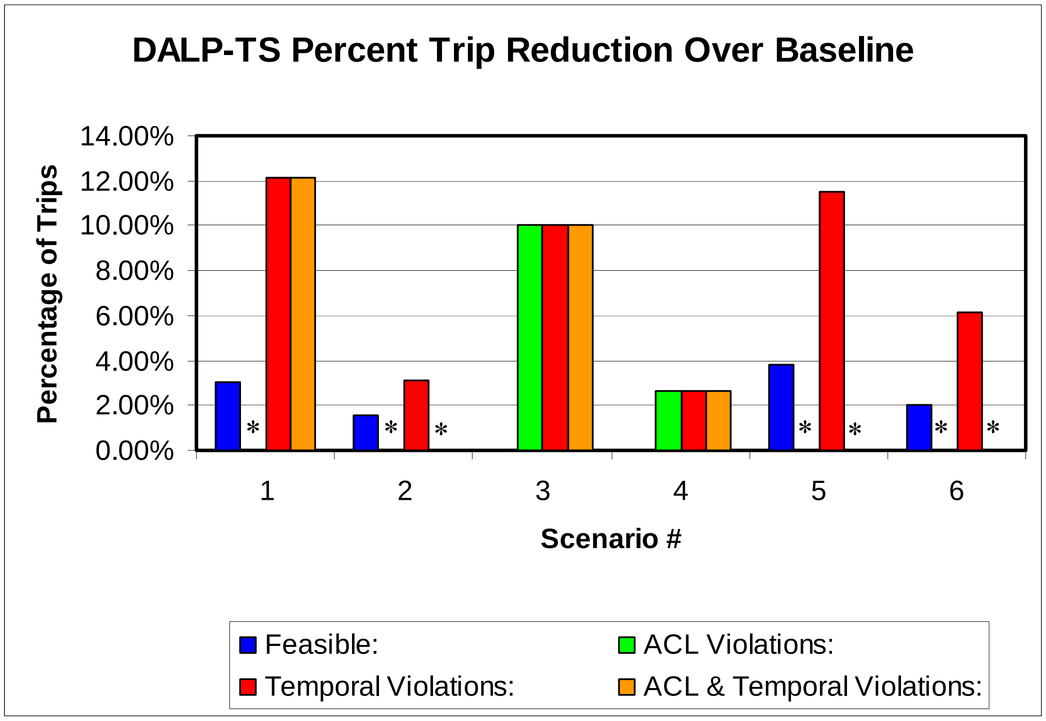

Figure 4.2 DALP-TS Percent Trip Reduction over Baseline

In terms of OF value, DALP-TS outperformed the baseline solution in every scenario. The percentage reduction in OF value is demonstrated in Figure

4.3. The reduction in the OF value ranges from 4.41% to 55.26%. DALP-TS produced feasible solutions which reduced the OF value from 4.41% to 48.28% with an average of 23.18%. In two scenarios, solutions were encountered with only ACL violations resulting in an OF reduction of 31.89% and 37.97% with an average of 34.93%. Solutions with only temporal violations were produced in every scenario with a range of 10.96% to 5.26% and an average of 31.53%. DALP-TS encountered solutions with both ACL and temporal violations in three scenarios. The trip reduction for these scenarios ranged from 12.20% to 37.61%; the average reduction was 30.13%.

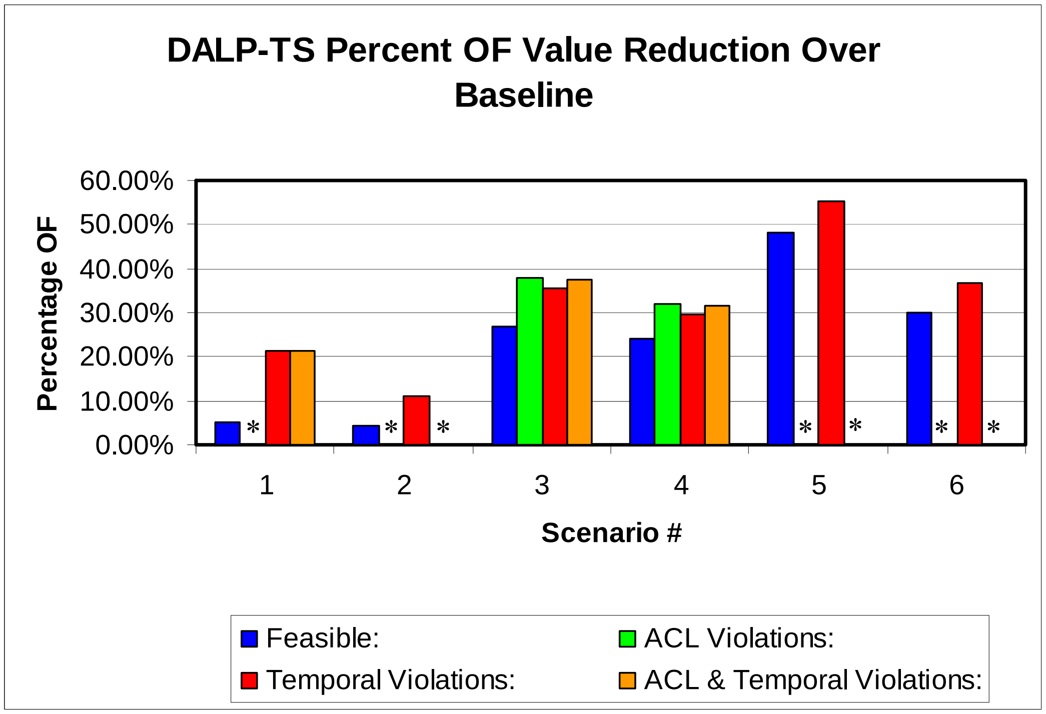

Figure 4.3 DALP-TS Percent OF Value Reduction over Baseline

The DALP-TS software provides complete detail on all aspects of each solution reported for the six scenarios described above. In the interest of brevity, detailed information on the temporal violations were not provided in the above discussion. In summary, however, the most extreme temporal identical violation occurred in scenarios 3 and 4 where one 10,000 pound pallet arrived seven days after its LAD. In several other scenarios, pallets arrived only one or two days after their LADs.

4.4 DALP SUMMARY

This chapter provided a detailed description of the DALP, the solution representation utilized in this research, and the DALP-TS algorithm developed as a proof of concept to solve the problem. The solution methodology used in DALP-TS produces effective results to an extremely difficult problem in a reasonable amount of time. In side-by-side testing, DALP-TS required fewer or the same number of aircraft as the baseline solution methodology.

Chapter 5: Conclusions

Chapters 3 and 4 presented the SALP-TS and DALP-TS. This chapter discusses the unique contributions of this research and presents possible enhancements or expansions to the research in the areas of aircraft loading and the overall end-to-end mobility problem.

5.1 MAJOR CONTRIBUTIONS

In addition to successfully defining and solving two previously unstudied problems, this research presented several improvements to Aircraft Load Planning and aircraft utilization which shows significant unique contributions in this area. These contributions are:

- the development of a practical, efficient, and effective solution methodology for the SALP
- the development of a unique dynamic neighborhood selection process for the SALP
- first formulation and modeling of the DALP
- the development of a practical, efficient, and effective solution methodology for the DALP
- the development of a unique dynamic neighborhood selection process for the DALP

The SALP-TS solution representation utilizes known aircraft configurations to assign pallets to specific locations within an aircraft. This representation lends itself to the TS methodology, specifically inserting and swapping of pallets and adding or removing aircraft. Testing pallet placements within multiple aircraft is exploited in the development of the SALP-TS neighborhoods.

The SALP-TS algorithm produces an array of excellent solutions to each instance of the ALP. The availability of multiple neighborhoods enables the dynamic neighborhood selection process to choose the neighborhood which is most applicable for use on a particular solution. An objective function was developed with penalty weights which can be modified, as desired, by the decision maker. This enables rapid modification of SALP-TS for a variety of different scenarios.

A unique dynamic neighborhood selection process was developed and implemented in SALP-TS. The dynamic neighborhood selection process incorporates problem specific knowledge as well as understanding of the TS search process. This process enables SALP-TS to effectively and efficiently search areas of quality solutions by ensuring the proper neighborhood is selected for a specific incumbent solution.

Since every constraint and condition for a particular SALP is not necessarily known in advance, if a solution type is encountered, three of each (for a possible total of nine) feasible, trivially infeasible, and marginally infeasible solutions are presented to the decision maker. From these, the decision maker can select their preferred solution. The levels of violations allowed in a trivially and marginally infeasible solution can also be user defined which empowers the decision maker with even more control over the process.

The efficiency and effectiveness of SALP-TS was presented using a set of realistic and academic scenarios. The SALP-TS results demonstrated an improvement in feasible solutions of, on average, 5% over the AALPS solution while requiring less than two minutes for problems involving up to 1000 pallets.

If AMC conducts an average of 2060 C-17 flights per year at an average cost of $523,350, and if 5% or 103 of these flights could be eliminated without loss of throughput, an estimated savings of *(103 * $523,350) = $53,905,050* per year could be obtained without loss of effectiveness. This estimate is only for *one aircraft type*; similar savings would be anticipated for the C-5. If one were to suppose the same number of C-5 flights were saved, an additional cost savings of *(103 * $873,180) = $89,937,540* would be realized. The combined total savings from these two airframes would be approximately *$53,905,050 + $89,937,540 = $143,842,590.* This estimate also does not consider the maintenance costs or wear and tear on the aircraft. Additionally, a 5% reduction allows fewer aircraft to be purchased. The potential reduction in cost due to requiring fewer aircraft purchases is also not incorporated into this cost savings estimate.

The above calculations are estimates of potential savings that could be realized if this methodology is utilized. The flight costs were derived from real-world estimates. An individual flight could require more (less) time and therefore cost more (less). Additionally, the number of flights was generated from real-world estimates, but the actual number of flights could be more (less) thereby increasing (decreasing) the potential savings.

Prior to this research, the DALP had not been formulated or modeled. More general models exist, but these models do not assign items to specific positions in an aircraft. Instead, these models assign items based upon weight to aircraft and assume that the aircraft has the spatial capacity to accommodate the items.

The DALP-TS algorithm expands upon the solution representation used in SALP-TS by including departure and arrival dates for aircraft as well as multiple instances of a specific aircraft. As with SALP-TS, this representation lends itself to the TS methodology, specifically inserting and swapping of pallets and adding and removing aircraft. The neighborhood definition in DALP-TS exploits the structure of the solution representation. The availability of multiple neighborhoods allowed the dynamic neighborhood selection process to select the neighborhood most applicable to a given solution.

As with SALP-TS, a unique dynamic neighborhood selection process was developed and implemented in DALP-TS. The dynamic neighborhood selection process incorporates problem specific knowledge as well as understanding of the TS search process. This process enables DALP-TS to effectively and efficiently search areas of quality solutions by ensuring proper neighborhood selection.

DALP-TS produces an array of solutions from which the decision maker can select the preferred solution. If encountered during the search process, three each of four different types of solutions (for a total of twelve) are presented at the conclusion of DALP-TS. The four different types of solutions can be described as feasible, those with weight only violations, those with temporal only violations, and those with both weight and temporal violations.

The efficacy of the DALP-TS algorithm was demonstrated on a series of problems with data sets created from information contained in a TPFDD. No previous baseline or comparison results were available for the DALP prior to this research. An initial solution generator similar to AALPS for the SALP was created to provide comparable baseline solutions for the DALP. This methodology could immediately be applied to a DALP and produce results analogous to an AALPS solution for the SALP. DALP-TS demonstrated the ability to produce multiple types of quality solutions in a timely manner.

5.2 FURTHER ENHANCEMENTS TO THE SALP-TS ALGORITHM

Expansion and/or enhancement to the SALP-TS algorithm will enable it to solve more general or more detailed ALP problems. One such enhancement is to incorporate non-palletized cargo and passengers into the loading process. For simplicity, non-palletized cargo could be modeled as joined or "married" pallets. For example, if a truck has the same footprint (required amount of aircraft floor space) as two pallets, it can be assigned to two adjacent pallets and the CG of the truck used to calculate the aircraft CB. Seating is available for passengers and the seat CG location is known. The aircraft CB contribution of passengers can easily be computed.

Adding non-palletized cargo and passengers greatly increases the number of constraints for the SALP-TS. For example, due to fire hazards, pallets cannot be loaded in front (toward the aircraft nose) of any wheeled or tracked vehicles.

Additionally, passengers cannot be seated adjacent to or behind any pallets. These additional constraints reduce the number of possible solutions, thereby reducing the search required for TS.

Multiple APOEs and APODs are not considered in the SALP-TS because these scenarios can be easily reduced to multiple problems with a single APOE and APOD. For example, if there are 3 APOEs and 4 APODs, there are a total of 12 (3*4) distinct different APOE/APOD pairs. In a SALP, aircraft cannot be reused, and hence cannot serve more than a single APOE/APOD pair. This allows all APOE/APOD pairs to be solved separately.

5.3 FURTHER ENHANCEMENTS TO THE DALP-TS ALGORITHM

The DALP-TS algorithm could also be expanded and/or enhanced to encompass a larger array of problems. As with the SALP-TS, the DALP-TS could be expanded to encompass non-palletized cargo items and passengers. The same methods previously described could be utilized.

Additionally, the DALP-TS could be expanded to handle multiple APOEs and APODs. In this scenario, an aircraft would not be required to return directly to its home base (or APOE) after every trip. Instead, the aircraft would be routed to the APOE to which it is most needed.

Nanry and Barnes (2000) is one of many instances where it has been demonstrated that adding constraints to a problem causes the new problem to be easier to solve than the original problem. Incorporating the above additional constraints in the DALP would likely create a problem where DALP-TS will perform even more efficiently because of the implicit reduction of the search space.

5.4 EXTENSIONS TO OTHER ASPECTS OF THE ALP

A possible extension to the ALP (both SALP and DALP) is to include pallet construction (i.e. three-dimensional packing). This requires individual tracking of each item (box, can, crate, etc.) to be transported by pallet. Additionally, the dimensions (size and weight) of each item must be known prior to solving the problem.

This extension would enable load planners to compute the exact pallet CG location rather than assuming it is at the center of the loaded pallet.

5.5 MERGING WITH OTHER RESEARCH

Combining the SALP-TS and DALP-TS algorithms with other research into a coherent framework could achieve greater impact and usability for the US military. ATS-SMMSP (McKinzie, 2005) could be utilized to select the mode of transportation, ATS-SAP (Lambert, 2004) could be used to determine the routing of the transported items, and DALP-TS could be used to determine the loading of the transported items. The fusion of these three methodologies would enable more efficient flow of items through the mobility network.

5.6 SUMMARY

The GAO report to the US Secretary of Defense argues that “inefficient use of aircraft capacity could cause higher operational tempo and may increase cost as well as wear and tear on aircraft” (2005). This research examined two large, complex problems which consider the utilization of AMC aircraft.

Effective and efficient methodologies for solving these two problems were presented.

SALP-TS incorporates an intelligent initial solution designed to simultaneously maximize both aircraft space and weight usage. A dynamic neighborhood selection process is incorporated in a TS approach to improve upon the solution producing excellent results. The results from SALP-TS were compared to the currently used methodology (AALPS), thereby demonstrating the efficacy of this approach.

DALP-TS utilizes a solution representation and methodology which are properly suited for a dynamic neighborhood selection methodology within a TS framework. This enables DALP-TS to produce quality results in a timely manner. The results of DALP-TS were compared to a baseline solution which is a DALP extension of the methodology used by AALPS to solve a SALP.

Appendix A – Sample SALP-TS Aircraft Input File

Aircraft ID Number	Aircraft Type	ACL
10001	1	90000
10002	3	150000
10003	1	90000
10004	3	150000
10005	1	90000
10006	3	150000
10007	1	90000
10008	3	150000
10009	1	90000
10010	3	150000
10011	1	90000
10012	3	150000
10013	1	90000
10014	3	150000
10015	1	90000
10016	3	150000
10017	1	90000
10018	3	150000
10019	1	90000
10020	3	150000

In this input file, 20 aircraft are available. As explained in Table 3.1, aircraft of type 1 are C-17s and type 3 are C-5s. In this input file, each C-17 has a 90,000 pound ACL while each C-5 has a 150,000 pound ACL.

Appendix B – Sample SALP-TS Pallet Input File

Pallet ID Num	Weight	Cargo Height	Cargo Width	Cargo Length
101	9385	76	84	104
102	9279	76	84	104
103	8921	76	84	104
104	8920	76	84	104
105	8661	76	84	104
106	8517	76	84	104
107	8380	76	84	104
108	8249	76	84	104
109	8230	76	84	104
110	8038	76	84	104
111	7820	76	84	104
112	7730	76	84	104
113	7074	76	84	104
114	7068	76	84	104
115	5501	76	84	104
116	5088	76	84	104
117	4664	76	84	104
118	4392	76	84	104
119	4125	76	84	104
120	3634	76	84	104
121	3474	76	84	104
122	2935	76	84	104
123	2904	76	84	104
124	2568	76	84	104
125	2514	76	84	104

In this input file, 25 pallets require transportation. The pallets are sorted in order of descending weight (in pounds). The cargo load height, width, and length are given in inches.

Appendix C – SALP-TS Pseudo-code

```
Read Pallet Data
Read Aircraft Data
Create and initialize solutions—Incumbent, Best Neighbor, Last Improvement

Calculate the Lower Bound:
 {
Three possible combinations of aircraft:
1.  The aircraft are the SAME type (each aircraft has the same number of pallet positions) and have the SAME ACL.
2.  The aircraft are the SAME type, but have DIFFERENT ACLs.
3.  The aircraft are DIFFERENT types with DIFFERENT ACLs.
        If (Combination 1)          // Determine lower bound
        {    # Aircraft Lower Bound = max ( $\lceil$total number pallets / number pallet positions in aircraft$\rceil$,
                                          $\lceil$total pallet weight / aircraft ACL$\rceil$        )
        }  // END If (Combination 1)
        Else if (Combination 2)     // Determine lower bound
        {    Lower Bound = Total number of aircraft
             While (can still remove aircraft)
             {  Get useable aircraft with smallest ACL
                   If  (can remove this aircraft and still meet total ACL AND number pallet position restrictions)
                   {   Decrement Lower Bound
                       Make Aircraft Unusable
                   }   // END if
                   Else {   cannot remove any more aircraft     }  // END Else
             }  // END While (can still remove aircraft)
        }  // END Else if  (Combination 2)
        Else  (Combination 3) // Determine lower bound
        {    While (can still remove aircraft)
             {     Get useable aircraft with *smallest* ratio of ACL to Number of Pallet Positions
                   If  (can remove this aircraft and still meet total ACL AND number pallet position restrictions)
                       {     Decrement Lower Bound
                             Make Aircraft Unusable
                       }     // END if
                       Else {     cannot remove any more aircraft           }  // END Else
             }  // End While (can still remove aircraft)
        }  // END Else (Combination 3)
        For each aircraft:          // All unusable aircraft need to be usable to get initial solution.
        {    If (aircraft is unusable)
             {     Make aircraft usable   }     // END If (aircraft is unusable)
        }  // END For each aircraft:
}  // END Calculate Lower Bound
```

```
Generate Initial Solution:
{ For 1 to total number of aircraft:
  {   Get useable aircraft with largest ratio of ACL to Number of Pallet Positions
      While ( (BigBin NOT empty) ∩ (All Aircraft NOT in State 4) )
      {   Increment iteration count (For TS memory structure use)
Four descriptive aircraft states during initial solution generation:
1. Not maximum for either weight or space.          2. Maximized for weight but not space.
3. Maximized for space but not weight.              4. Maximized for space and weight.
          If (State 1)
          {   For (each group)
              {   If (can add pallet without violating space and weight)
                  {   Add pallet to aircraft  }  // END If
              }  // END For (each group)
          }  // END If (State 1)
          If (State 2)
          {   Get heaviest pallet in aircraft, Remove pallet to BigBin
              Update Tabu Memory Structure (Make return move tabu)
              Get head pallet of lightest non-empty group
              While ((aircraft NOT maxed weight) ∩ (aircraft NOT maxed space) ∩ (NOT at END of BigBin list))
              {  If  ((aircraft can hold pallet) ∩ (NOT Tabu move) ){Add pallet to aircraft }// END If
              }  // END While
          }  // END If (State 2)
          If (State 3)
          {   Get lightest pallet in aircraft, Remove pallet to BigBin,
              Update Tabu Memory Structure (Make return move tabu)
              Get head pallet of heaviest non-empty group
              While ( (aircraft NOT maxed weight) ∩ (aircraft NOT maxed space) ∩ (NOT at END of BigBin list) )
              {   If ( (aircraft can hold pallet) ∩ (NOT Tabu Move) ) {   Add pallet to aircraft   }  // END If
                  Get next pallet
              } // END While
          }  // END If (State 3)
          If ((number loaded pallets unchanged) ∩ (cargo weight changed) )
          {   Increment CycleCount      }     // END If
          Else{ CycleCount = 0 }     // END If
          If  (( CycleCount > 10)  ∪ OR (Case 4))
          {   EXIT While (BigBin NOT empty) loop }          // END If
      }  // END While ( (BigBin NOT empty) ∩ (All Aircraft NOT in State 4) )
  }  // END For 1 to total number of aircraft
} // END Generate Initial Solution

Output Initial Solution
```

```
Commence Local Search
{    Create and Initialize Tabu Memory Structure, Initialize Variables
     Adjust CB:
     { For (each aircraft)
          {    While ((Number of Trial Improvements < 5) AND (Number of Disimproving Moves = 0) )
               { Perform Intra Aircraft Swap-Select best possible swap between pallets on same aircraft.     }  // END While
          }  // END For (each aircraft)
     }  // END Adjust CB
     Set Iterate Count = 0
     Set Fix CB Count = 0
     Dynamic Neighborhood Search:
     {  While ( (Sequential Disimproving Move Count<=20) AND (Sequential Trivial Move Count<=20) AND (Iterate Count<=500) )
     {    // Increment Iterate Count
Four possible solution states:
1.  BigBin is NOT empty               2.  (Aircraft CB is out of window) ∩ (Fix CB Count <= 5)
3.  {Weight Usage of Aircraft ? 25% }
∪ { [Number of Non-Empty Aircraft ? Lower Bound on Number of Aircraft]  ∩ [15 Adjacent Trivial Improvements ] }
∪ { [Number of Non-Empty Aircraft ? Lower Bound on Number of Aircraft]  ∩ [15 Sequential Disimproving Moves ] }
4.  Neither State 1, 2 or 3
          If (State 1)
          {    Set Fix CB Count = 0
               While (BigBin NOT empty)
               {    If (still empty pallet positions on *any* usable aircraft)
                    {    Get aircraft with ( (maximum weight capacity remaining) AND (one or more pallet positions empty) )  }
                    Else
                    {    Get *unusable* aircraft with *largest* ratio of ACL to Number of Pallet Positions.  Make aircraft usable.    }
                    Get heaviest pallet in BigBin. Add pallet to Aircraft.
                    Update Tabu Memory Structure (pallet cannot return to BigBin for Tabu Tenure iterations)
               }  // END While (BigBin NOT empty)
          }  // END If (State 1)
          Else if (State 2)
          {    Increment Fix CB Count
               Perform Intra-Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert within an aircraft)
               If (Disimproving Move)
               {    Undo move.  Fix CB Count = 6    } // END If (Disimproving Move)
               Else
               {    Update Tabu Memory Structure (pallet may not return to its *previous position* for Tabu Tenure iterations)
               }    // END Else
          }  // END Else If (State 2)
```

```
            Else If (State 3)
            {     Set Fix CB Count = 0
                  Get lightest loaded, useable aircraft.  Make aircraft unusable.
                  While (aircraft not empty)
                  {     Remove heaviest pallet, Insert into lightest loaded, useable aircraft
                        Update Tabu Memory Structure (pallet may not return to *any position* in the aircraft for Tabu Tenure iterations).
                  } // End While (aircraft not empty)
            }  // END Else If (State 3)
            Else (State 4)
            {     Set Fix CB Count = 0
                  Perform an Inter Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert between two aircraft)
                  Update Tabu Memory Structure (pallet may not return to *any position* in losing aircraft for Tabu Tenure iterations)
            }  // END Else (State 4)
            If ( (New Solution OF Value*1.025) < Previous Solution OF Value)
            { Increment Improving Move Count, Disimproving Move Count = 0 }   // END If
            Else If ( New Solution OF Value < Previous Solution OF Value)
            { Increment Trivial Move Count, Disimproving Move Count = 0 }         // END Else If
            Else { Increment Disimproving Move Count, Improving Move Count = 0}  // END Else
        }  // END While
      } // END Dynamic Neighborhood Search
}  // END Commence Local Search

Output Best Solutions
```

Appendix D – FS Location (in inches) for CG of Pallet Positions

Position	C-130 E/H	C-141	C-17:	C-17:	C-5	KC-10A:	KC-10A	KC-135E	KC-135R
Number			**Logistics**	**Airdrop**		**17 Pallets**	**23 Pallets**		
1	330	381	444	401	443	1011	684	505	505
2	420	471	444	491	443	1011	684	615	615
3	510	561	554	581	566	1120	703	725	725
4	600	651	554	671	566	1120	703	835	835
5	690	741	664	761	656	1229	902	955	955
6	803	831	664	851	656	1229	902	1065	1065
7		921	774	941	746	1338	1011		
8		1011	774	1031	746	1338	1011		
9		1101	884	1121	836	1447	1120		
10		1191	884	1229	836	1447	1120		
11		1281	994	1319	926	1556	1229		
12		1371	994		926	1556	1229		
13		1472	1104		1016	1665	1338		
14			1104		1016	1665	1338		
15			1227		1106	1774	1447		
16			1227		1106	1774	1447		
17			1337		1196	Not Used	1556		
18			1337		1196		1556		
19					1286		1665		
20					1286		1665		
21					1376		1774		
22					1376		1774		
23					1466		Not Used		
24					1466				
25					1556				
26					1556				
27					1646				
28					1646				
29					1736				
30					1736				
31					1826				
32					1826				
33					1916				
34					1916				
35					2065				
36					2065				

NOTE: Although the KC-10A has models with 17 and 23 pallet positions available, the last position is located immediately adjacent to the boom operator and is not typically used for transportation.

Appendix E – Aircraft CB Allowable Windows and Target Locations

Cabin Load	C-130 E/H		C-141		C-17		C-5		KC- 135E		KC- 135R	
(pounds)	Window	Target	Window	Target	Window	Target	Window	Target	Window	Target	Window	Target
0-5,000	400-550	475	630-1000	815	355-1080	717.5	400-1000	1000	609-878	744	425-1062	744
5,001-10,000	400-550	475	770-1000	885	370-1020	695	400-1000	1000	723-867	795	631-963	797
10,001-15,000	475-503	502.5	770-1000	885	380-985	682.5	400-1000	1000	761-864	813	700-929	815
15,001-20,000	485-530	507.5	835-970	902.5	380-985	682.5	400-1370	1040	780-862	821	734-913	824
20,001-25,000	485-530	507.5	835-970	902.5	450-960	705	400-1370	1040	791-861	826	755-903	829
25,001-30,000	510-530	520	860-960	910	520-955	737.5	670-1380	1153	799-860	830	768-896	832
30,001-40,000	515-530	522.5	870-950	915	610-950	780	835-1380	1207	808-859	834	785-888	837
40,001-50,000			890-950	920	660-945	802.5	935-1380	1242				
50,001-60,000			890-940	915	700-940	820	1000-1380	1265				
60,001-70,000					730-935	832.5	1050-1389	1280				
70,001-80,000					745-935	840	1085-1390	1293				
80,001-90,000					760-935	847.5	1115-1390	1309				
90,001-100,000					775-930	852.5	1135-1390	1320				
100,001-120,000					800-925	862.5	1170-1390	1324				
120,001-150,000					830-915	872.5	1200-1390	1332				
150,001-175,000					830-915	872.5	1225-1390	1338				
175,001-245,000							1280-1390	1345				
245,001-291,000							1315-1390	1350				

Appendix F – Sample DALP-TS Aircraft Input File

Aircraft ID Number	Aircraft Type	ACL	RLD	Travel Time	Crew Rest Time
10001	1	90000	1	1	1
10002	3	150000	1	1	1
10003	1	90000	1	1	1
10004	3	150000	1	1	1
10005	1	90000	1	1	1
10006	3	150000	1	1	1
10007	1	90000	1	1	1
10008	3	150000	1	1	1
10009	1	90000	1	1	1
10010	3	150000	1	1	1
10011	1	90000	1	1	1
10012	3	150000	1	1	1
10013	1	90000	1	1	1
10014	3	150000	1	1	1
10015	1	90000	1	1	1
10016	3	150000	1	1	1
10017	1	90000	1	1	1
10018	3	150000	1	1	1
10019	1	90000	1	1	1
10020	3	150000	1	1	1

In this input file, 20 aircraft are available. As explained in Table 3.1, aircraft of type 1 are C-17s and type 3 are C-5s. In this input file, each C-17 has a 90,000 pound ACL while each C-5 has a 150,000 pound ACL. Additionally, each aircraft has an initial RLD of day 1, a travel time from APOE to APOD of 1 day, and a crew rest time at the APOD of 1 day. The travel time from APOD to APOE is assumed to be equivalent to the travel time from APOE to APOD.

Appendix G – Sample DALP-TS Pallet Input File

Pallet ID Num	Weight	Cargo Height	Cargo Width	Cargo Length	ALD	EAD	LDD	RDD
101	10000	76	84	104	1	2	3	3
102	6000	76	84	104	1	2	3	3
103	5000	76	84	104	1	2	3	3
104	4000	76	84	104	1	2	3	3
105	2600	76	84	104	1	2	3	3
106	2500	76	84	104	1	2	3	3
107	10000	76	84	104	3	4	7	7
108	7500	76	84	104	3	4	7	7
109	6800	76	84	104	3	4	7	7
110	6000	76	84	104	3	4	7	7
111	5000	76	84	104	3	4	7	7
112	3850	76	84	104	3	4	7	7
113	3575	76	84	104	3	4	7	7
114	3500	76	84	104	3	4	7	7
115	3000	76	84	104	3	4	7	7
116	5000	76	84	104	4	5	7	9
117	4750	76	84	104	4	5	7	9
118	5000	76	84	104	4	7	9	9
119	4000	76	84	104	4	7	9	9
120	3500	76	84	104	4	7	9	9
121	3000	76	84	104	4	7	9	9
122	2500	76	84	104	4	7	9	9
123	2500	76	84	104	4	7	9	9
124	10000	76	84	104	5	6	8	8
125	9000	76	84	104	5	6	8	8

In this input file, 25 pallets require transportation. The pallets are first sorted in order of ascending ALD, EAD, LDD, and RDD, respectively, then by descending weight (in pounds). The cargo load height, width, and length are given in inches.

Appendix H – DALP-TS Pseudo-code

```
Read Pallet Data
Read Aircraft Data
Create and initialize solutions—Incumbent, Best Neighbor, Last Improvement

Generate Initial Solution:
{ For 1 to total number of pallets):
  {   Get pallet at head of BigBin.

      Three possible states for pallet/aircraft combination:
      1. (aircraft exists with ( (acceptable departure and arrival dates) ∩ (sufficient space and weight remaining)
             ∩ (at least one pallet loaded) ) )
      2. If (empty aircraft exists with ( (acceptable departure and arrival dates) ∩ (Aircraft Instance > 1) ) )
      3. Neither State 1 or 2
      If (State 1)
      {     Select applicable aircraft with ( (largest ratio of ACL to Number of Pallet Positions)
                 ∩   (smallest weight capacity remaining)          )          }          // END If
      Else If (State 2)
      {     Select applicable aircraft with (largest instance number) ∩ (largest ratio of ACL to Number of Pallet Positions)
      }     // END Else If
      Else (State 3)
      {     Select new aircraft      }          // END Else

      Remove pallet from BigBin and insert into first available position in Aircraft

      If ( (State 2) ∪ (State 3) )
      {     Increment Number of Flights Required.
            If (State 3)
            {     Increment Number of Aircraft Used.     }          // END If
            Set Aircraft Arrival Date = Pallet Latest Delivery Date.
            Set Aircraft Departure Date = Aircraft Arrival Date – Aircraft Travel Time.
            Insert additional instance of aircraft.
      }     // END If (Number of Loaded Pallets in Aircraft == 1)

  }  // END For 1 to total number of pallets
} // END Generate Initial Solution

Output Initial Solution
```

```
Commence Local Search
{    Create and Initialize Tabu Memory Structure, Initialize Variables
     Adjust CB:
     { For (each aircraft)
          {    While ((Number of Trial Improvements < 5) AND (Number of Disimproving Moves = 0) )
               { Perform Intra Aircraft Swap-Select best possible swap between pallets on same aircraft.        }  // END While
          }  // END For (each aircraft)
     }  // END Adjust CB
     Set Iterate Count = 0
     Set Fix CB Count = 0
     Set Count Complete Unload = 0
     Set Exit Criteria = False
     Dynamic Neighborhood Search:
     {  While (Exit Criteria == False )
     {     Increment Iterate Count

          Three possible solution states:
          1.  (Aircraft CB is out of window) $\cap$ (Fix CB Count <= 5)
          2.  { (15 Adjacent Trivial Improvements) $\cup$ (15 Sequential Disimproving Moves) }
          3.  Neither State 1 or 2
          If (State 1)
          {    Increment Fix CB Count
               Perform Intra-Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert within an aircraft)
               If (Disimproving Move)
               {    Undo move.  Fix CB Count = 6    } // END If (Disimproving Move)
               Else
               {    Update Tabu Memory Structure (pallet may not return to its *previous position* for Tabu Tenure iterations)
               }    // END Else
          }  // END Else If (State 2)
          Else If (State 2)
          {    Set Fix CB Count = 0
               Get lightest loaded, useable aircraft.  Make aircraft unusable.
               While (aircraft not empty)
               {    Remove heaviest pallet, Insert into lightest loaded, useable aircraft
                    Update Tabu Memory Structure (pallet may not return to *any position* in the aircraft for Tabu Tenure iterations).
               } // End While (aircraft not empty)
          }  // END Else If (State 2)
          Else (State 3)
          {    Set Fix CB Count = 0
               Perform an Inter Aircraft Insert/Swap (select best possible non-null and non-tabu swap or insert between two aircraft)
               Update Tabu Memory Structure (pallet may not return to *any position* in losing aircraft for Tabu Tenure iterations)
          }  // END Else (State 4)
```

```
            If ( (New Solution OF Value*1.025) < Previous Solution OF Value) )
            { Increment Improving Move Count, Disimproving Move Count = 0 }   // END If
            Else If ( New Solution OF Value < Previous Solution OF Value)
            { Increment Trivial Move Count, Disimproving Move Count = 0 }        // END Else If
            Else { Increment Disimproving Move Count, Improving Move Count = 0}  // END Else

            If ( (Count Disimproving Move == 20) ∪ (Count Trivial Improvement == 20) ∪ (Count Complete Unload == 2)
            ∪ (Iterate Count == 500) )
            {     Exit Criteria = true      }     // END If
        }  // END While (Exit Criteria == False )
      } // END Dynamic Neighborhood Search
}  // END Commence Local Search

Output Best Solutions
```

Acronyms

Acronym:	Definition:
1-D BPP	1-Dimensional Bin Packing Problem
2-D BPP	2-Dimensional Bin Packing Problem
3-D BPP	3-Dimensional Bin Packing Problem
AALPS	Automated Air Load Planning Software
ACL	Available Cabin Load
ACSP	Air Crew Scheduling Problem
AD	Arrival Date
AF	Air Force
AFCSP	Aerial Fleet Crew Scheduling Problem
AFRC	Air Force Reserve Command
AFRP	Aerial Fleet Refueling Problem
AFSAA	Air Force Studies and Analysis
AIL	Aircraft Identification Letter
ALD	Available Load Date
ALM	Air Load Model
ALP	Airlift Loading Problem
ALP-TS	Airlift Loading Problem-Tabu Search
AMC	Air Mobility Command
AMOS	Air Mobility Operations Simulation
ANG	Air National Guard
AOR	Area of Responsibility
APOD	Aerial Port of Debarkation
APOE	Aerial Port of Embarkation
APP	Aircraft Packing Problem
ASP	Aircraft Selection Problem
ATS	Adaptive Tabu Search
B&B	Branch and Bound
BIIN	Bulk Item Identification Number
B-L	Bottom-Left
C-130	C-130 Hercules
C-17	C-17A Globemaster III

Acronym:	Definition:
C-5	C-5B Galaxy
CALM	Computer Aided Load Manifesting
CB	Center of Balance
CCGA	Cooperative Co-evolutionary Genetic Algorithm
CG	Center of Gravity
CIN	Cargo Identification Number
CINC	Commander in Chief
CLP	Cargo Loading Problem
CONUS	Continental United States
COP	Combinatorial Optimization Problem
CPP	Cargo Partitioning Problem
CRAF	Civil Reserve Air Fleet
DALP	Dynamic Airlift Loading Problem
DD	Departure Date
DMES	Deployable Mobility Execution System
DOD	Department of Defense
DPP	Distributor's Packing Problem
EAD	Earliest Arrival Date
FS	Fuselage Station
GA	Genetic Algorithm
GAP	Generalized Assignment Problem
GLS	Guided Local Search
GTTS	Group Theoretic Tabu Search
HA	Heuristic Algorithm
HAZMAT	Hazardous Materials
ID	Identification
JFAST	Joint Flow and Analysis System for Transportation
KP	Knapsack Problem
LB	Lower Bound
LAD	Latest Arrival Date
MBPP	Modified Bin Packing Problem
MCPP	Multiple Container Packing Problem
MOG	Maximum-On-The-Ground
MPP	Manufacturer's Packing Problem

Acronym:	Definition:
MSC	Military Sealift Command
OF	Objective Function
OSA	Operational Support Airlift
P|O	Partitioning and Ordering
PAX	Passengers
PP	Packing Problem
PPP	Pallet Packing Problem
RDD	Required Delivery Date
RDL	Reference Datum Line
RLD	Ready to Load Date
RTS	Reactive Tabu Search
SA	Simulated Annealing
SAAM	Special Assignment Airlift Mission
SAM	Special Air Mission
SALP	Static Airlift Loading Problem
SAP	Strategic Airlift Problem
SCP	Set Covering Problem
SDDC	Surface Deployment and Distribution Command
SMMSP	Strategic Mobility Mode Selection Problem
SPP	Set Partitioning Problem
TALCE	Tanker Airlift Control Element
TDVRSP	Theater Distribution Vehicle Routing and Scheduling Problem
TPFDD	Time-Phased Force Deployment Document
TS	Tabu Search
TSP	Traveling Salesman Problem
US	United States
USAF	United States Air Force
USTRANSCOM	United States Transportation Command

References

Abramson, David and Marcus Randall. 1999. A simulated annealing code for general integer linear problems. *Annals of Operations Research* 86:3-21.

AFDD 1. 2003. Air Force Basic Doctrine, Complement of Joint Publication 1: Joint Warfare of the Armed Forces of the United States.

AFDD 2-6.1. 1999. Airlift Operations: Air Force Doctrine Document 2-6.1, Part of Joint Publication 3-17, Joint Tactics, Techniques, and Procedures for Air Mobility Operations.

Air Force Link. Retrieved on August 15, 2006 from http://www.af.mil/pressreleases/release.asp?storyID=123011597.

Air Force Modeling and Simulation Resource Repository. Retrieved on August 15, 2006 from http://afmsrr.afams.af.mil/index.cfm.

Automated Air Load Planning System. 2004. Retrieved on August 15, 2006 from http://www.tis.army.mil/AALPS/default.htm.

Battiti, Roberto and G. Tecchiolli. 1994. The Reactive Tabu Search. *ORSA Journal on Computing* 6(2): 126-140.

Beasley, J. E. 1990. OR-Library: Distributing test problems by electronic mail. *Journal of Operational Research Society* 41(11): 1069-1072.

Berkey, J. O. and P.Y. Wang. 1987. Two Dimensional Finite Bin Packing Algorithms. *Journal of Operational Research Society* 38: 423-429.

Bhatia, A.K. and S.K. Basu. 2004. Packing Bins Using Multi-Chromosomal Genetic Representation and Better-Fit Heuristic. *Lecture Notes in Computer Science* 3316: 181-186.

Brusco, M.J., G.M. Thompson, and L.W. Jacobs. 1997. A morph-based simulated annealing heuristic for a modified bin-packing problem. *Journal of the Operational Research Society* 48: 433-439.

Bourjolly, Jean-Marie and Vianney Rebetez. 2005. An analysis of lower bound procedures for the bin packing problem. *Computers & Operations Research* 32: 395-405.

Callander, Bruce D. 1998. The Evolution of Air Mobility, *Air Force Magazine Online*, Journal of the Air Force Association **81**(2). (http://www.afa.org/magazine/Feb1998/0298evolution.asp)

Chocolaad, Christopher A. 1998. *Solving Geometric Knapsack Problems using Tabu Search Heuristics*. M.S. Thesis, Air Force Institute of Technology.

Cochard, Douglas D. and Kirk A. Yost. 1985. Improving Utilization of Air Force Cargo Aircraft, *Interfaces* 15(1): 53-68

Computer Sciences Corporation. 1997. Unit Type Code Development, Tailoring, And Optimization (UTC-DTO) Phase 2 Final Report. Contract DCA100-94-D-00144. Falls Church, VA: Defense Enterprises Integration Services.

Colletti, Bruce W. 1999. *Group Theory and Metaheuristics*. Ph.D. Dissertation, The University of Texas at Austin.

Combat Aircraft Reference Guide. Retrieved on August 15, 2006 from http://www.inetres.com/gp/military/ar/misc/463L.html.

Combs, Todd. 2002. *A Group Theoretic Tabu Search Methodology for Solving the Crew Scheduling Problem with an Air Tanker Crew Application*. Ph.D. Dissertation, AIR FORCE INSTITUTE OF TECHNOLOGY.

Crino, John R. 2002. *A Group Theoretic Tabu Search Methodology for Solving Theater Distribution Vehicle Routing and Scheduling Problems*. Ph.D. Dissertation, AIR FORCE INSTITUTE OF TECHNOLOGY.

Defense Transportation Regulation, DOD Regulation 4500.9-R-Part III (DTR 450.9-R-Part III). 2004. Mobility.

Eilon, Samuel and Nicos Christofides. 1971. The loading problem. *Management Science* 17: 259-267.

Eglese, R.W. 1990. Simulated Annealing: A tool for Operational Research. *European Journal of Operational Research* 46: 271-281.

Elhedhli, Aamir. 2003. Ranking lower bounds for the bin-packing problem. *European Journal of Operational Research*, v160, 34-46.

Garey, Michael R. and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman. San Francisco, CA.

Giangreco, DM, and Robert E. Griffin. 1988. *Airbridge to Berlin—The Berlin Crisis of 1948, its Origins and Aftermath*. Presidio Press, Novato, CA.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. Reading, MA.

Government Accountability Office. 2005. Air Mobility Command Needs to Collect and Analyze Better Data to Assess Aircraft Utilization, Report to the US Secretary of Defense.

Guéret, C, N. Jussien, O. Lhomme, C. Pavageau and C Prins. 2003. Loading aircraft for military operations. *Journal of the Operational Research Society* 54: 458-465.

Harwig, John J. 2003. *An Adaptive Tabu Search Approach to Cutting and Packing Problems*. Ph.D. Dissertation, The University of Texas at Austin.

Haupt, Randy L. and Sue Ellen Haupt. 2004. *Practical Genetic Algorithms*. John Wiley & Sons, Inc. New York, NY.

Heidelberg, Kurt R., Gregory S. Parnell and James E. Ames IV. 1998. Automated Air Load Planning. *Naval Research Logistics* 45: 751-768.

Kinney, Gary J., Jr. 2005. *A Group Theoretic Approach to Metaheuristic Local Search for Partitioning Problems*. Ph.D. Dissertation, The University of Texas.

Koulmas, C., Antony, S.R. & Jaen, R. 1994. A Survey of Simulated Annealing Applications to Operations Research Problems, *Omega International Journal of Management Science* 22(1): 41-56.

Labbe, M., G. Laporte, H. Mercure. 1991. Capacitated vehicle on trees. *Operations Research* 39:616-622.

Lambert, Garry R. 2004. *A Tabu Search Approach to the Strategic Airlift Problem*. Ph.D. Dissertation, The University of Texas at Austin.

Lambert, Garry R., and J. Wesley Barnes. 2006. A Tabu Search Approach to the Strategic Airlift Problem, in second review with the *Journal of. Military Operations Research*: Submitted article in second review.

Lewis, James E., Rammohan K. Ragade, Anup Kumar, and William E. Biles. 2005. A distributed chromosome genetic algorithm for bin-packing. *Robotics and Computer-Integrated Manufacturing* 21: 486-495.

Lodi, Andrea, Silvano Martello, and Daniele Vigo. 1999. Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. *INFORMS Journal on Computing* 11(4): 345-357.

Lodi, Andrea, Silvano Martello, and Michele Monaci. 2002. Two-dimensional packing problems: A Survey. *European Journal of Operational Research* 141: 241-252.

Lodi, Andrea, Silvano Martello, and Daniele Vigo. 2002. Recent Advances on Two-Dimensional Bin Packing Problems. *Discrete Applied Mathematics* 123: 379-396.

Lodi, Andrea, Silvano Martello, and Daniele Vigo. 2004. TSPack: A Unified Tabu Search Code for Multi-Dimensional Bin Packing Problems. *Annals of Operations Research* 131: 203-213.

Martello, Silvano and P. Toth. 1990. Lower Bounds and Reduction Procedures for the Bin Packing Problem. *Discrete Applied Mathematics* 28: 59-70.

Martello, Silvano and Daniele Vigo. 1998. Exact Solution of the Two-Dimensional Finite Bin Packing Problem. *Management Science* 44(3): 388-399.

McKinzie, Kaye. 2005. *A Tabu Search Approach to Strategic Mobility Mode Selection*. Ph.D. Dissertation, The University of Texas at Austin.

McKinzie, Kaye and J. Wesley Barnes. 2006. A Tabu Search Approach to the Strategic Mobility Mode Selection Problem,. *Air Force Journal Of Logistics*, (in press).: Accepted for publication.

Nanry, W. P. and J. Wesley Barnes. 2000. Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search, *Transportation Research Part B* 34:107-121.

Pimpawat, Chaiwat and Nachol Chaiyaratana. 2004. Three-Dimensional Container Loading Using a Cooperative Co-evolutionary Genetic Algorithm. *Applied Artificial Intelligence* 18: 581-601.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press. Cambridge, MA.

Raidl, Gunther R. and Gabriele Kodydek. 1998. Genetic Algorithms for the Multiple Container Packing Problem. *Lecture Notes in Computer Science* 1498: 875-884.

Romaine, Jonathan M. 1999. *Solving the Multidimensional Multiple Knapsack Problem with Packing Constraints using Tabu Search*. M.S. Thesis, Air Force Institute of Technology.

United States Transportation Command. Retrieved on August 15, 2006 from http://www.transcom.mil/organization.cfm.

Wikipedia: The Free Encyclopedia. Retrieved on August 15, 2006 from http://en.wikipedia.org/wiki/Force_projection.

Wiley, Victor. 2001. *The Aerial Fleet Refueling Problem*. Ph.D. Dissertation, The University of Texas at Austin.

Vita

August Gibson Roesener was born on April 18, 1976 in Topeka, KS, the son of Stacey L. Roesener and Kathy S. Stump. After completing his work at the Oklahoma School of Science and Mathematics, Oklahoma City, OK, in 1994, he entered the United States Air Force Academy, Colorado Springs, CO. He received a Bachelors of Science in Operations Research in 1998 and was commissioned as a Second Lieutenant in the United States Air Force. While performing duties as Chief, Operational Analysts Branch at Eglin AFB, FL, August began studies at the University of Florida's Graduate Engineering and Research Center. He graduated with a Masters of Science in Industrial and Systems Engineering in 2002. August began his doctoral studies at the University of Texas at Austin in the month of August of 2003. His subsequent assignment is to the Air Force Institute of Technology, Wright-Patterson AFB, OH, as an instructor in the Operational Sciences department.

Permanent Address: 18702 NE 85th Ave, Battle Ground, WA 98604

This dissertation was typed by the author.