A REACTIVE TABU SEARCH METAHEURISTIC EXTENSION OF THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

THESIS

Ümit Hilmi TEKELİOĞLU, First Lieutenant, TUAF

AFIT/GOR/ENS/01M-16

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GOR/ENS/01M-16

A REACTIVE TABU SEARCH METAHEURISTIC EXTENSION OF THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Ümit Hilmi TEKELİOĞLU

First Lieutenant, TUAF, B.S.

March 2001

AFIT/GOR/ENS/01M-16

A REACTIVE TABU SEARCH METAHEURISTIC EXTENSION OF THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

Ümit Hilmi TEKELİOĞLU, B.S.
First Lieutenant, TUAF

Approved:

James T. Moore — 1 Mar 01

James T. Moore, Ph.D. (Advisor) — date

Raymond Hill — 1 Mar 01

Raymond Hill, Ph.D., Lt Col, USAF (Reader) — date

William Nanry — 1 MAR 01

William Nanry, Ph.D., LTC (P), US ARMY (Reader) — date

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Moore and my readers, Lt. Col. Hill and LTC (P) Nanry, for reading many of my works through this process. They provided all the motivation and help I needed.

I would like to thank my wife, Özlem, and my baby-girl Nazlı, for being with me all the time. And without the help of Erhan, Hakan, İbrahim, Kaan, and Engin this thesis would be a difficult experience for me.

Ümit Hilmi TEKELİOĞLU

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS ..... **IV**

LIST OF FIGURES ..... **VII**

LIST OF TABLES ..... **VIII**

ABSTRACT ..... **X**

CHAPTER 1. INTRODUCTION **1**

1.1 BACKGROUND 1
1.2 PROBLEM STATEMENT 3
1.3 RESEARCH OBJECTIVES 4
1.4 SCOPE 4
1.5 CONTRIBUTION OF RESEARCH 6
1.6 REPORT OVERVIEW 6

CHAPTER 2. LITERATURE REVIEW **7**

2.1 TANKER SCHEDULING TOOLS 7
2.1.2 Hostler's Air Refueling Tanker Scheduling Tool *8*
2.1.3 Quick Look Tool for Tanker Deployment *9*
2.1.4 TAP Tool *10*
2.1.5 AMC's Efforts on TAP Tool *13*
2.2 SCHEDULING THEORY 13
2.2.1 Parallel Machine Models *14*
2.2.1.1 Precedence Constraints 14
2.3 HEURISTIC APPROACHES 15
2.3.1 Tabu Search *15*
2.3.2 RTS *20*
2.3.3 Candidate List Strategies *21*
2.3.3.1 Elite Candidate List 22
2.4 CONCLUSION 22

CHAPTER 3. METHODOLOGY **23**

3.1 VERIFICATION & VALIDATION EFFORT 23
3.1.1 Coordinate Problem *24*
3.1.2 Modification of the Objective Function Evaluation *24*
3.1.2.1 Old Mission Evaluation 25
3.1.2.2 New Mission Evaluation 27
3.2 TABU SEARCH IMPLEMENTATION 29
3.3 REACTIVE TABU SEARCH IMPLEMENTATION 32

3.3.1 Memory Usage *38*
3.4 ELITE CANDIDATE LIST STRATEGY 38
CHAPTER 4. RESULTS AND CONCLUSIONS 43
4.1 SOUTHEAST ASIA DEPLOYMENT 43
4.2 SOUTHEAST ASIA DEPLOYMENT WITHOUT GREEDY CONSTRUCTION 54
4.3 DECREASING THE AVAILABLE TANKERS FOR SOUTHEAST ASIA DEPLOYMENT 58
4.4 CONCLUSION 62
4.4.1 Problems with the TAP Tool *63*
CHAPTER 5. FUTURE RESEARCH 65
5.1 RECOMMENDATIONS 65
APPENDIX A 67
BIBLIOGRAPHY 99

List of Figures

Page

Figure 1: Tanker Escort of an F-16 Fighter 11

Figure 2: Tanker Escort of a B-52 Bomber 12

Figure 3: Illustration of coordinate change from point 1 to point 2. 24

Figure 4: Sending one F-16 from KBAD to KAST 28

Figure 5: Sending one F-16 from KBAD to PHIK 28

Figure 6: Tanker Routes for one F-16 from KBAD to PHIK 31

Figure 7: Revised TAP Tool Flowchart 36

Figure 8: RTS Flowchart 37

Figure 9: Number of Solutions for all neighborhoods 41

Figure 10: Number of Solutions after Forming Candidate List 42

Figure 11: Southeast Asia Deployment 44

Figure 12: Initial Mission Schedule for South East Deployment in RTS 48

Figure 13: Best Mission Schedule for South East Deployment in RTS 49

Figure 14: Initial Schedule for South East Deployment in RTS without Greedy 57

Figure 15: Best Schedule for South East Deployment in RTS without Greedy 58

Figure 16: Best Schedule for South East Deployment with 5 Tankers in Each Base 61

List of Tables

Page

Table 1: Tanker deliverable fuel capacities 2

Table 2: Escort Requirements 5

Table 3: Example for a non-cycling Series 19

Table 4: Example for a Cycling Series 19

Table 5: Bases Activated for Neighborhood Scenario 39

Table 6: Bases supporting RPs for Example Deployment 39

Table 7: Objective Function Evaluations for Example Deployment 40

Table 8: Bases supporting RPs after Elite Candidate List 40

Table 9: Objective Function Evaluations after Elite Candidate List 41

Table 10: Receiver Groups for Southeast Asia Deployment 43

Table 11: Number of Aircraft on the bases used for Southeast Asia Deployment 45

Table 12: Initial Solution of Southeast Asia Deployment in RTS 46

Table 13: Solution of Southeast Asia Deployment after 100 iteration in RTS 47

Table 14: Timeline for Initial and Best Mission Plan in RTS 48

Table 15: Initial Mission Plan for South East Deployment in TAP Tool 50

Table 16: Initial TOD and TOD for South East Deployment in TAP Tool 51

Table 17: Final TOD and TOA for South East Deployment in TAP Tool 51

Table 18: Best Mission Plan Evaluation for South East Deployment in TAP Tool .. 52

Table 19: General Comparison of TAP Tool and RTS 53

Table 20: Initial Solution of Southeast Asia Deployment without greedy approach 55

Table 21: Solution after 100 iteration without greedy 56

Table 22: Timeline for Initial and Best Mission Plan in RTS, without Greedy 57

Table 23: Decreasing the Number of Aircraft on the bases 58

Table 24: Initial Solution of RTS with 5 tankers in each base 59

Table 25: Best Solution after 100 Iterations with 5 tankers in each base 60

Table 26: TOD and TOA with 5 Tankers in Each Base 61

ABSTRACT

The Combined Mating and Ranging Planning System (CMARPS) is the system used by AMC to schedule air refueling for deployments from the continental US to other parts of the world. The tool developed by Capehart (2000) provided results similar to CMARPS in less time. Capehart's tool allows AMC to input several receiver groups consisting of various aircraft types and numbers. Each receiver group contains a point of origin and destination, with the option of providing one waypoint along the path, a ready to load date (RLD) and required delivery date (RDD). The user is also able to specify the locations of military tanker aircraft. The main goal of this tool is to assign the tankers to the different refueling points of the receiver groups so that all receiver groups arrive before their RDD. Secondary goals include the reuse of tankers and limiting the total flight distance for all tanker aircraft. The main purpose of this research is to introduce a dynamic feature of tabu search, reactive tabu search, into the tool. This method changes tabu tenure when necessary in the hope of finding better solutions by diversifying the search to the unexplored areas of the solution space.

A REACTIVE TABU SEARCH METAHEURISTIC EXTENSION OF THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

CHAPTER 1. INTRODUCTION

1.1 BACKGROUND

The United States Air Force (USAF) wants to maintain a force structure prepared to fight and win two major regional wars at approximately the same time. The 1997 Quadrennial Defense Review (QDR) reinforced and institutionalized the two war strategy (A/TQ, 2000). In order to achieve this goal for aerial refueling missions, USAF desires a tool that helps to schedule the deployment of its fighter/bomber aircraft to areas of conflict around the globe. The word "deployment" refers to moving military forces so that they are ready to respond to a crisis (Applications of Metaheuristics to Air Force Problems, 2000).

The USAF's Air Mobility Command (AMC) typically uses tanker aircraft for refueling the Air Force's fighter/bomber aircraft as the aircraft deploy. All tanker operations are commanded from the Tanker Airlift Control Center (TACC) located at Scott Air Force Base, Illinois.

The KC-135 tankers were introduced in 1957 and were intended to accompany the B-52 bombers. In the early 1970's the Department of Defense (DoD) sought to expand its

tanker fleet by purchasing new generation wide-body commercial transports. The Air Force selected McDonnell-Douglas' DC-10, designated for military use as the KC-10, as an advanced tanker (Congressional Budget Office, 1982). Despite the fact that a good many bombers have been retired over the last 40 years, the USAF still needs a large number of tankers. With the ending of cold war, mission priorities and threat concepts of USAF changed tremendously. Humanitarian aids and first strike forces gained importance over other missions. For this reason, the need for tankers never decreased.

The KC-135 and KC-10 aircraft provide in-flight refueling for almost every aircraft in the Air Force. The KC-135 Stratotankers are used for these deployments. With in-flight refueling, non-stop flying capabilities increase greatly giving each aircraft the ability to reach its destination before its required arrival time. The amount of fuel that can be delivered by tanker aircraft, within a flying radius of 2,500Nm is shown in Table 1 (A/TQ, 2000).

Table 1: Tanker deliverable fuel capacities

Tanker type	Fuel (pounds)
KC-135A	63,000
KC-135R	94,500
KC-135E	75,600
KC-10	162,000

As forward basing and forward mobility give way to rapid mobility from the United States, the role of the tanker will become a critical part of almost every air operation. Air refueling allows combat aircraft to deploy, to strike targets deep in enemy territory and extends the time fighter aircraft can protect friendly forces from attack by enemy aircraft.

Nearly the entire USAF air refueling fleet was needed to execute the campaign during the Air War Over Serbia (AWOS), and Lieutenant General Michael Short said that

"without tankers we could not have fought this war" (A/TQ, 2000). Due to the limited number of tanker aircraft available compared to the number needed during an operational deployment, not all refueling requests can be met simultaneously. This may cause some deploying receiver groups to arrive at their deployment destinations later than scheduled. Therefore, the allocation and scheduling of tankers is extremely important in ensuring receivers are not tardy.

In order to compare the effort required by the Combined Mating and Ranging Planning System, (CMARPS) to that of the Tanker Assignment Problem (TAP) tool (Capehart, 2000), two simple mission plans were compared in Capehart's research. The computation time for the TAP tool was significantly less than that of CMARPS. This computational time benefit is of major interest to AMC.

1.2 PROBLEM STATEMENT

During crisis situations, air refueling tankers provide necessary support for deployment of combat and combat support aircraft. During Operation Allied Force (OAF), tanker aircraft transferred over 355,800,000 pounds of fuel during inflight refuelings (A/Q, 2000). To schedule these tanker aircraft, AMC uses CMARPS. This system provides actual tanker/receiver aircraft schedules, but it often takes a long time to generate a schedule. For some scenarios, it may take up to two weeks to schedule tankers and receivers for a deployment. It is desirable to find a tool that gives good solutions for deployments in a short time when compared with CMARPS.

Another drawback of CMARPS is that it is not interoperable with AMC's airlift simulation. Since tanker and airlift missions are interrelated and compete for limited airbase resources, there should be some interaction between the two tools.

1.3 RESEARCH OBJECTIVES

The objective of this research is to improve TAP tool performance by using Reactive Tabu Search (RTS). The introduction of intensification and diversification schemes through RTS is a positive contribution of this research. The elite candidate list strategy, which evaluates the most promising regions of the solution space in order to decrease computation time, is also introduced.

While improving the TAP Tool, a strongly related task of the research is verification and validation (V&V) of the TAP Tool. V&V highlights the strengths and weaknesses of the model. Comparing tools with some small deployments does verification and validation.

1.4 SCOPE

This problem is both an assignment and scheduling problem. KC-135 tankers need to be assigned to receiver groups. So, this problem is an assignment problem. Since tankers have assigned times to meet with receiver groups, it is also a scheduling problem.

The receiver groups consist of both fighter and bomber aircraft. Fighters need tanker escort while flying over open waters. Bombers do not need escort. The aircraft

types that need tanker escort are shown as "Y", and the aircraft types that do not need escort are shown as "N" in Table 2.

Table 2: Escort Requirements

A/OA10	Y	AC130	N
AV8	Y	B1	N
EA6	Y	B2	N
F/A18	Y	B52	N
F117	Y	C141	N
F14	Y	C17	N
F15	Y	C5	N
F15E	Y	E3	N
F16	Y	E8	N
F18	Y	KC10	N
		MC130	N
		RC135	N

Factors effecting this problem include the tanker and receiver aircraft fuel capacities and burn rates, ground speed, true air speed, altitude, deployment distances, number of aircraft to be supported, time frames, locations of both tanker and receiver group origins and destinations, wind, escort requirement, formation size, and crew duty limitations.

Although altitude affects the fuel burn rate, we model a nominal flight altitude. Wind has an enormous effect on the fuel burn rate and ground speed of an aircraft, but this research does not incorporate the effect of wind. Flying an aircraft from the continental US to the other parts of the world is not an easy task for a pilot, as it requires about 10-13 hours of continuous flying. Flight and duty time regulations can be interpreted as a means of ensuring that reasonable minimum rest periods are provided to crews (NASA Technical Memorandum, 1991). Crew duty limitations of the fighter pilots are not accounted for, even though three hours of turn around time for the tankers is assumed.

1.5 CONTRIBUTION OF RESEARCH

This effort provides AMC with a quick running tanker assignment tool. The algorithm uses RTS and can serve as a starting point for a future tanker assignment tool that can be interoperable with an airlift model.

1.6 REPORT OVERVIEW

The remainder of this thesis is organized as follows; Chapter 2 reviews the literature. Chapter 3 reviews the verification and validation efforts that were applied to the TAP tool and describes the methodology for solving the tanker assignment problem with RTS. The results of this methodology are presented in Chapter 4. Finally, Chapter 5 concludes the research and discusses future research.

CHAPTER 2. LITERATURE REVIEW

This chapter begins with the introduction of the tanker scheduling tools. Then comes the basic principles of scheduling theory, and continues with the detailed explanations of heuristic search techniques such as tabu search, reactive tabu search, and elite candidate list strategies.

2.1 TANKER SCHEDULING TOOLS

A limited amount of research has been accomplished in this area. The first work is AMC's CMARPS. Hostler (1987) tried to schedule Strategic Air Command's (SAC) air refueling tanker fleet by using a preemptive goal programming approach. The other work is Russina, Ruthsatz, and Russ's Quick Look Tool for Tanker Deployment. Capehart extended the Quick Look Tool and developed the TAP Tool. AMC performed the most recent work in this area by incorporating a user-friendly interface to the TAP Tool.

CMARPS and the Quick Look Tool were good to some extent, but they had serious drawbacks, i.e. long runtimes, and not accounting for tankers located at different bases. Even though the TAP tool has some drawbacks, it certainly produces high quality solutions given the short time to find those solutions. The TAP tool is very attractive since it has decreased runtime compared to the other models. AMC worked on the TAP Tool, introduced a new spreadsheet, and made the tanker input easier for the user to access.

2.1.1 CMARPS

AMC uses CMARPS for scheduling tanker aircraft. This model was introduced in 1982 and assigns specific tankers to refueling points. The mission routing determines the fuel requirements for the receiver group. CMARPS routes the mission considering the following criteria: avoid restricted airspace, minimize threat exposure, deconflict routes in a strike zone, and satisfy time over targets. Once the fuel requirements are determined, CMARPS assigns tankers to meet the requirements. CMARPS does this by minimizing use of tanker resources, minimizing tanker fuel consumption, using air refuelable tankers, regenerating tankers for tanker reuse, and satisfying the abort base requirements. Once the tanker assignments are made, CMARPS simulates the aircraft mission using formulas for winds and fuel consumptions to generate a final mission schedule and flight plan (Logicon, 1996).

2.1.2 Hostler's Air Refueling Tanker Scheduling Tool

Hostler attempted to schedule SAC's air refueling tanker fleet to perform more than one refueling mission during a flight. He used a preemptive goal programming approach by using three priority levels. He considered three objectives: maximize the number of tanker requests satisfied, maximize the number of category B requests satisfied, and minimize the total flight time to perform all the missions.

Each request for a tanker is prioritized based on the type of training mission conducted by the receiver aircraft. Category A training is normal recurring air refueling training. Category B training is in support of formal course training, exercises,

predeployment air refueling, deployments, rotations, and tests (Air Force Regulation, 1:2-3).

A preemptive goal programming approach is used to solve the scheduling problem. This means that the desired goals to be achieved, when scheduling the tankers to the requests, are identified and prioritized according to their perceived importance. When solving the goal program, each of the objectives is optimized in order of priority, with the highest priority objective being optimized first (Winston, 1994).

Hostler developed a preprocessor to transform the inputs from the tanker and receiver scheduling units into a usable format executable by the mixed integer programming package. This preprocessor determined all of the possible refuelings that could take place, computed the flight times of the missions, and determined all of the variables to be used in the constraints and objective functions. By using a list of flight time constraints, all of the refuelings that are not possible are sifted out before they reach the integer program. This sifting allowed the integer program to work with only those refuelings that are possible.

2.1.3 Quick Look Tool for Tanker Deployment

Russina, Ruthsatz, and Russ (1999) provided a prototype tool to evaluate tanker allocation for the aircraft deployment mission. Their Quick Look Tool makes basic assumptions regarding aircraft capabilities and interactions between receiver groups and tankers. The Quick Look Tool functions as a relatively simple tool for modeling and predicting air refueling tanker capabilities for supporting deployment of combat and

combat support aircraft. The AMC tanker-scheduling problem involves a wide scope of system variables, constraints, and potential analysis areas. The example provided by Russina, Ruthsatz, and Russ presents an in-depth explanation of the deployment issues associated with tanker deployment.

The issues of tanker availability and reusing tankers are very important in scheduling a complex deployment. The scheduling precision in the Quick Look Tool is in terms of days, but it is more desirable to schedule missions in terms of hours or minutes. The user interface involves a Microsoft Excel Workbook with ten worksheets. All calculations made by the Quick Look Tool are done via Excel macros written in Visual Basic for Applications code. The current Quick Look Tool accounts for tanker availability on a day-by-day basis, and does not consider the tankers located at separate bases. These two issues are the major deficiencies in the Quick Look Tool.

2.1.4 TAP Tool

The Quick Look Tool accounted for tanker availability on a day-by-day basis, but it did not consider tankers located at separate bases. Capehart's research extended the approach, increasing the tanker's capability to multiple origins. The scheduling precision in the TAP Tool is in terms of minutes no matter how complex the deployment is.

Due to the complexity of deploying large numbers of receiver aircraft, and scheduling them in terms of minutes, a heuristic is required to obtain solutions in a reasonable length of time. Capehart used a tabu search (TS) method to solve this tanker assignment problem.

In order to illustrate the tanker aircraft's route we should know what kind of receiver aircraft needs refueling, because a tanker aircraft's route differs according to the type of receiver aircraft. For a fighter aircraft, the tanker aircraft reaches the refueling point (RP), refuels the receiver aircraft, escorts them to the next refueling point or the final destination of the receiver aircraft, and returns to its original take-off base. An example scenario is created to illustrate the tanker aircraft's route is shown in Figure 1. The mission consists of sending an F-16 from Barksdale AFB (KBAD) to the Port of Astoria (KAST).

If the receiver type classifies as heavy, the tanker aircraft's route differs slightly. In this situation the tanker aircraft just refuels the heavy and returns to its base of origin. This is illustrated when sending one B-52 from Elmendorf AFB (PAED) to King Khalid AFB (OEKK), which is in Saudi Arabia. This is shown in Figure 2.

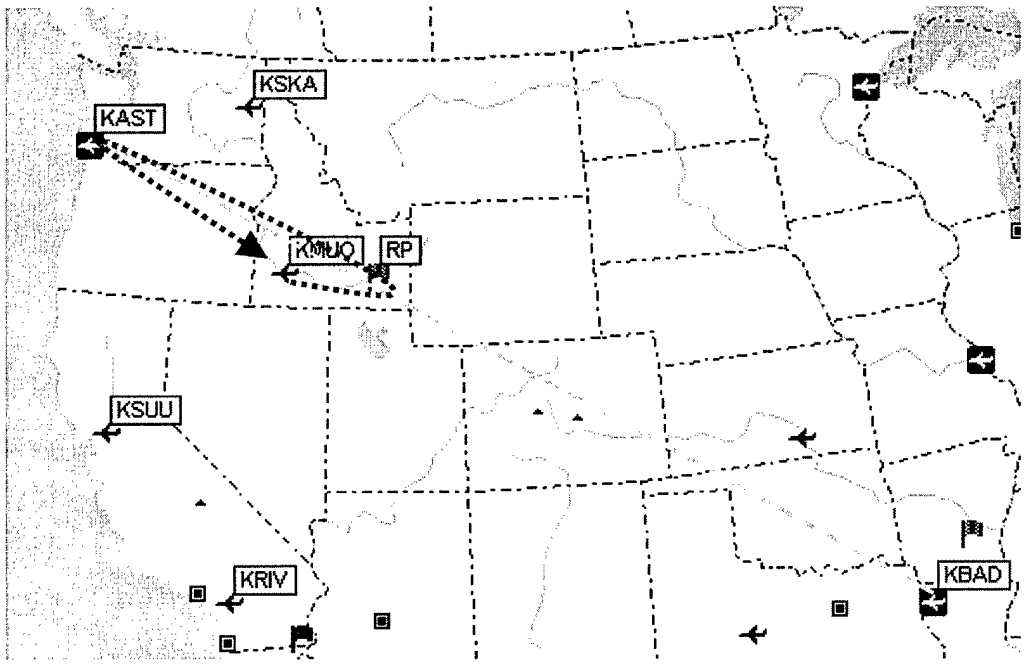

Figure 1: Tanker Escort of an F-16 Fighter

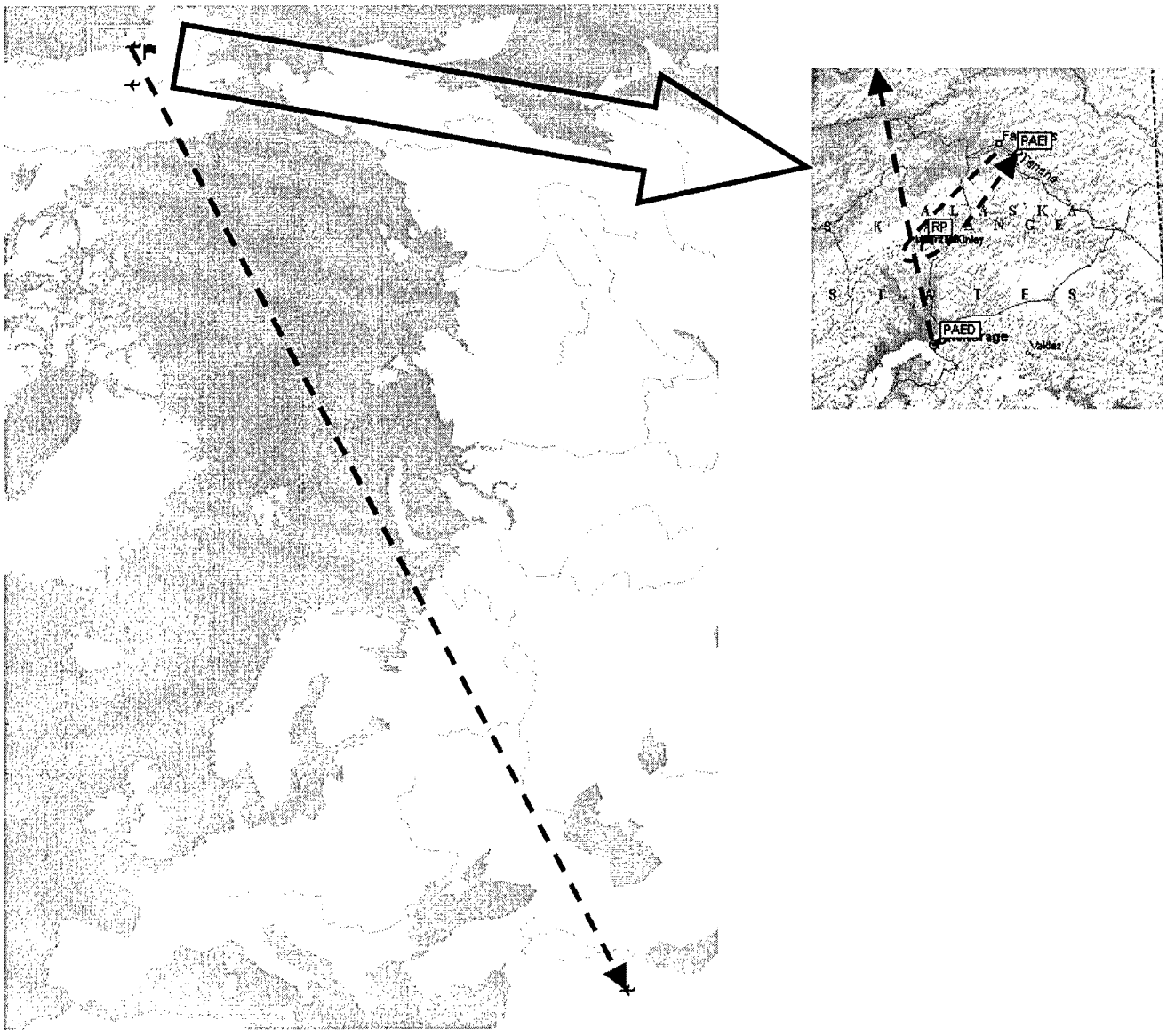

Figure 2: Tanker Escort of a B-52 Bomber

2.1.5 AMC's Efforts on TAP Tool

Once Capehart finished his efforts, AMC continued working on the TAP Tool. AMC divided the "Aircraft Performance" worksheet into two separate worksheets named "Receiver Aircraft Performance", and "Tanker Aircraft Performance". They created the ability to change the available tanker bases and number of tanker aircraft at them, by adding a table in the "Input" worksheet. These changes help the user easily make changes to the number of available tanker aircraft and tanker bases. There are no important changes in the running of the original TAP Tool.

2.2 SCHEDULING THEORY

Scheduling concerns the allocation of limited resources to tasks over time. It is a decision making process that has as a goal the optimization of one or more objectives (Pinedo, 1995). Scheduling impacts almost every military operation. Flight scheduling is the most common for Air Force operations.

Resources and tasks may take many forms. Resources might be machines in a workshop, tanker aircraft on a refueling mission, runways at an airfield, and so on. The tasks may be operations on an assembly line, take-offs and landings at an airport, or refuelings in the mission. Each task may have a different priority level, earliest possible starting time, and due date. Objectives include the minimization of the completion time of the last job, minimization of maximum tardiness, or minimization of the total number of late tasks.

The basic problems in scheduling include single machine or parallel machine problems. In real life single machine environments are rare, but analysis of single machine models led to heuristics for more complicated machine environments (Pinedo, 1995).

2.2.1 Parallel Machine Models

A machine can be thought of as a finite resource required for completing a task, such as a cashier in a checkout line. Parallel machines can be thought as cashiers in a checkout line, or different tail numbered tankers for refueling points. Minimizing the makespan is one of the most common objectives when working with parallel machines. Makespan is equivalent to the completion time of the last job to leave the system. A shorter makespan usually implies a high utilization of the machines. In this research we want all the receiver groups to arrive their destinations before their RDD. We applied a hard penalty even one of the receivers are tardy. We also want the latest receiver groups TOD to be less than the time limit of the deployment.

2.2.1.1 Precedence Constraints

Precedence constraints may appear in a single machine or in parallel machine environments. These require that one or more jobs must complete before another job is allowed to start its processing.

Precedence constraints define timing requirements between activity pairs within projects. Predecessor activity must end before its successor activity may start. During

refueling the same tail numbered tanker cannot meet the fuel requirements of two different missions at the same time.

2.3 HEURISTIC APPROACHES

Heuristic methods can be considered solving a mathematical problem by an intuitive approach (Silver et al., 1980). Heuristic algorithms provide near-optimal solutions to difficult problems in less time when compared with direct, analytic methods. A local search heuristic starts from an initial solution and tries to obtain a better one in an iterative way by searching the neighborhood of the current solution. A fast near-optimal solution to exact problems makes more sense than a time consuming exact solution to an inexact problem (Zanakis and Evans, 1981). Previous research on the tanker assignment problem applied tabu search. This research uses reactive tabu search, an extension of TS.

2.3.1 Tabu Search

Introduced by Glover (1986), TS is an iterative procedure for solving large combinatorial optimization problems. TS has proven to be extremely successful in solving to optimality or near-optimality a variety of classical and practical problems (Glover, 1990). Some basic concepts used in TS are a move, a neighborhood, candidate lists, the tabu list, intensification and diversification. Specific definitions for these terms change from problem to problem.

Two key aspects characterize many applications of TS. One of them is TS is used to complement local, and neighborhood search. The other one is the main modifications to local search are obtained through the prohibition of selected moves available at the

current point. Local search is effective if the neighborhood is appropriate to the problem structure, but its effectiveness stops as soon as the first local optimal solution is encountered and no improving moves are available. TS continues the search beyond the first local optimal solution without wasting the work already completed, as is the case if a new local search is initiated from a different starting point. TS enforces appropriate amounts of diversification to prevent the search trajectory from remaining confined near a given local optimal solution (Ben-Daya, Al-Fawzan, 1998).

In some cases, the solution phase of a specific problem may get trapped at a local optimum. TS is designed to escape the trap of local optimality (Glover, 1990). An algorithm is needed that recognizes that the procedure is at a local optimum and drives the search away from the local optimum. In other words, TS should have its own weapons to get away from local optimums. Examining how recently or how frequently solution attributes are encountered triggers those weapons. TS uses flexible memory structures to explore the solution space more thoroughly than by rigid memory systems or memoryless systems (Glover, 1990).

The tabu search algorithm "moves" through the solution space in search of good solutions. When the algorithm leaves one solution and visits another, this is called a "move". Specifically, the search examines a neighborhood of a given solution. This examination does not depend on randomization (Glover, 1986). Move definition is very problem specific. Examples of moves are changing the value assigned to a variable, adding or deleting an element from a set, or interchanging the position of two jobs on a machine, and so on (Glover, 1990). The set of potential moves from the current solution that the TS algorithm can operate on is called a move neighborhood.

Move neighborhood size depends on the problem type and the move definition. For illustrative purposes, suppose we have a deployment, which has only one refueling point, and again suppose that only tanker Air Force Bases (AFBs) A, B, and C can support that refueling point. We define a move as replacing a tanker assigned to the refueling point with another tanker within range that can also support that refueling point. For this specific problem the neighborhood structure is composed of A, B, and C AFBs. If we choose AFB A as a supporting base for the refueling point, the possible moves include: swapping AFB A with AFB B, or swapping AFB A with AFB C. As the number of supporting bases and the number of refueling points increase, the size of the neighborhood grows dramatically.

Candidate list strategies decrease neighborhood search time and increase the number of iterations the algorithm performs per unit time. A candidate list is a subset of the entire neighborhood. Determining the best admissible candidate is a critical step. It might be found randomly or based on some predetermined rules. The evaluation of a move can be based on the change produced in the objective function value (Glover, 1990). Glover advocates the use of rules to determine candidate lists, stating, "efficiency and quality can be greatly affected by using intelligent procedures for isolating effective candidate moves" (Glover and Laguna, 1997: 343).

Tabu search uses memory extensively for many different purposes, one of which is to prevent cycling with a "tabu list". The tabu list is closely related to the short-term memory component of TS, and forms the core of TS (Glover, 1990). A tabu list usually consists of a list of moves, or their attributes the search has recently encountered. The moves on the tabu list cannot be revisited for a specified number of iterations called the

tabu tenure. Once a move has been on the tabu list for a number of iterations equal to the tabu tenure, it is removed from the list and is again a valid move choice. The size of the list can either be fixed or variable (Ben-Daya, Al-Fawzan, 1998). Checking tabu status is the first step in screening for admissibility of moves (Glover, 1990). Tabu lists may force the tabu search algorithm to make an unimproving move if all the moves that improve the objective function value are on the tabu list.

An intensification strategy takes advantage of the idea that "good solutions at one level are often found close to good solutions at an adjacent level" (Glover and Laguna, 1997: 138). TS uses flexible memory for intensification. If the search encounters a good solution, it may be desirable to intensify the search within the local area in hopes of finding an optimum. Intensification strategies reinforce move combinations and solution features historically found good (Glover, 1990). An elite solution is a good solution found and saved for future exploration. It is a good idea to keep a rank ordered list of the best solutions found so far so that the algorithm knows what solutions to visit for intensification purposes. Once the intensification phase has been completed, TS may terminate or start a diversification phase.

There are times when the tabu search algorithm will get stuck in an unproductive region of the solution space. The algorithm needs something more than the tabu list to escape these bad regions of the solution space. Long-term memory operates primarily as a basis for diversifying the search (Glover, 1990 and Lokketangen and Glover 1998). If a search begins cycling within, and cannot escape a region, it needs some triggering event forcing the search to another region. Cycling is described returning to past solutions or revisiting past sequences of solutions. Cycling is unproductive so mechanisms in TS are

used to prevent it. First a procedure is needed to identify cycling, and as soon as it detects cycling, it forces the algorithm to diversify to other regions of the solution space. Two different number series are shown in Tables 3 and 4. Table 3's series is an example of a non-cycling while Table 4's series is an example of cycling.

Table 3: Example for a non-cycling Series

10	20	30	10	20	10	30

Table 4: Example for a Cycling Series

10	20	30	10	20	30	10

The numbers represented in Table 4 are cycling. The numbers 10, 20, and 30 are repeating themselves once again, making the cycle period 3 for this series.

In many applications, the short-term memory component itself produces high quality solutions. However, long-term memory might be needed to obtain better results for complex problems (Skorin-Kapov, 1989). The main difference between intensification and diversification is that during an intensification stage, the search focuses on examining neighbors of elite solutions, while the diversification stage encourages the search process to examine unvisited regions and to generate solutions that differ from those seen before (Glover, 1990). Intensification requires short-term memory, while diversification uses long-term memory to accomplish its purpose.

2.3.2 RTS

RTS was developed by Battiti and Tecchiolli (1994), and can be considered as a useful blend of intensification and diversification. In the RTS algorithm, by examining the solution quality, the tabu tenure is automatically adjusted to control the search. This allows the search to move between intensification and diversification (Battiti, 1996:61). The tabu tenure is learned automatically by reacting to the occurrence of cycles (Battiti, Tecchiolli, 1994).

Some tabu search implementations are based on the fact that cycles are avoided if the repetition of previously visited configurations is prohibited (Battiti, 1996 and Battiti and Tecchiolli, 1995). The reverse elimination method (REM) is an example of this. REM can be described as a kind of strict tabu search where the only local movements that become tabu are those that would lead to previously visited solutions. But this method may converge to a good solution very slowly if the suboptimal solution is surrounded by large basins of attractions. For this reason, the proper choice of list size is very important to the success of the algorithm. Actually, more robust schemes are based on a randomly varying list size, but this is problem specific and variations on the list size must be prescribed (Battiti, Tecchiolli, 1994 and Lokketangen and Glover 1998).

Some problems require a careful selection of parameters to find a good solution. This may be achieved either from a deep knowledge of the problem or by a simple trial and error process (Battiti, 1996). RTS changes tabu tenure according to the quality of the search. Solution quality is a function of the length of time a move remains tabu. Generally, tabu tenure starts out small and grows rapidly if cycling occurs. Tabu list size decreases gradually as cycling is diminished.

The use of RTS requires a mechanism to properly identify when the search has cycled. Hashing functions, as described by Glover and Laguna (1997:246), are widely used and considered a fast, efficient way to determine cycling in complicated problems. The overriding purpose of any hashing function is to enable a tabu search algorithm to detect whenever it transitions to a solution that has already been visited (Carlton and Barnes, 1996; Woodruff and Zemel, 1993). If a problem requires excessive memory space to store all configurations, one may resort to compression techniques (Battiti, Tecchiolli, 1994). For small problems, entire solutions can be compared (Battiti, 1996:67).

Increasing the list size is the first step when encountering previously visited solutions. But this may not be enough to force the search trajectory to another portion of the solution space. There must be another mechanism that counts the number of configurations that are repeated many times. If this number exceeds a predetermined threshold, a diversification step is triggered.

2.3.3 Candidate List Strategies

The simplest candidate list strategy is to examine the full neighborhood of available moves (Lokketangen and Glover, 1998). However, this can be an expensive process in terms of computation time. The purpose of introducing candidate lists is to decrease the neighborhood structure, so that reaching a better solution is improved by not spending time on all neighborhood evaluations. Candidate list strategies include: aspiration plus strategy, elite candidate list strategy, successive filter strategy, sequential fan candidate

list strategy, and bounded change candidate list strategy. The tool implemented in this research applies the elite candidate list strategy.

2.3.3.1 Elite Candidate List

The elite candidate list approach builds a list of good solutions by saving certain good solutions encountered during the tabu search. Periodically the current best move from the elite list is selected and tabu search is restarted. The elite candidate list process continues adding to the list and periodically restarting the search. This technique is motivated by the assumption that good solutions may be revisited and their neighborhoods, searched more thoroughly, may yield better solutions (Glover and Laguna, 1997:63).

2.4 CONCLUSION

Intelligent and aggressive use of candidate lists can speed up the search by forcing an evaluation of a subset of the neighborhood. TS was chosen as the method for this research. RTS is a dynamic procedure that changes the tabu tenure when required. Introducing RTS represents major progress for this research. The basic purpose is to further decrease the solution time without reducing the solution quality.

CHAPTER 3. METHODOLOGY

This chapter describes the verification and validation effort applied to the TAP Tool, and the specifics of the reactive tabu search used to find solutions to the aerial refueling missions during deployments. The first section describes the verification and validation effort done by correcting the coordinate problems, and modifying the objective function formulation. The second section concerns the implementation of reactive tabu search and elite candidate list strategy.

3.1 VERIFICATION & VALIDATION EFFORT

After Capehart (2000) composed the TAP Tool, AMC began to work on the tool. They added some coding to the original TAP Tool. Their effort did almost nothing to the execution of the original TAP Tool, although there were some problems with the TAP Tool. First of all, there was a coordinate problem for some deployments. The second basic problem was choosing the wrong tanker bases for some deployments. Other minor problems had already been corrected. For example in some deployments, there was no answer at all. However, adding overseas tanker AFBs to the original composition of the tool solved this difficulty.

3.1.1 Coordinate Problem

In one of the deployments, we encountered a coordinate problem. When we sent one B-52 from Barksdale AFB (KBAD) to Andersen AFB (PGUA), the tool was choosing refueling point 1 when it should have selected refueling point number 2 as shown in Figure 3.

We identified an error in implementing the sign conversion for indicating direction. In the original model the sign was positive (+) making the coordinate west, and it should have been negative (-) making the coordinate east. After the correction, the RP moved from 1 to 2, which is the correct coordinate.

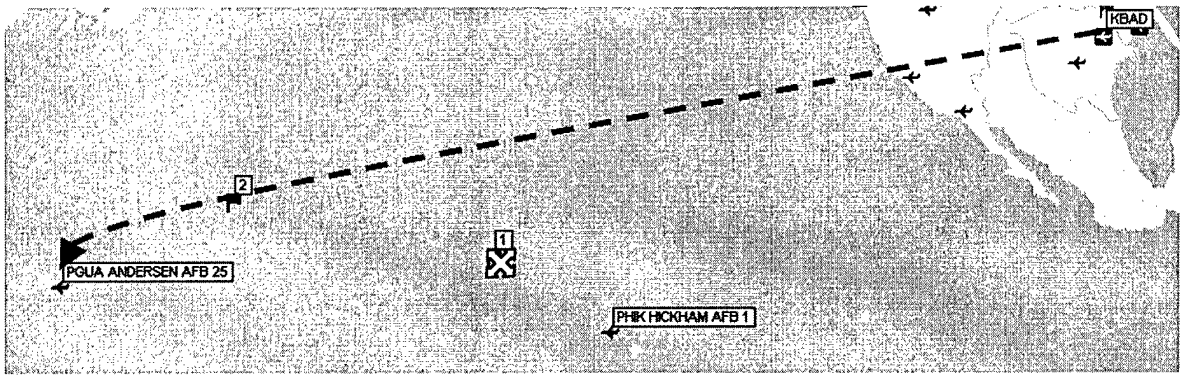

Figure 3: Illustration of coordinate change from point 1 to point 2.

3.1.2 Modification of the Objective Function Evaluation

Before discussing what was done to the objective function, it is necessary to present the original objective function.

3.1.2.1 Old Mission Evaluation

There are many specified goals for this problem. One goal is to have the receiver groups arrive at their destinations on time. This is represented by a hard constraint, resulting in an infeasible mission plan if any receiver groups are scheduled to arrive past their required delivery date (RDD). The second goal is to minimize the total distance traveled by the tanker aircraft. Another possible goal is to use the fewest tankers possible. The original objective function evaluated a mission plan using a single value incorporating each goal, and the objective was to minimize mission evaluation:

Mission Evaluation = Distance Penalty – Reuse Bonus – Early Bonus + (Sync Penalty + Conflict Penalty + Tardiness Penalty + Early Penalty + Negative TOD Penalty) + Tanker Index

The TS search engine primarily uses the following three values during the search process.

Distance Penalty: For each decision variable, the distance penalty is the sum of the distances traveled by all tankers in the current mission. The solutions with smaller distance penalties are more attractive.

Reuse Bonus: Each time a tanker is reused in a deployment, this value is increased.

Early Bonus: Although we are most concerned with having each receiver group arrive by its RDD, it is valuable to have them arrive earlier. Thus, this value represents the number of hours a receiver group arrives at its destination before its RDD.

The next set of values encourages feasibility throughout the search process.

Sync Penalty: If a receiver group requires more than one tanker at a refueling point along its path, these tankers must arrive at that location at the same time. A heavy

penalty is applied when the tankers assigned to this multi-tanker refueling point are not scheduled to arrive at the same time.

Conflict Penalty: In order to ensure that a tanker does not refuel two receiver groups at the same time, a conflict penalty is imposed. Each tanker is given a 3-hour turnaround period between missions. If any of the tankers in the solution break this 3-hour separation between assignments, a conflict occurs. When this happens, a large conflict penalty is added to the objective function.

To expedite the search process, we attempt to generate an initial feasible mission plan. This plan is generated using a greedy construction heuristic. For each decision variable (DV), representing a refueling point, we assign the closest tanker that has not yet been assigned to another DV. If the maximum number of tankers at the closest base is reached, we repeat the tanker assignments at this base. This is when a tanker conflict may occur, in which the same tanker is scheduled for more than one refueling point at the same time. Increasing the number of tankers at each base eliminates this conflict, but this also increases the computation time required during the search process. Therefore, we allow the initial solution to be infeasible for some deployment problems. When we begin with an infeasible solution, the first goal is to reach a feasible solution. To accomplish this, the large negative conflict penalty in the objective function encourages the TS to generate moves to decrease the number of conflicts.

Early Penalty: The deployment is scheduled to begin at time zero. Thus, a heavy penalty is applied when a receiver group's time of departure (TOD) is before the beginning of the planning horizon.

Tardiness Penalty: In order to ensure that each receiver group arrives before its RDD, a large penalty is applied to the objective function when a receiver group's TOD is tardy.

Negative TOD Penalty: In the case when a receiver groups TOD is before zero, which might happen when starting with an infeasible solution, this value directs the search process into the feasible region. This term applies large negative values to the missions that correct this problem, thus making those missions attractive choices for the next mission plan.

Tanker Index: Tankers are indexed at each base. Indexing encourages using the lower indexed tankers. We use this value for cosmetic reason only, since all tankers are of the same type (Capehart, 2000).

3.1.2.2 New Mission Evaluation

Before explaining what is being added to the model it is important to understand why the changes were made. To illustrate, we examine some sample deployments. The first deployment sends one F-16 from Barksdale AFB (KBAD) to the Port of Astoria (KAST). This deployment is shown in Figure 4. It is apparent that Mountain Home AFB (KMUO) should support the refueling point since it is the nearest AFB to the RP (140 NM). However, the TAP Tool chose Fairchild AFB (KSKA) even though the distance (357 NM) was much greater than Mountain Home AFB's.

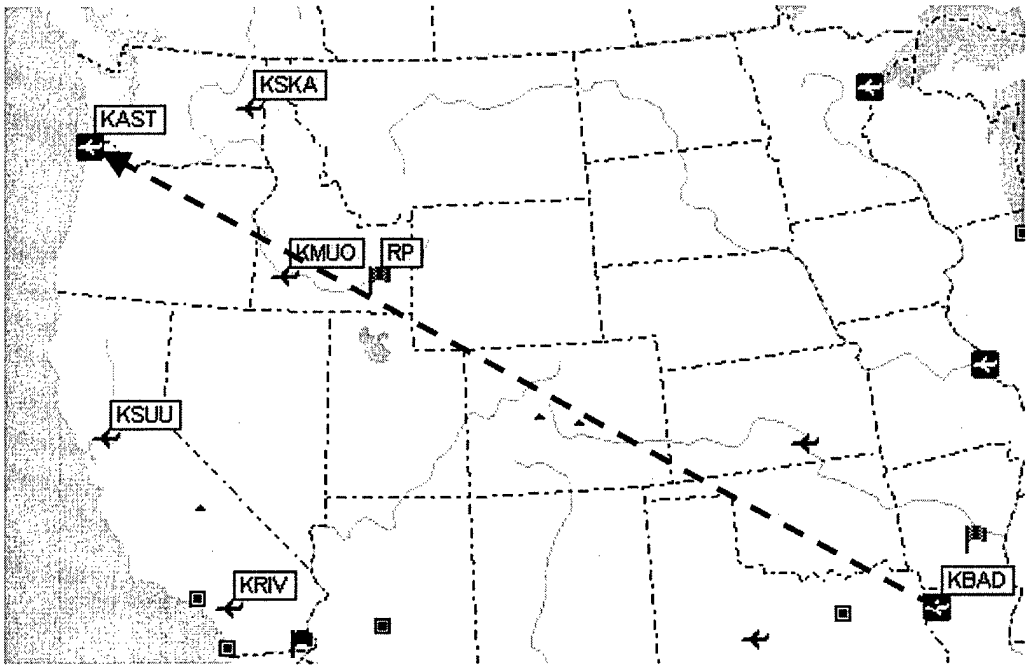

Figure 4: Sending one F-16 from KBAD to KAST

The second deployment sends one F-16 from Barksdale AFB (KBAD) to Hickham AFB (PHIK). This deployment is shown in Figure 5. The TAP Tool chose Fairchild AFB (KSKA) for the first RP, and Hickham AFB (PHIK) for the second RP. However, it is obvious that March AFB (KRIV) should be chosen for the first RP.

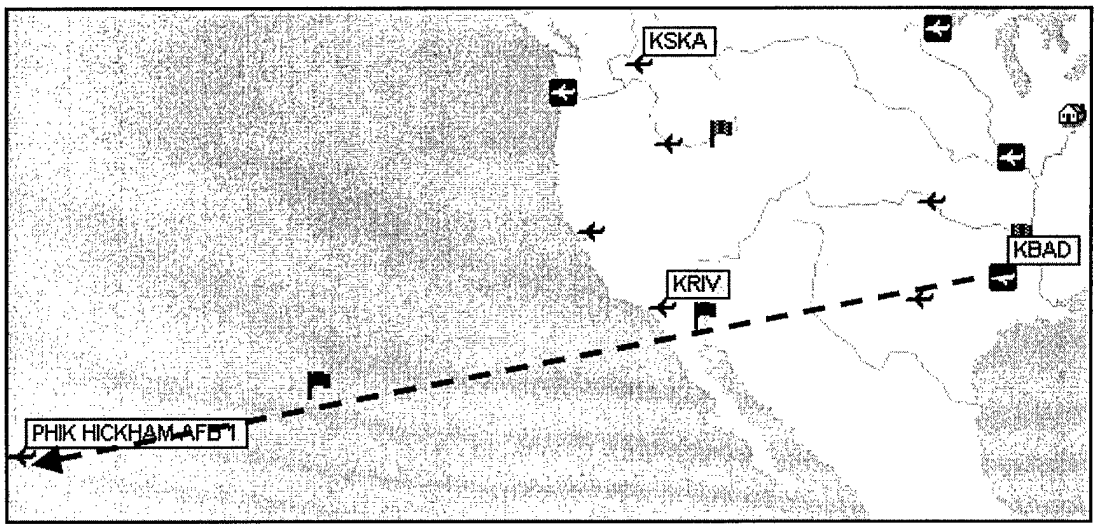

Figure 5: Sending one F-16 from KBAD to PHIK

Since the tool was not choosing the correct AFBs for these simple deployments, the objective function was modified. The term added to the objective function considers distance for a tanker from base to the RP. The tanker index is excluded from the new objective function. After the changes, the new minimization objective function is:

Mission Evaluation = Distance Penalty – Reuse Bonus – Early Bonus + (Sync Penalty + Conflict Penalty + Tardiness Penalty + Early Penalty + Negative TOD Penalty) + distance from base to RP

Distance from base to RP: This is the distance for a tanker to travel to its assigned refueling point. Closer tanker bases are more attractive. If the tanker base is close to the RP it receives a high value.

3.2 TABU SEARCH IMPLEMENTATION

In order to use the TAP, we first determine how to represent the solution. There must be a tanker aircraft assigned to each refueling point, so the solution consists of those tankers assigned to the refueling points plus the time at which the tankers take-off from their base of origin. In addition, the tanker take-off times directly determine the times at which the receiver groups must leave their base of origin. Thus, the receiver group take-off times are also part of the solution.

We first generate the list of solution attributes, which are modified to form different solutions. Let **RG1** be the first receiver group request. We first determine if this receiver group requires refueling. If **RG1** must be refueled during its deployment, we calculate the location for the required refueling(s). Let $\mathbf{RG1_1}$ be a decision variable (DV) representing the first refueling point for receiver group one. Next, we check each of the bases with tanker aircraft to determine if there exists a tanker capable of satisfying **RG1's** fuel requirements. If **RG1** consists of aircraft classified as "heavy," the amount of fuel that group requires is the amount of fuel needed to successfully continue the flight to its destination or to the next refueling point. On the other hand, if **RG1** contains aircraft classified as "fighter," then this group must have a tanker escort between refueling points. Although "fighter" groups only require tanker escort over open water, we assume for this model that they require tanker escort between each refueling point and their final leg of the mission. For this type of receiver group, a tanker must be capable of traveling to the refueling point, fulfill the receiver group's fuel requirements, escort the group to the next refueling point, and finally return to its base of origin. The tanker routes for sending one F-16 from KBAD to PHIK are shown in Figure 6. For each tanker capable of satisfying the requirements at $\mathbf{RG1_1}$, we generate a number of alternatives for the decision variable $\mathbf{RG1_1}$. A separate alternative is generated for each tanker and for discrete take-off times, in one-hour increments. We generate this list of DV alternate values for all bases with tankers capable of satisfying the fuel requirements for $\mathbf{RG1_1}$.

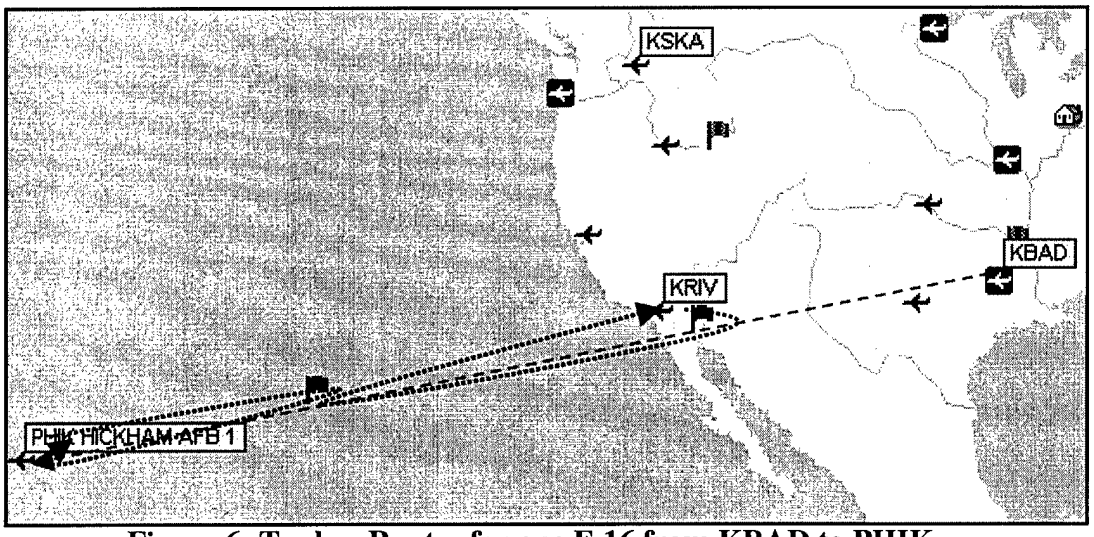

Figure 6: Tanker Routes for one F-16 from KBAD to PHIK

Suppose that there exists no tanker with enough fuel to satisfy $\mathbf{RG1_1}$**'s** requirements. In this case, we attempt to assign two tankers to this refueling point, with each satisfying half of the off-load required of **RG1**. This process continues until the appropriate number of tankers are assigned to $\mathbf{RG1_1}$. For each tanker assigned to $\mathbf{RG1_1}$, a new DV is generated, with the same list of alternative choices for that DV. For example, if two tankers are needed for $\mathbf{RG1_1}$, then two DVs are generated, each having the same set of alternate tankers, since both DVs have the same fuel off-load.

If **RG1** requires more than one refueling point, we apply this same technique to each of the these extra refueling points, with each of them representing another DV; however, each tanker has only one take-off time, as opposed to the different distinct take-off times for the tankers assigned to a first refueling point. This take-off time is a continuous value that is based on the take-off time for the tanker assigned to the receiver group's first refueling point.

This method continues for each receiver group until all refueling points and numbers of tankers assigned to each are determined. At this point, the list of DV alternatives represents all options for assigning tankers to receiver groups for the final mission plan. Which of these alternatives to choose is now the focus of the search engine.

3.3 REACTIVE TABU SEARCH IMPLEMENTATION

The user has an ability to select the initial solution by entering Y/N to the Greedy option on the "Input" worksheet. If the user sets this value to "Y", then the greedy heuristic forms the initial solution. Otherwise, the initial solution is constructed based on tanker availability, not accounting for the tanker distances. Producing the initial solution randomly causes many conflict problems. After constructing the initial solution using either a greedy approach or randomly generating a solution, the reactive tabu search finds all the elite solutions by simply checking the move evaluations of every support base for every RP. Reactive tabu search changes one support base, which is in the elite candidate list, at a time and evaluates the corresponding solution. Each solution contains the time of refueling, place where the refueling will take place, take-off times for tankers, and tanker bases that support the mission. All the variables are kept in an array, named "DV", for future examinations. This DV array also contains the move evaluations for that solution.

If the selected DV is already in the solution, then we set the penalty to Big-M (1,000,000), so that it is not chosen for the current move. Each move is evaluated by penalizing receivers if they take off too early, for late receivers, for having the same

tanker aircraft take off sooner than 10 hours apart, for not having multiple first refueling points at the same time, and rewarding if the number of conflicts has decreased.

After forming DV's for each RP, every changing point of tanker base names is indexed by "Rpind" variable to allow easy access for future evaluations. All the decision variable evaluations are saved to an array named "objectivefunc", and corresponding support bases are saved to an array named "supportbases"

The number of support bases for each RP are counted, and stored to an array named "numb". Furthermore, the minimum objective function evaluation for each RP is found and stored in a variable "minforthatRP".

Each mission is evaluated in a very aggressive manner. The minimum values for each RP are found and used to find the initial solution and this solution is stored. In order to maintain the best solution found so far, a variable called "bestsofar" is created. This variable always keeps track of the best solution found to date for the search process. This solution is always saved in the "bestSol" array.

Now begins the process of moving between solutions. First of all, we should begin with an explanation of the move used in this search. A move is defined as removing an existing tanker assigned to a refueling point and replacing it with another tanker within range to satisfy the needs of the receiver group at that refueling point.

As soon as the algorithm finds the minimum values for each RP, it indexes the minimum RP values in order to use them in future swap operations. Moreover, the reactive tabu search algorithm also finds the minimum among these minimum values obtained for each RP, and uses this as a beginning move. Once the algorithm detects this

value, it marks the base that includes this minimum value and stores it to the variable "Bul".

Since we found our starting move, we are ready to introduce the neighborhood structure that is being used in this search. The RP associated with the "Bul" variable is supported by some number of tanker bases. This value can never be more than the value of the variable "numtbases" which is entered by the user on the "INPUT" worksheet. This number is subject to change. Basically, it is the number of tanker bases that can be used in a contingency.

The number of tanker bases that supports the RP forms the neighborhood structure. Since we have our swap element, which is the minimum value for that RP, we have to find which support base to swap with. The answer to this question forms the basis for our reactive tabu search procedure. The base having the second minimum objective function evaluation is chosen as the DV to swap with, if this DV is not on the tabu list for that iteration.

As soon as the move is made, the swap element is put in a tabu list, the objective function evaluation is recalculated, and the support bases for that mission are saved.

Now begins the basic tabu search principal of "remembering the past". In this search, both short and long-term memory components are used in order to trigger a mechanism in the case of finding the same solution as the previous move, or in the case of long term cycling.

At the end of every iteration, the objective function value found on the previous iteration is compared with the current objection function value. RTS keeps track of recently visited solutions, in order to determine if the search is stuck in a local minimum.

If the search finds the same results for consecutive iterations, it diversifies the search by randomly examining another portion of the solution space and increases the tabu tenure by one.

More robust schemes are based on a randomly varying list size, although one must prescribe suitable limits for its variation (Battiti and Tecchiolli, 1994:126). The tabu tenure was initialized to one. When the search encountered repetitions, it increased the tenure by one. The main reason for doing so is completely problem specific.

The neighborhood structure in this problem is composed of tanker bases that support a specific refueling point. When you look closely at a neighborhood, it is obvious that in some scenarios there are only one or two tanker bases that support a specific refueling point. To account for this situation, it is logical to increase tabu tenure by only one. Another reason for this is that the neighborhood structure can never exceed twelve bases for this problem. Even though the number twelve can be increased; the number of tanker bases supporting any deployment will be relatively small. Therefore, it is logical to increase tabu tenure by only one. This feature allows the tool to explore the promising regions more closely and makes the search more aggressive. To account for these small numbers, it is logical to increase tabu tenure gradually and to decrease tabu tenure as soon as repetitions and/or cycles are diminished.

The second memory component is long-term memory. The cycle detection algorithm in the tool looks for a sign of cycling. At the end of every iteration, the algorithm looks at the results of all iterations regressively. If it detects the same result in one of the previous iterations, it tries to determine if it is a cycle. If it is a cycle, this algorithm diversifies the search to another region of solution space by randomly

examining another portion of the solution space. "... Cycle avoidance is not an ultimate goal of the search process... the broader objective is to continue to stimulate the discovery of new high quality solutions" (Battiti and Tecchiolli, 1994:127).

Fortunately, we have the ability to illustrate all the solution space. "If a problem requires an extensive memory space to store all configurations, one may resort to compression techniques, i.e. hashing" (Battiti and Tecchiolli, 1994:126). We do not need further storage requirements for hash tables. All the solutions are kept in an array. This helps us to compare all the solutions easily. The likelihood of collision described as meeting the exact same solution is very low; for this reason, we did not use hash tables.

The TAP tool uses three phases to arrive at an output. (See Figure 7). The three phases consist of DV alternative generation, initial solution generation, and RTS. RTS flowchart is shown in Figure 8.

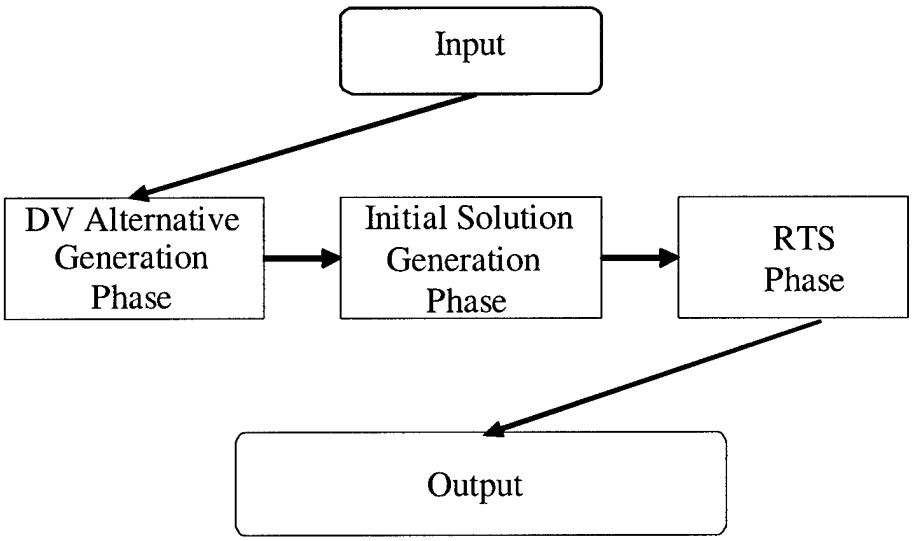

Figure 7: Revised TAP Tool Flowchart

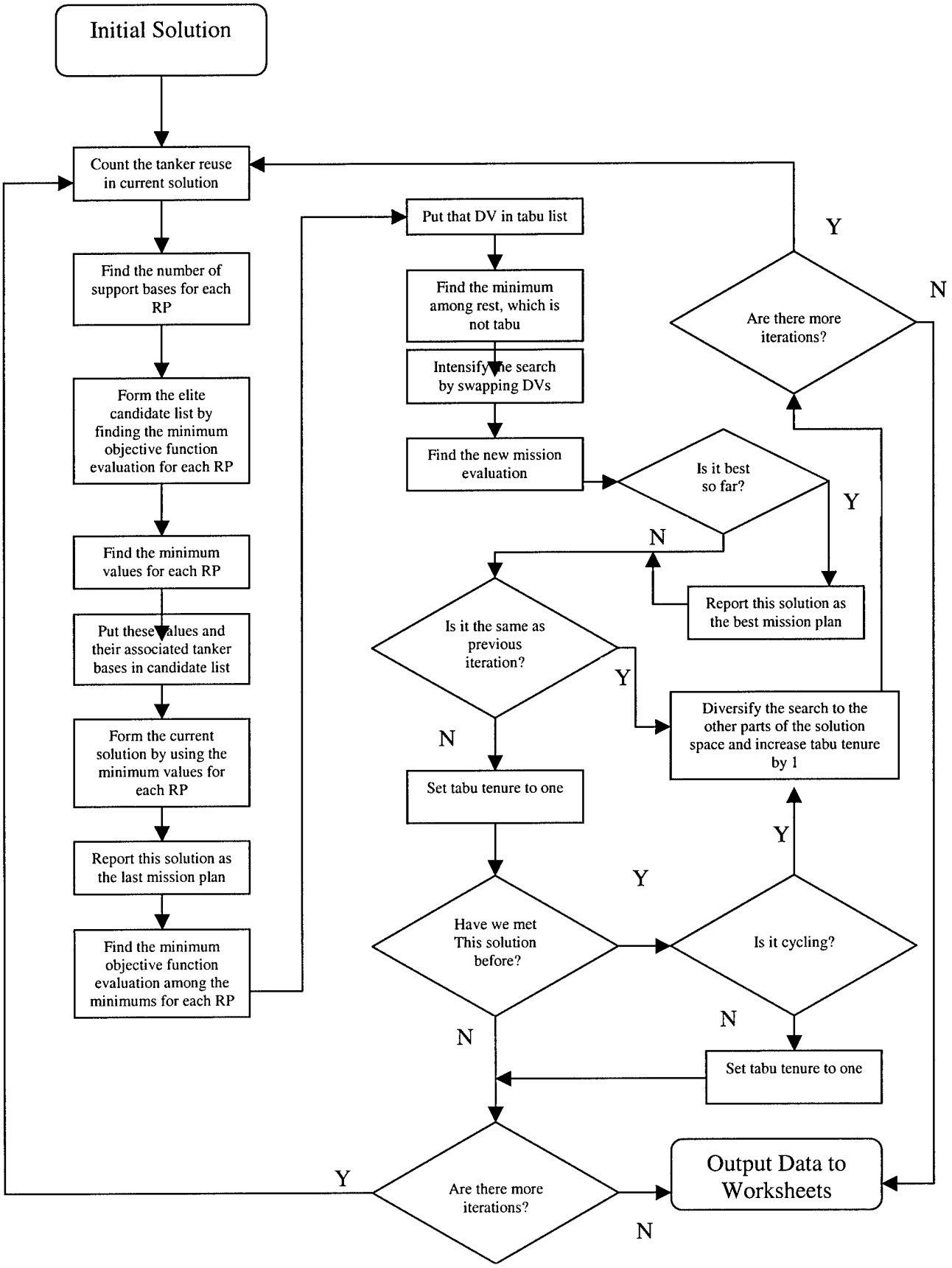

Figure 8: RTS Flowchart

3.3.1 Memory Usage

Explicit memory is used in this research. Explicit memory records complete solutions visited during the search. All the solutions are kept in an array. These solutions are used for measuring cycling in the short-term. Objective function values of corresponding solutions are kept in another array. These values are used to check for long term cycling.

The attributive memory component of tabu search is not used in this research. Attributive memory records information about solution attributes that change in moving from one solution to another.

3.4 ELITE CANDIDATE LIST STRATEGY

If a move neighborhood is too large to efficiently evaluate, a candidate list strategy intelligently isolates "effective candidate moves, rather than trying to evaluate every possible move in a current neighborhood of alternatives" (Glover and Laguna, 1997). Even though the neighborhood of alternatives are not large for this problem, it is logical to use effective candidate moves in order to speed up neighborhood evaluation and allow the algorithm to explore more areas of the solution space in less time. This idea is illustrated using a sample scenario. The sample mission consists of sending one B-52 from Elmendorf AFB (PAED) to King Khalid (OEKK), and one F-16 from Barksdale (KBAD) to Andersen AFB (PGUA). The bases activated for this scenario are shown in

Table 5. The B-52 mission requires only one refueling. While the F-16 mission requires five refuelings. The bases supporting these RPs and their objective function evaluations are shown in Tables 6 and 7, respectively.

Table 5: Bases Activated for Neighborhood Scenario

Bases	# of Tankers
KWRB	25
PAEI	25
KMCF	25
KSKA	25
PAED	1
KIAB	25
KMUO	25
PGUA	25
PHIK	1
KDYS	10
KRIV	10
KSUU	10

Table 6: Bases supporting RPs for Example Deployment

B-52	F-16				
1st RP	1st RP	2nd RP	3rd RP	4th RP	5th RP
KMUO	KMCF	KRIV	PGUA	PGUA	PGUA
KRIV	KMUO	KSKA	PHIK		
KSKA	KRIV	KSUU			
KSUU	KSKA	PAED			
KWRB	KSUU	PHIK			
PAED	KWRB				
PAEI	PAED				
PHIK	PAEI				
	PHIK				

Table 7: Objective Function Evaluations for Example Deployment

B-52	F-16				
1st RP	1st RP	2nd RP	3rd RP	4th RP	5th RP
6992	6761	3521	5198	9073	9259
6357	4826	2858	2280		
6179	5306	9064			
6339	5314	1605			
6155	4551	2453			
1151	6175				
1266	6050				
6485	6337				
	6733				

Trying to allocate tankers to refueling points for this deployment results in 720 different solutions. It is obvious that some of the solutions are not attractive. In order to no visit unattractive solutions, we introduced the elite candidate list strategy, and thus decreased the number of possible solutions. The bases supporting these RPs and their objective function evaluations after introducing the elite candidate list strategy are shown in Tables 8 and 9, respectively.

Table 8: Bases supporting RPs after Elite Candidate List

B-52	F-16				
1st RP	1st RP	2nd RP	3rd RP	4th RP	5th RP
KWRB	KMUO	KSKA	PGUA	PGUA	PGUA
PAED	KRIV	PAED	PHIK		
PAEI	KSUU	PHIK			

Table 9: Objective Function Evaluations after Elite Candidate List

B-52	F-16				
1st RP	1st RP	2nd RP	3rd RP	4th RP	5th RP
6155	4826	2858	5198	9073	9259
1151	5306	1605	2280		
1266	4551	2453			

The number of possible solutions decreased from 720 to 54 (3*3*3*2*1*1). The number of possible solutions and their objective function evaluations are represented in Tables 10 and 11 for all neighborhoods and for the elite candidate list neighborhood, respectively.

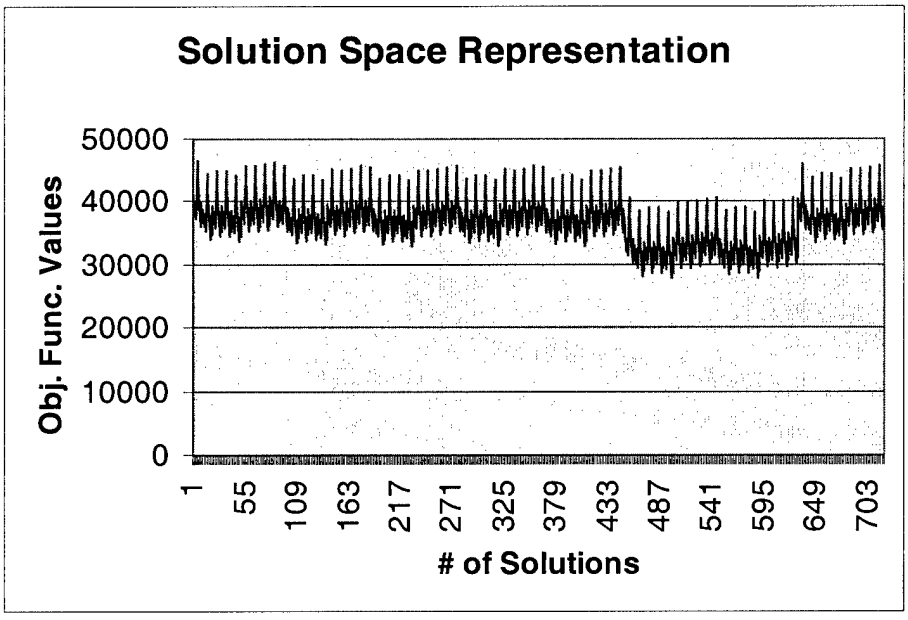

Figure 9: Number of Solutions for all neighborhoods

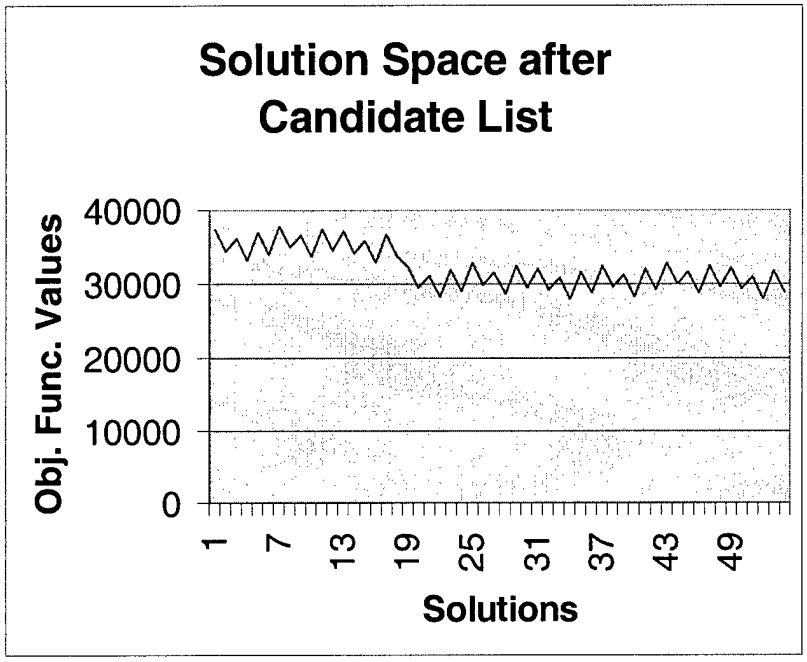

Figure 10: Number of Solutions after Forming Candidate List

When we increase the size of the problem by either increasing the number of RPs, increasing the number of missions, or increasing the number of support bases, it is obvious that looking at all the possible moves dramatically increases the solution time.

In chapter 4, we test our model using a large deployment. Results and conclusions are presented.

CHAPTER 4. RESULTS AND CONCLUSIONS

In this chapter we tested our model. The test deployment consists of sending 11 receiver groups from the continental US to the Southeast Asia.

4.1 SOUTHEAST ASIA DEPLOYMENT

This deployment involves receiver groups departing the U.S. and arriving in Southeast Asia. Table 12 provides a list of the 11 receiver groups shown in Figure 9.

The tanker bases activated for this deployment include McConnell, Mountain Home, Grand Forks, Fairchild, Kadena, and Eielson, with 15 KC-135 tankers located at each base. Tanker bases activated for this deployment are shown in Table 13. Tankers located at these bases are capable of satisfying all the receiver groups' fuel requirements during the deployment.

Table 10: Receiver Groups for Southeast Asia Deployment

Mission	Receivers	# of Receivers	Origin	Destination	ALD	RDD
1	F117	2	Holloman	Osan	1	5
2	F15	6	Mt. Home	Osan	1	5
3	F15	6	Elmendorf	Osan	1	5
4	F16	6	Eielson	Osan	1	5
5	A/OA10	6	Eielson	Osan	1	5
6	B1	1	Mt. Home	Andersen	1	5
7	B1	1	Ellsworth	Andersen	1	5
8	B1	1	Dyess	Andersen	1	5
9	B52	1	Barksdale	Andersen	1	5
10	B52	1	Minot	Andersen	1	5
11	F117	2	Holloman	Osan	1	5

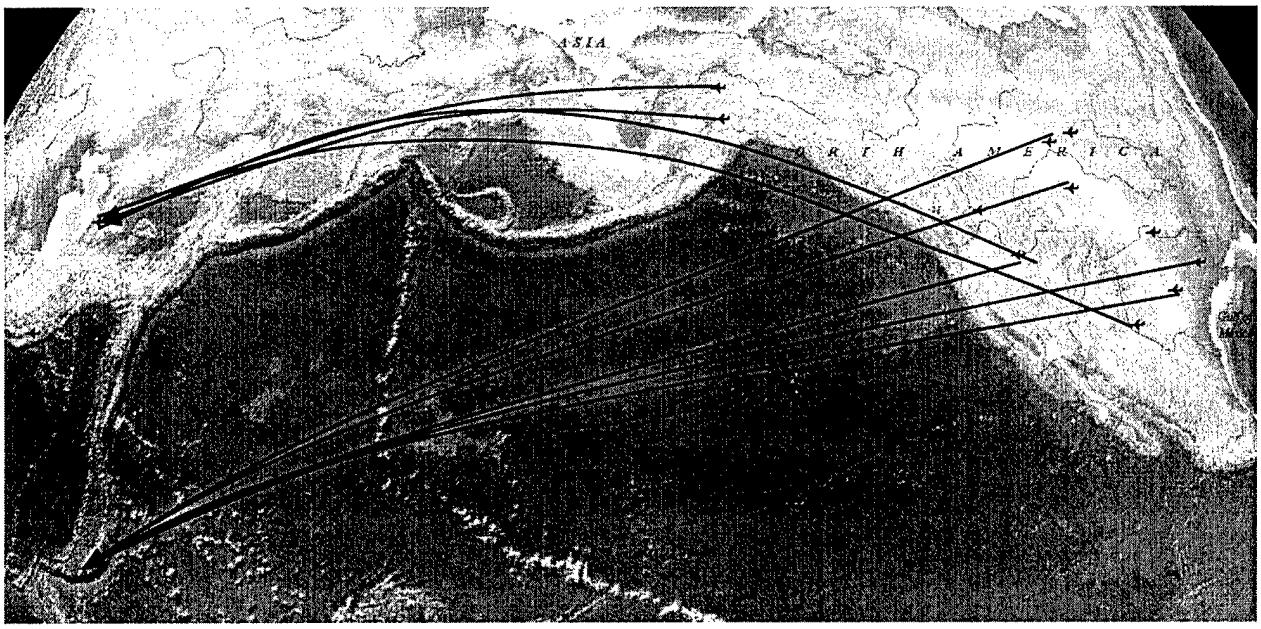

Figure 11: Southeast Asia Deployment

Table 11: Number of Aircraft on the bases used for Southeast Asia Deployment

TANKER BASE	ICAO CODE	NUMBER OF TANKERS
McCONNELL	KIAB	15
MOUNTAIN HOME	KMUO	15
GRAND FORKS	KRDR	15
FAIRCHILD	KSKA	15
KADENA	RODN	15
EIELSON	PAEI	15

The tool that AMC worked on does not run the Southeast Deployment. Therefore, we will compare the results with the TAP Tool.

The initial mission plan from this research is shown in Table 14, and this plan is constructed by using the greedy algorithm. Reactive tabu search sought better solutions by diversifying among the solution space through the iterations. The best mission plan from this research after 100 iteration is presented in Table 15 (All computer runs are made on a Pentium III 700Mhz, 128Mb RAM computer). As the iteration count increased the number of conflicts decreased and the schedule is stretched over time. The number of tanker aircraft used decreased from 27 to 14. The number of conflicts decreased from one to zero after 100 iterations. The initial mission plan is not feasible since there is a conflict between the two highlighted lines. Tanker tail #11 out of Kadena is scheduled to depart before it has time to accomplish its first assignment and take a 3-hour maintenance period.

Table 12: Initial Solution of Southeast Asia Deployment in RTS

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	KSKA	FAIRCHILD AFB	6	1
1	2	PAEI	EIELSON AFB	7.7493514	1
1	3	PAEI	EIELSON AFB	9.44891366	2
1	4	RODN	KADENA AB	10.378072	1
1	5	RODN	KADENA AB	15.4611866	2
2	1	PAEI	EIELSON AFB	6	3
2	1	PAEI	EIELSON AFB	6	4
2	3	RODN	KADENA AB	13.2606619	3
3	1	RODN	KADENA AB	1	4
3	2	RODN	KADENA AB	8.63640049	5
4	1	PAEI	EIELSON AFB	1	5
4	1	PAEI	EIELSON AFB	1	6
4	2	RODN	KADENA AB	2.96460085	6
5	1	RODN	KADENA AB	1	7
5	1	RODN	KADENA AB	1	8
5	2	RODN	KADENA AB	10.4902287	9
6	1	RODN	KADENA AB	11	10
7	1	RODN	KADENA AB	6	11
8	1	RODN	KADENA AB	6	12
9	1	PAEI	KADENA AB	11	7
9	1	PAEI	KADENA AB	11	8
10	1	PAEI	KADENA AB	6	9
11	1	KSKA	FAIRCHILD AFB	6	2
11	2	PAEI	EIELSON AFB	7.7493514	10
11	3	PAEI	EIELSON AFB	9.44891366	11
11	4	RODN	KADENA AB	10.378072	13
11	5	RODN	EIELSON AFB	15.4611866	14
11	5	RODN	KADENA AB	15.4611866	11
11	5	RODN	KADENA AB	15.4611866	11

Table 13: Solution of Southeast Asia Deployment after 100 iteration in RTS

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	PAEI	EIELSON AFB	2.92974044	1
1	2	PAEI	EIELSON AFB	7.7493514	2
1	3	PAEI	EIELSON AFB	9.44891365	3
1	4	RODN	KADENA AB	10.378072	1
1	5	RODN	KADENA AB	15.4611866	2
2	1	PAEI	EIELSON AFB	17.7493514	1
2	1	PAEI	EIELSON AFB	17.7493514	2
2	3	RODN	KADENA AB	25.0100133	1
3	1	RODN	KADENA AB	27.3736128	4
3	2	RODN	KADENA AB	35.0100133	1
4	1	PAEI	EIELSON AFB	29.4489137	2
4	1	PAEI	EIELSON AFB	29.4489137	1
4	2	RODN	KADENA AB	31.4135145	6
5	1	RODN	KADENA AB	50.378072	7
5	1	RODN	KADENA AB	50.378072	1
5	2	RODN	KADENA AB	59.8683007	9
6	1	RODN	KADENA AB	60.378072	1
7	1	RODN	KADENA AB	61.4135145	6
8	1	RODN	KADENA AB	70.378072	1
9	1	PAEI	EIELSON AFB	39.4489137	2
9	1	PAEI	EIELSON AFB	39.4489137	1
10	1	PAEI	EIELSON AFB	49.4489137	1
11	1	KSKA	FAIRCHILD AFB	57.6995623	1
11	2	PAEI	EIELSON AFB	59.4489137	1
11	3	PAEI	EIELSON AFB	61.1484759	2
11	4	RODN	KADENA AB	62.0776342	13
11	5	RODN	KADENA AB	67.1607489	14
11	5	RODN	KADENA AB	67.1607489	11

There is no conflict in the best mission plan output. This plan used 14 tankers, and was found in 13 minutes. The last receiver group to complete its trip has a TOA of 79 hours after the deployment begins. The timeline for the initial and best mission plan is

shown in Table 16. Scheduling of the initial and best missions on a time scale is shown in Figures 10 and 11, respectively.

Table 14: Timeline for Initial and Best Mission Plan in RTS

SORTIE ID	INITIAL MISS. PLAN TOA	TOD	BEST MISSION PLAN TOA	TOD
1	4.5	17.2	4.5	17.2
2	4.1	15.1	15.8	26.9
3	3.5	11	29.9	37.3
4	1.09	8.5	29.5	36.9
5	1.06	12.3	50.4	61.6
6	4.05	16.2	53.4	65.6
7	1.1	14.5	56.5	69.9
8	0.7	15.1	65.1	79.5
9	4.5	19.5	33.02	48.03
10	0.5	13.8	44.01	57.2
11	4.5	17.2	56.2	68.9

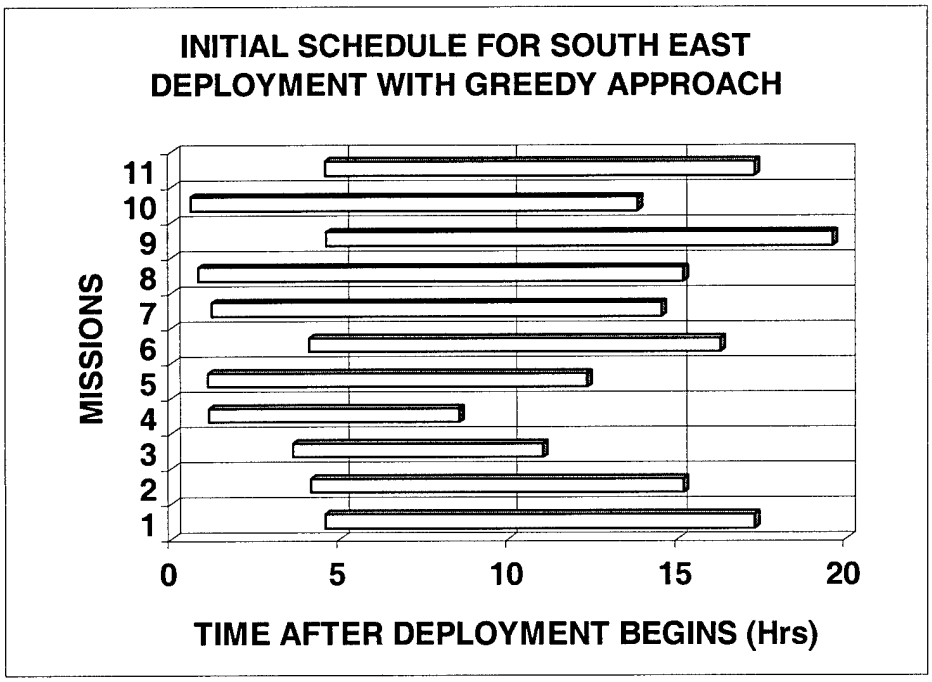

Figure 12: Initial Mission Schedule for South East Deployment in RTS

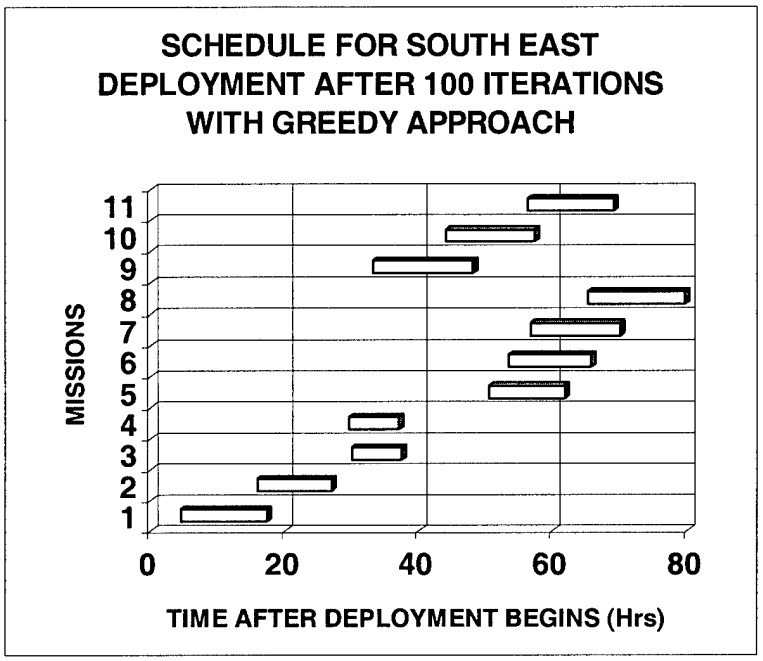

Figure 13: Best Mission Schedule for South East Deployment in RTS

When we examine the results of the TAP Tool, we see that even though the assignment of tanker bases to the refueling points has minor discrepancies both can be considered as correct allocations. For the TAP Tool, the latest receiver groups TOA is 75 hours after the deployment begins. Table 17 shows the resulting initial mission plan generated by the greedy method. Table 18 displays the TOD and TOA produced by the TAP Tool for each receiver group. The TOD and TOA values for tankers and receiver groups are represented in hours after the deployment begins.

Table 15: Initial Mission Plan for South East Deployment in TAP Tool

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	FAIRCHILD AFB	1	6.0
1	2	EIELSON AFB	1	7.7
1	3	EIELSON AFB	2	9.4
1	4	KADENA AB	1	10.4
1	5	KADENA AB	2	15.5
2	1	EIELSON AFB	3	6.0
2	1	EIELSON AFB	4	6.0
2	2	KADENA AB	3	5.7
2	2	KADENA AB	4	5.7
2	2	KADENA AB	5	5.7
2	3	KADENA AB	6	13.3
3	1	KADENA AB	7	1.0
3	1	KADENA AB	8	1.0
3	1	KADENA AB	9	1.0
3	2	KADENA AB	10	8.6
4	1	EIELSON AFB	5	1.0
4	1	EIELSON AFB	6	1.0
4	2	KADENA AB	11	3.0
5	1	KADENA AB	12	1.0
5	1	KADENA AB	13	1.0
5	2	KADENA AB	14	10.5
6	1	KADENA AB	15	11.0
7	1	KADENA AB	1	21.0
8	1	KADENA AB	1	31.0
9	1	EIELSON AFB	7	11.0
9	1	EIELSON AFB	8	11.0
10	1	EIELSON AFB	9	6.0
11	1	FAIRCHILD AFB	2	6.0
11	2	EIELSON AFB	10	7.7
11	3	EIELSON AFB	11	9.4
11	4	KADENA AB	12	10.4
11	5	KADENA AB	11	15.5

This initial mission plan uses of 28 tankers. Each receiver group arrives at its destination before their RDD. However, this mission plan is not feasible since there is a conflict between the two highlighted tankers. Tankers 11 and 12 out of Kadena are both scheduled to depart before they have time to accomplish their first assignments and take a 3-hour maintenance period.

Table 16: Initial TOD and TOD for South East Deployment in TAP Tool

Receiver Group #	TOD (hours)	TOA (hours)
1	4.5	17.3
2	4.1	15.2
3	3.6	11.0
4	1.1	8.5
5	1.1	12.3
6	4.1	16.2
7	16.2	29.5
8	25.8	40.2
9	4.6	19.6
10	0.6	13.8
11	4.5	17.3

After 100 iterations, the TAP Tool arrives at the best mission plan found which is presented in Table 19 and Table 20. This plan uses 13 tankers. The computation time for that run was approximately 20 minutes (all the runs were made on a Pentium II 350Mhz, 64Mb RAM computer).

Table 17: Final TOD and TOA for South East Deployment in TAP Tool

Receiver Group #	TOD (hours)	TOA (hours)
1	4.5	17.3
2	49.1	60.2
3	3.6	11.0
4	1.1	8.5
5	31.1	42.3
6	19.1	31.2
7	11.2	24.5
8	60.8	75.2
9	29.6	44.6
10	10.6	23.8
11	59.5	72.3

Table 18: Best Mission Plan Evaluation for South East Deployment in TAP Tool

RG #	Refueling Point #	Tanker Base	Tanker #	Tanker TOD (hours)
1	1	FAIRCHILD AFB	1	6.0
1	2	EIELSON AFB	1	7.7
1	3	EIELSON AFB	2	9.4
1	4	KADENA AB	1	10.4
1	5	KADENA AB	11	15.5
2	1	EIELSON AFB	5	51.0
2	1	EIELSON AFB	4	51.0
2	2	KADENA AB	3	50.7
2	2	KADENA AB	4	50.7
2	2	KADENA AB	13	50.7
2	3	KADENA AB	11	58.3
3	1	KADENA AB	3	1.0
3	1	KADENA AB	8	1.0
3	1	KADENA AB	13	1.0
3	2	KADENA AB	10	8.6
4	1	EIELSON AFB	5	1.0
4	1	EIELSON AFB	4	1.0
4	2	KADENA AB	11	3.0
5	1	KADENA AB	3	31.0
5	1	KADENA AB	13	31.0
5	2	KADENA AB	11	40.5
6	1	KADENA AB	11	26.0
7	1	KADENA AB	3	16.0
8	1	KADENA AB	13	66.0
9	1	EIELSON AFB	5	36.0
9	1	EIELSON AFB	4	36.0
10	1	EIELSON AFB	4	16.0
11	1	FAIRCHILD AFB	1	61.0
11	2	EIELSON AFB	10	62.7
11	3	EIELSON AFB	4	64.4
11	4	KADENA AB	11	65.4
11	5	KADENA AB	3	70.5

The concept of tabu tenure used in the TAP Tool is slightly different than that of reactive tabu search's. In the TAP Tool, the user can set different values of tabu tenure, each resulting in different solutions. In the previous research, a tabu tenure for 7 gave the best results. But tabu tenure in reactive tabu search displays a dynamic feature. It begins

at a value of one and changes as cycles are encountered. For that reason we do not compare the two tools on a tabu tenure basis. A general comparison of tools is presented in Table 21.

Table 19: General Comparison of TAP Tool and RTS

	TAP Tool	RTS
Time spent	20 min	13 min
Sol. Quality	Good	Good
Scheduling	Good	Good
Tankers used	13	14
Latest TOA	75.2	79.5

"Time spent" is described as the time that the computer spends for 100 iterations of the South East Deployment. The result of TAP Tool came from a Pentium II 350Mhz, 64Mb RAM computer. On the other hand, result of RTS came from a Pentium III 700Mhz, 128Mb RAM computer.

"Sol. Quality" is defined as allocating the correct tankers to the refueling points. In this deployment, many of the refueling points are supported by only one base. The tankers from Kadena AFB generally supported the refueling points over the Pacific Ocean. This is the main reason the results are so similar. For the first refueling points of fighter groups, the number of tanker bases supporting these refueling points increased. This situation increased the possibility of assigning different tankers for the refueling points. The total distance traveled by the tankers is 101.939Nm in RTS approach, but we do not have any information about this value for the TAP Tool results. For this reason both assignments are considered "good".

"Scheduling" is defined as stretching the deployment to its allowable time limits. For this deployment ALD is 1 and RDD is 5. We have four days to complete the deployment. Both tools finished the deployment in less than 96 hours (4 days). Latest receiver group's TOA for both tools are around 75-80 hours after the deployment begins. Since none of them arrived after the 96 hours of deployment limit, and both of the latest TOA are close, both tools are considered as "good" from the scheduling perspective.

"Tankers used" is defined as the total number of different tankers used. A low value in this field does not mean that the number of sorties flown by the tankers is low. The actual number of sorties flown is always the same. We reward reuse tankers, which results in a decreased number of tankers used.

"Latest TOA" is the latest time that a receiver group reaches to its destination. Our goal is to finish the deployment before its allowable time limit. We want this value to be less than the time limit of the deployment.

4.2 SOUTHEAST ASIA DEPLOYMENT WITHOUT GREEDY CONSTRUCTION

If the user does not prefer to use the greedy construction heuristic, then the initial mission plan is shown in Table 22. This initial mission plan has 24 conflicts and obviously is not a good mission plan either from the assignment perspective or from the scheduling perspective. The best mission plan uses 10 tanker aircraft and has no conflicts in the solution. We reached this solution in 7 minutes.

Table 20: Initial Solution of Southeast Asia Deployment without greedy approach

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	KMUO	MOUNTAIN HOME AFB	1	1
1	2	KMUO	MOUNTAIN HOME AFB	0.94669016	1
1	3	PAEI	EIELSON AFB	4.24690027	1
1	4	RODN	KADENA AB	5.1760586	1
1	5	PAEI	EIELSON AFB	4.3473912	1
2	1	PAEI	EIELSON AFB	1	1
2	1	PAEI	EIELSON AFB	1	1
2	3	PAEI	EIELSON AFB	2.44828868	1
3	1	RODN	KADENA AB	1	1
3	2	RODN	KADENA AB	8.63640049	1
4	1	PAEI	EIELSON AFB	1	1
4	1	PAEI	EIELSON AFB	1	1
4	2	RODN	KADENA AB	2.96460085	1
5	1	RODN	KADENA AB	1	1
5	1	RODN	KADENA AB	1	1
5	2	PAEI	EIELSON AFB	4.5300729	1
6	1	RODN	KADENA AB	1	1
7	1	RODN	KADENA AB	1	1
8	1	RODN	KADENA AB	1	1
9	1	KMUO	MOUNTAIN HOME AFB	1	1
9	1	KMUO	MOUNTAIN HOME AFB	1	1
10	1	PAEI	EIELSON AFB	1	1
11	1	KMUO	MOUNTAIN HOME AFB	1	1
11	2	KMUO	MOUNTAIN HOME AFB	0.94669016	1
11	3	PAEI	EIELSON AFB	4.24690027	1
11	4	RODN	KADENA AB	5.1760586	1
11	5	PAEI	EIELSON AFB	4.3473912	1

When we examine the initial solution created without using the greedy construction heuristic, we see that even though the assignment of tanker bases to refueling points seems good, there are many conflicts in the mission plan. The same tanker aircraft is assigned to different kinds of missions at the same time. This is not practical. These

conflicts are reduced by reactive tabu search algorithm at each iteration. The final mission plan after 100 iteration does not have any conflicts and is shown in Table 23.

Table 21: Solution after 100 iteration without greedy

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	KMUO	MOUNTAIN HOME AFB	1	1
1	2	KMUO	MOUNTAIN HOME AFB	0.9466902	2
1	3	PAEI	EIELSON AFB	4.2469003	1
1	4	RODN	KADENA AB	5.1760586	1
1	5	PAEI	EIELSON AFB	4.3473912	2
2	1	PAEI	EIELSON AFB	14.347391	1
2	1	PAEI	EIELSON AFB	14.347391	2
2	3	PAEI	EIELSON AFB	15.79568	3
3	1	RODN	KADENA AB	15.176059	1
3	2	RODN	KADENA AB	22.812459	2
4	1	PAEI	EIELSON AFB	25.79568	1
4	1	PAEI	EIELSON AFB	25.79568	3
4	2	RODN	KADENA AB	27.760281	1
5	1	RODN	KADENA AB	37.760281	1
5	1	RODN	KADENA AB	37.760281	2
5	2	PAEI	EIELSON AFB	41.290354	1
6	1	RODN	KADENA AB	47.760281	1
7	1	RODN	KADENA AB	57.760281	1
8	1	RODN	KADENA AB	67.760281	1
9	1	KMUO	MOUNTAIN HOME AFB	11	1
9	1	KMUO	MOUNTAIN HOME AFB	11	3
10	1	PAEI	EIELSON AFB	51.290354	1
11	1	KMUO	MOUNTAIN HOME AFB	73.584222	1
11	2	KMUO	MOUNTAIN HOME AFB	73.530912	4
11	3	PAEI	EIELSON AFB	76.831122	1
11	4	RODN	KADENA AB	77.760281	1
11	5	PAEI	EIELSON AFB	76.931613	2

For this best solution the latest TOA is 84 hours after the deployment begins. The timeline for initial and best mission plan is shown in Table 24. Scheduling of the initial and best mission plan on a time scale is shown in Figures 12 and 13, respectively.

Table 22: Timeline for Initial and Best Mission Plan in RTS, without Greedy

SORTIE ID	INITIAL MISS. PLAN TOD	INITIAL MISS. PLAN TOA	BEST MISSION PLAN TOD	BEST MISSION PLAN TOA
1	0.6	12.05	0.6	12.05
2	0.8	10.9	12.4	23.5
3	3.5	11	17.7	25.1
4	1.09	8.5	25.9	33.3
5	1.06	12.3	37.8	49.06
6	5.9	12.1	40.8	53
7	3.8	13.3	52.9	66.2
8	4.2	14.5	62.5	76.9
9	4.6	14.9	5.3	20.3
10	4.4	13.2	45.8	59.09
11	0.6	12.05	71.9	84.6

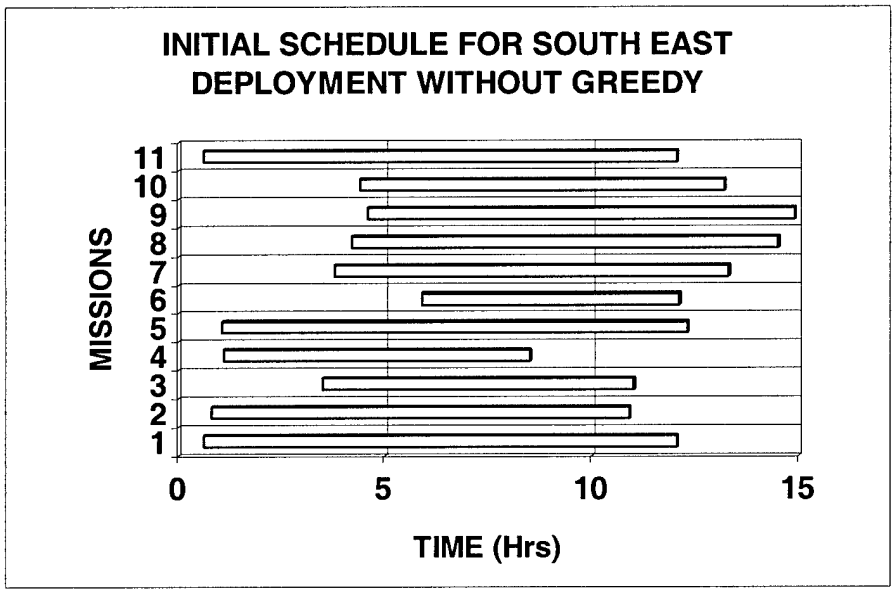

Figure 14: Initial Schedule for South East Deployment in RTS without Greedy

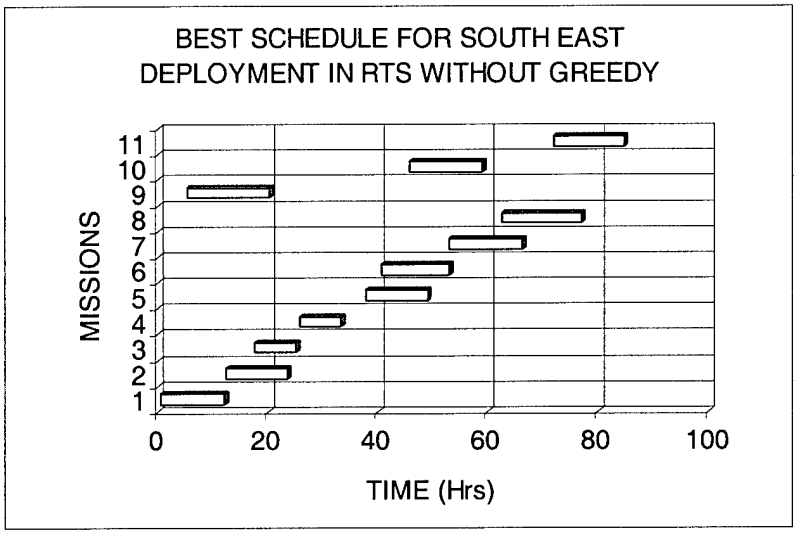

Figure 15: Best Schedule for South East Deployment in RTS without Greedy

4.3 DECREASING THE AVAILABLE TANKERS FOR SOUTHEAST ASIA DEPLOYMENT

We decreased the available tanker numbers from 15 to 5 at each of the bases. The tanker bases activated for this deployment remained the same. Initial conditions are shown in Table 25. Since we decreased the available tanker numbers, the tool produced fewer decision variables, so the solution time decreased considerably. Initial mission plan and best mission plan after 100 iterations are shown in Tables 26 and 27, respectively.

Table 23: Decreasing the Number of Aircraft on the bases

TANKER BASE	ICAO CODE	NUMBER OF TANKERS
McCONNELL	KIAB	5
MOUNTAIN HOME	KMUO	5
GRAND FORKS	KRDR	5
FAIRCHILD	KSKA	5
KADENA	RODN	5
EIELSON	PAEI	5

This initial mission plan has 15 conflicts, which are highlighted in the Table 26.

Table 24: Initial Solution of RTS with 5 tankers in each base

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	KSKA	FAIRCHILD AFB	6	1
1	2	PAEI	EIELSON AFB	7.749351399	1
1	3	PAEI	EIELSON AFB	9.448913655	2
1	4	RODN	KADENA AB	10.37807198	1
1	5	RODN	KADENA AB	15.46118662	2
2	1	PAEI	EIELSON AFB	6	3
2	1	PAEI	EIELSON AFB	6	4
2	3	RODN	KADENA AB	13.26066185	3
3	1	RODN	KADENA AB	1	4
3	2	RODN	KADENA AB	8.636400492	5
4	1	PAEI	EIELSON AFB	1	5
4	1	PAEI	EIELSON AFB	1	5
4	2	RODN	KADENA AB	2.964600846	4
5	1	RODN	KADENA AB	21	1
5	1	RODN	KADENA AB	21	4
5	2	RODN	KADENA AB	30.4902287	4
6	1	RODN	KADENA AB	31	1
7	1	RODN	KADENA AB	41	1
8	1	RODN	KADENA AB	51	1
9	1	KSKA	FAIRCHILD AFB	6	2
9	1	KSKA	FAIRCHILD AFB	6	3
10	1	PAEI	EIELSON AFB	21	1
11	1	KSKA	FAIRCHILD AFB	6	4
11	2	KSKA	FAIRCHILD AFB	6.670367819	5
11	3	PAEI	EIELSON AFB	9.448913655	3
11	4	RODN	KADENA AB	10.37807198	2
11	5	RODN	KADENA AB	15.46118662	1
11	5	RODN	KADENA AB	15.46118662	1
11	5	RODN	KADENA AB	15.46118662	1

Table 25: Best Solution after 100 Iterations with 5 tankers in each base

SORTIE ID	RP #	ICAO CODE	BASE NAME	TAKE-OFF TIME	TANKER TAIL #
1	1	PAEI	EIELSON AFB	2.92974044	1
1	2	PAEI	EIELSON AFB	7.749351399	2
1	3	PAEI	EIELSON AFB	9.448913655	3
1	4	RODN	KADENA AB	10.37807198	1
1	5	RODN	KADENA AB	15.46118662	2
2	1	PAEI	EIELSON AFB	18.20052477	1
2	1	PAEI	EIELSON AFB	18.20052477	2
2	3	RODN	KADENA AB	25.46118662	2
3	1	RODN	KADENA AB	22.74167149	4
3	2	RODN	KADENA AB	30.37807198	1
4	1	PAEI	EIELSON AFB	30.77707064	1
4	1	PAEI	EIELSON AFB	30.77707064	2
4	2	RODN	KADENA AB	32.74167149	4
5	1	RODN	KADENA AB	50.37807198	1
5	1	RODN	KADENA AB	50.37807198	2
5	2	RODN	KADENA AB	59.86830068	4
6	1	RODN	KADENA AB	65.46118662	2
7	1	RODN	KADENA AB	69.86830068	4
8	1	RODN	KADENA AB	75.46118662	2
9	1	KSKA	FAIRCHILD AFB	16	1
9	1	KSKA	FAIRCHILD AFB	16	2
10	1	PAEI	EIELSON AFB	40.77707064	1
11	1	KSKA	FAIRCHILD AFB	81.08311464	1
11	2	KSKA	FAIRCHILD AFB	81.75348246	2
11	3	PAEI	EIELSON AFB	84.53202829	2
11	4	RODN	KADENA AB	85.46118662	2
11	5	RODN	KADENA AB	90.54430126	1
11	5	RODN	KADENA AB	90.54430126	3

This best solution uses 10 tankers. We reached this solution in 5 minutes. There are no conflicts in this solution. Total distance traveled by all tankers is 101.371Nm. This distance is the minimum distance that we find up to this point. We do not pass the allowable time limits for this deployment. Latest receiver groups TOA is 92 hours after the deployment begins. Decreasing the available number of tankers at each base

decreased the computation time for this deployment, but we do not know exactly how much we should decrease the tanker numbers. Receiver group TOD and TOA values are shown in Table 28. Schedule for this deployment is shown in Figure 14.

Table 26: TOD and TOA with 5 Tankers in Each Base

Receiver Group #	TOD (hours)	TOA (hours)
1	4.5	17.2
2	16.3	27.3
3	25.3	32.7
4	30.8	38.2
5	50.4	61.8
6	58.5	70.7
7	65.03	78.3
8	70.2	84.6
9	10.2	25.2
10	35.3	48.5
11	79.6	92.3

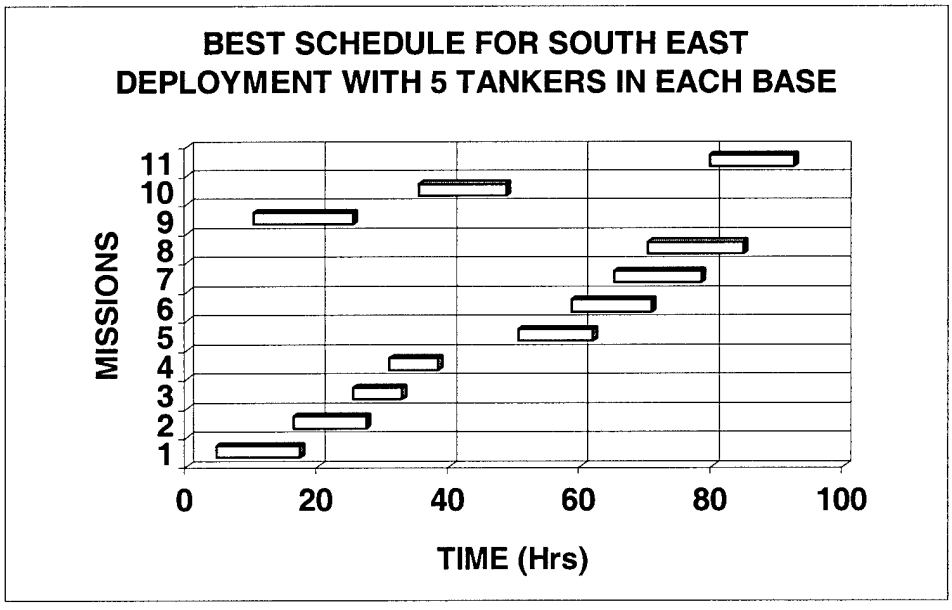

Figure 16: Best Schedule for South East Deployment with 5 Tankers in Each Base

4.4 CONCLUSION

The initial mission plan may have conflicts. It is observed that the initial solution produced by the greedy approach has fewer conflicts when compared with the random method. As the iteration count increased, the number of conflicts decreased in both the TAP Tool and the revised tool.

A primary goal is that every receiver group reaches its destination before its required delivery date. This goal is achieved in both of the tools. At the end of 100 iterations, both tools completed the deployment within its time limits.

A secondary goal is the reuse of tankers. In both tools the number of tankers used decreased gradually. We should bear in mind that actual tanker sorties are always the same. We reused the same tail numbered tankers by stretching the deployment to its allowable time limits. By doing so, we had an ability to reuse some tankers.

Computer time spent for this deployment decreased from 20 minutes to 13 minutes. The main reason behind this is the use of different computers. In any case, the time for 100 iterations is short when compared with CMARPS.

Assignment of tanker bases to refueling points by both tools is good. Even though some of the support bases have changed for some refueling points, it is concluded that the overall evaluation of the assignment of tankers for the missions is appropriate for both tools.

When we decreased the number of available tankers from 15 to 5 for South East Deployment, the time to find a solution decreased. The algorithm required fewer decision variables. The solution quality for this deployment was better than the previous ones. We

do not have the ability to determine how much to decrease the available number of tankers and still find a good solution.

By using an elite candidate list strategy, we decreased the possible number of solutions considerably during the assignment phase. In the scheduling phase, the number of possible solutions increased tremendously because we introduced the time factor.

4.4.1 Problems with the TAP Tool

Air Force regulations require tanker escort for fighter aircraft while crossing large bodies of water. But tanker aircraft escort fighter aircraft even if its mission route is on the land, so we need to use a database consisting of land formation locations.

Tankers are always required to return to their take off base. This reduces the capability of tanker offloading, and also reduces the total distance a tanker can escort a mission.

A tanker is assigned to a refueling point. A tanker might be able to satisfy the fuel requirements of a receiver group. Even in this situation, one tanker is assigned to every refueling point of that mission.

Although high altitude jet streams affect fuel burn rates, ground speed, and true course, it is not included in the tool.

The previous quick look tool used a third order polynomial equation to determine the fuel burned by an aircraft. The TAP Tool determines flight distances based on fuel flow. Because of this discrepancy, the TAP Tool does not use the fuelburn functions in the code. Fuel burn rates can be found in flight manuals of every aircraft. This value is

dependent upon many factors such as altitude, true air speed, wind, etc, so it is subject to change for different scenarios. A database for fuel burn rates would decrease the run time, but increase the precision of the solutions.

CHAPTER 5. FUTURE RESEARCH

5.1 RECOMMENDATIONS

The code is written in Visual Basic for Applications (VBA) within Excel. Translating the code to Java might decrease the computation time.

Adding wind effects into tool would improve result precision. Forming a database for fuelburn rates for different types of receiver aircraft by using their original flight manual values would increase the solution precision.

Using a database of land formation locations would help achieve more accurate results, because the results for total tanker distance traveled are misleading right now.

Adding a user-friendly map, and being able to add tanker bases and flight routes on it, would help visualize the scenarios, and help the decision makers understand the scenarios more readily. Adding restricted airspace information, thus being able to produce different deployment routes would be a positive contribution. Adding Gantt charts to the output files will help the decision maker to visualize the deployment schedule.

The ability to give weights to missions might produce more effective schedules. By using preemptive goal programming techniques, more flexible schedules can be created.

Using probabilities for missed refueling, delayed disconnects, and aircraft malfunctions might be added for future analyses.

Adding the ability of using KC-10 Extender for these deployments might produce more flexible solution alternatives. The available offload of a KC-10 is almost two times

greater than a KC-135R. A check would need to be made on the tanker type before it is scheduled.

An upgrade to the scheduling process would be to allow the user to interact with the computer as it prepares the schedule.

APPENDIX A

```
ada(1, 1) = initSol(1, 3)
ada(1, 2) = 1
adasize = 1
flaga = True
For x = 2 To RPcount
 For ys = 1 To adasize
  If ada(ys, 1) = initSol(x, 3) Then
   flaga = False
  End If
 Next ys
 If flaga = True Then
  adasize = adasize + 1
  ada(adasize, 1) = initSol(x, 3)
  ada(adasize, 2) = x
 End If
 flaga = True
Next x


div = 0
Dim cu As Boolean
cu = False
Dim flag As Boolean
flag = True
Cycle = 0
sss = 1
hh = 0
m = 0
v = 0
For k = 1 To numIterations
TENURE = 1
bestVal = 1000000
   ' Display the current iteration.
   Sheets("input").Cells(2, 12) = k

   Skip = (k Mod modNum) + 1

   ' Count how many tankers are reused in the current solution.
   For q = 1 To RPcount - 1
      p = 1
      goon = True
      While goon = True
         If sched(q, 6) & sched(q, 3) = sched(q + p, 6) & sched(q + p, 3) Then
```

```
            If p + q <> i And q <> i Then
               currentReuse = currentReuse + 1
               goon = False
            End If
         End If
         If p = RPcount Then
            goon = False
         Else
            p = p + 1
         End If
      Wend
   Next q

   i = 1
   While i < RPcount + 1

      ' First look at the current solution to count how many conflicts there are.

      conflictNumbers = 0
      currentNegTODs = 0
      If CloseLook(i) = True Then
         If sched(i, 2) = 1 Then

            ' Count how many TODs are less than zero.
            If sched(i, 9) < 0 Then currentNegTODs = currentNegTODs + 1

            h = 0
            While sched(i + h, 1) = sched(i, 1)
               For n = 1 To RPcount
                  If sched(n, 6) & sched(n, 3) = sched(i + h, 6) & sched(i + h, 3) And i +
h <> n Then
                     If sched(n, 5) > sched(i + h, 5) Then
                        temp1 = (sched(n, 5) - (3 + sched(i + h, 5) + sched(i + h, 7) / 430))
                     Else
                        temp1 = (sched(i + h, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
                     End If
                     If temp1 < 0 Then conflictNumbers = conflictNumbers + 1
                  End If
               Next n
               h = h + 1
            Wend
         Else

            For n = 1 To RPcount
```

```
            If sched(n, 6) & sched(n, 3) = sched(i, 6) & sched(i, 3) And i <> n Then
              If sched(n, 5) > sched(i, 5) Then
                temp1 = (sched(n, 5) - (3 + sched(i, 5) + sched(i, 7) / 430))
              Else
                temp1 = (sched(i, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
              End If
              If temp1 < 0 Then conflictNumbers = conflictNumbers + 1
            End If
          Next n
        End If
        ' If there are no problems with the current solution, don't take
        ' a close look at it again.
        If currentNegTODs = 0 And conflictNumbers = 0 Then CloseLook(i) = False
      End If


    ' Look at each neighbor by changing one RP at a time
    TENURE = 1
    j = RPindex(i)
    While j < RPindex(i + 1)


      ' Save the current solution
      For p = 1 To RPcount
        For n = 1 To 15
          tempVar(p, n) = sched(p, n)
        Next n
      Next p

      ' Change to a new DV for this RP
      sched(i, 1) = DV(j, 1)
      sched(i, 2) = DV(j, 2)
      sched(i, 3) = DV(j, 3)
      sched(i, 4) = DV(j, 4)
      sched(i, 6) = DV(j, 8)
      sched(i, 7) = DV(j, 5)
      sched(i, 11) = DV(j, 12)

      ind = Find(sched(i, 1), missplan, missions)
'XXXXXXXX

      ' If we are changing a RP other than a 1st RP, calculate the
      ' take-off time for this tanker.
      If DV(j, 2) <> 1 Then
        a = 0
```

```
            temp1 = False
            While temp1 = False
              a = a + 1
              If sched(i - a, 2) < 2 Then temp1 = True
            Wend
              If sched(i, 1) = Empty Then
              sched(i, 5) = Empty
              Else
              If ind = -1 Then
              ind = missions
              End If
              sched(i, 5) = sched(i - a, 8) + (sched(i, 2) - sched(i - a, 2)) * (rpvals(ind, 6) /
recrqmts(ind, 3)) - DV(tempIndex, 12) / tankers(3, 3)
              End If
            Else
              sched(i, 5) = DV(tempIndex, 7)
            End If

            ' Determine the time that this refueling will take place
            If sched(i, 1) = Empty Then
            sched(i, 8) = Empty
            Else
            sched(i, 8) = sched(i, 5) + DV(tempIndex, 12) / tankers(3, 3)
            End If
          ' If this is the first refueling point, determine when the receiver
          ' group will arrive at the destination
          If sched(i, 2) = 1 Then

            If sched(i, 1) & sched(i, 2) = sched(i + 1, 1) & sched(i + 1, 2) Then
              sched(i + 1, 8) = sched(i, 8)
            End If

            ' Also calculate the new take-off time for any other tankers
            ' assigned to other refueling points with this same RG.
            n = 1
            While sched(i + n, 1) = sched(i, 1)
              sched(i + n, 5) = sched(i, 8) + (sched(i + n, 2) - sched(i, 2)) * (rpvals(ind, 6) /
recrqmts(ind, 3)) - sched(i + n, 11) / tankers(3, 3)
              n = n + 1
            Wend

            TOA = ((missplan(ind, 7) - rpvals(ind, 8)) / recrqmts(ind, 3)) + sched(i, 8)
            sched(i, 10) = TOA
            TOD = TOA - (missplan(ind, 7) / recrqmts(ind, 3))
            sched(i, 9) = TOD
```

```
        End If

        ' Evaluate this new solution somehow
        distPen = 0
        temp1 = 0
        newConflict = 0
        conflictPen = 0
        conflictBonus = 0
        latePen = 0
        earlyBonus = 0
        newReuse = 0
        reuseBonus = 0
        earlyPen = 0
        negTODpen = 0
        syncPen = 0

        ' If this is a DV already in the solution, set the penalty to Big-M
        ' so that it is not chosen for the current move.
        If sched(i, 3) & sched(i, 5) & sched(i, 6) = tempVar(i, 3) & tempVar(i, 5) &
tempVar(i, 6) Then
          If i = 1 Then

            For d = 1 To numTbases
              If DV(j, 3) = tankerplacement(d, 1) Then
                dd = tankerplacement(d, 2)
              End If
            Next d

            If dd = 1 Then
              DV(j, 10) = 1000000
            Else
              'j = j + 15 - 1
              DV(j, 10) = 100 * DV(j, 12)
              DV(j - 1, 10) = DV(j, 10)
            End If


          Else                    'if not the first rp


            For d = 1 To numTbases
              If DV(j, 3) = tankerplacement(d, 1) Then
                dd = tankerplacement(d, 2)
              End If
            Next d
```

```
            If dd = 1 Then
              DV(j, 9) = 0
              DV(j, 10) = 100 * DV(j, 12)
            Else
              'j = j + 15 - 1
              DV(j, 10) = 100 * DV(j, 12)
              DV(j - 1, 10) = DV(j, 10)
            End If

          End If

        Else

          ' penalize for making the receivers take off too early.
          If TOD < 0 Then earlyPen = 500000

          If earlyPen <> 500000 Then

            ' penalize for late receivers
            If sched(i, 2) = 1 Then
              If Max(0, ((TOA) - 24 * missplan(ind, 6))) > 0 Then latePen = 600000
              earlyBonus = 10 * Max(0, ((24 * missplan(ind, 6)) - (TOA)))
            End If

            If latePen <> 600000 Then
              For n = 1 To RPcount
                distPen = distPen + sched(n, 7)
                ' penalize for having the same aircraft take off sooner than 10 hours
apart
                If sched(n, 6) & sched(n, 3) = sched(i, 6) & sched(i, 3) And i <> n Then
                  If sched(n, 5) > sched(i, 5) Then
                    temp1 = (sched(n, 5) - (3 + sched(i, 5) + sched(i, 7) / 430))
                  Else
                    temp1 = (sched(i, 5) - (3 + sched(n, 5) + sched(n, 7) / 430))
                  End If
                  If temp1 < 0 Then newConflict = newConflict + 1
                  If sched(i, 6) & sched(i, 3) <> tempVar(i, 6) & tempVar(i, 3) Then
newReuse = newReuse + 1
                End If

                ' penalize for not having multiple first refueling points at the same time
                If sched(n, 2) = 1 And sched(i, 2) = 1 And sched(n, 1) = sched(i, 1)
Then
                  If sched(n, 8) <> sched(i, 8) Then syncPen = 40000
```

```
            End If
        Next n

        ' If this is the first refueling point for a receiver group,
        ' check the other refueling points to make sure you haven't
        ' created a conflict.
        If sched(i, 2) = 1 Then
            h = 0
            While sched(i + h, 1) = sched(i, 1)
                If h > 0 Then
                    ' Take-off Time
                    sched(i + h, 5) = sched(i, 8) + (sched(i + h, 2) - sched(i, 2)) *
(rpvals(ind, 6) / recrqmts(ind, 3)) - sched(i + h, 11) / tankers(3, 3)
                    ' Time of refueling
                    sched(i + h, 8) = sched(i + h, 5) + sched(i + h, 11) / tankers(3, 3)
                End If
                For n = 1 To RPcount
                    If sched(n, 6) & sched(n, 3) = sched(i + h, 6) & sched(i + h, 3) And
i + h <> n Then
                        If sched(n, 5) > sched(i + h, 5) Then
                            temp1 = (sched(n, 5) - (3 + sched(i + h, 5) + sched(i + h, 7) /
430))
                        Else
                            temp1 = (sched(i + h, 5) - (3 + sched(n, 5) + sched(n, 7) /
430))
                        End If
                        If temp1 < 0 Then newConflict = newConflict + 1
                    End If
                Next n
                h = h + 1
            Wend
        End If

        reuseBonus = 500 * newReuse + 2000 * currentReuse

        ' Reward if the number of conflicts has decreased.
        ' Penalize if there are more conflicts.
        If newConflict < conflictNumbers Then
            conflictBonus = -100000
        ElseIf newConflict > 0 Or conflictNumbers > 0 Then
            conflictPen = 90000
        End If
        If currentNegTODs > 0 Then
            negTODpen = -100000 * currentNegTODs
        End If
```

```
            End If
          End If

'         DV(j, 18) = conflictPen
'         DV(j, 19) = conflictBonus
'         DV(j, 20) = distPen
'         DV(j, 21) = reuseBonus
'         DV(j, 22) = earlyPen
'         DV(j, 23) = negTODpen
'         DV(j, 24) = earlyBonus
'         DV(j, 25) = latePen
'         DV(j, 26) = newConflict
'         DV(j, 27) = conflictNumbers
'         DV(j, 28) = CloseLook(i)
'         DV(j, 29) = infeasible


  DV(j, 10) = conflictPen + conflictBonus + (100 * DV(j, 12)) - reuseBonus + syncPen +
earlyPen + negTODpen - earlyBonus + latePen
 End If
 For p = 1 To RPcount
  For n = 1 To 15
   sched(p, n) = tempVar(p, n)
  Next n
 Next p
 j = j + 1
Wend
u = 1
t = i
yy = True
For j = RPindex(i) To RPindex(i + 1) - 1
 If yy = True Then
  NewSol(j, 1) = DV(j, 10)
  New2Sol = DV(j + 1, 10)
  If New2Sol <> NewSol(j, 1) Then
   j = j + 2
  Else
   If New2Sol = DV(j, 10) Then
    US = DV(j, 3)
    x = j + 1
    While DV(x, 3) = US
     x = x + 1
    Wend
    RPind(i, u) = x - 1
   Else
```

```
    If New2Sol <> Empty Then
     New2Sol = DV(j + 2, 10)
     US = DV(j + 2, 3)
     x = j + 1
     While DV(x, 3) = US
      x = x + 1
     Wend
     RPind(i, u) = x
    Else
     If New2Sol = Empty Then
      RPind(i, u) = j
      yy = False
     End If
    End If
   End If
   If j + 2 < RPindex(i + 1) - 1 Then
    objectivefunc(t, u) = DV(RPind(i, u), 10)
    supportbases(t, u) = DV(RPind(i, u), 3)
    j = x - 1
   Else
    j = DVcount - 1
   End If
   u = u + 1
  End If
 End If
Next j
i = i + 1
Wend
For t = 1 To RPcount
 sayi = 0
 For e = 1 To numTbases
  If supportbases(t, e) <> Empty Then
   sayi = sayi + 1
  End If
 Next e
 numb(t) = sayi
 u = 1
 minfortthRP = objectivefunc(t, u)
 If t <> Bul Then
  For u = 1 To sayi
   If objectivefunc(t, u) <= minfortthRP Then
    minfortthRP = objectivefunc(t, u)
    chosenbaseforthatRP(t, u) = supportbases(t, u)
    minforthatRP(t, u) = minfortthRP
    y(k, t) = u
```

'k: iteration number
't: RPnumber (t'th RP)
'u: u'th base supporting t'th RP

```
    y(k + 1, t) = u
   End If
  Next u
  If z = Empty Then
   z = u - 2
   If z < 1 Then
    z = 1
   End If
   y(k, t) = u
   y(k + 1, t) = u
  End If
 End If
Next t
If k = 1 Then
 f = 0
 For i = 1 To RPcount
  s = objectivefunc(i, (y(k, i)))
  f = s + f
  totobjforkthiteration(k) = f
 Next i
Else
 f = 0
 For i = 1 To RPcount
  If i <> Bul Then
   s = objectivefunc(i, (y(k, i)))
  Else
   s = objectivefunc(i, z)
  End If
  f = s + f
  totobjforkthiteration(k) = f
 Next i
End If
If k = 1 Then
 bestsofar = totobjforkthiteration(1)
End If
If totobjforkthiteration(k) <= bestsofar Then
 bestsofar = totobjforkthiteration(k)
End If
If f <> totobjforkthiteration(k - 1) Then
 TENURE = 1
 For i = 1 To RPcount
  oankicozum(k, i) = supportbases(i, (y(k, i)))
 Next i
 For i = 1 To RPcount
  s = objectivefunc(i, (y(k, i)))
```

```
 If s < objectivefunc(i - 1, (y(k, i - 1))) Then
  Bul = i
 End If
Next i
sayi = 0
For e = 1 To numTbases
 If objectivefunc(Bul, e) <> Empyt Then
  sayi = sayi + 1
 End If
Next e
If sayi = 2 Then
 degisken1 = objectivefunc(Bul, 1)
 degisken2 = objectivefunc(Bul, 2)
 If degisken1 < degisken2 Then
  Min = degisken1
  tabu(k + 1, 1) = degisken1
  swap = degisken1
 Else
  Min = degisken2
  tabu(k + 1, 1) = degisken2
  swap = degisken2
 End If
 For i = 1 To sayi
  If objectivefunc(Bul, i) = swap Then
   y(k + 1, Bul) = i
   cc = y(k + 1, Bul)
  End If
 Next i
End If
If sayi > 2 Then
 degisken1 = objectivefunc(Bul, 1)
 degisken2 = objectivefunc(Bul, 2)
 For x = 3 To sayi - 1
  If objectivefunc(Bul, x) < degisken1 Then
   h = objectivefunc(Bul, x)
   If degisken1 < degisken2 Then
    degisken2 = degisken1
   End If
   degisken1 = h
  Else
   If objectivefunc(Bul, x) < degisken2 Then
    zu = objectivefunc(Bul, x)
    If degisken2 < degisken1 Then
     degisken1 = degisken2
    End If
```

```
  degisken2 = zu
 End If
End If
Next x
If degisken1 < degisken2 Then
 Min = degisken1
 tabu(k + 1, 1) = degisken1
 swap = degisken1
Else
 Min = degisken2
 tabu(k + 1, 1) = degisken2
 swap = degisken2
End If
If Min = tabu(k, 1) Or Min = tabu(k, 2) Then
 degisken1 = objectivefunc(Bul, 1)
 degisken2 = objectivefunc(Bul, 2)
 For x = 1 To sayi
  If tabu(k, 1) <> objectivefunc(Bul, x) Then
   If objectivefunc(Bul, x) <= degisken1 Then
    h = objectivefunc(Bul, x)
    If degisken1 < degisken2 Then
     degisken2 = degisken1
     degisken1 = h
    End If
   Else
    If objectivefunc(Bul, x) <= degisken2 Then
     zu = objectivefunc(Bul, x)
     If degisken2 < degisken1 Then
      degisken1 = degisken2
      degisken2 = zu
     End If
    End If
   End If
  End If
 Next x
 If degisken1 < degisken2 Then
  Min = degisken1
  tabu(k + 1, 1) = degisken1
  swap = degisken2
 Else
  Min = degisken2
  tabu(k + 1, 1) = degisken2
  swap = degisken1
 End If
End If
```

```
 For i = 1 To sayi
  If objectivefunc(Bul, i) = swap Then
   y(k + 1, Bul) = i
   cc = y(k + 1, Bul)
  End If
 Next i
End If
If sayi = 1 Then
 Bul = Bul + 1
 If Bul > RPcount Then
  Bul = Bul - RPcount
  cc = 1
 End If
End If
Else
 TENURE = TENURE + 1
 For i = 1 To RPcount
  oankicozum(k, i) = supportbases(i, (y(k, i)))
 Next i
 Bul = Bul + 1
 If Bul > RPcount Then
  Bul = Bul - RPcount
 End If
 sayi = 0
 For e = 1 To numTbases
  If supportbases(Bul, e) <> Empyt Then
   sayi = sayi + 1
  End If
 Next e
 If sayi = 2 Then
  degisken1 = objectivefunc(Bul, 1)
  degisken2 = objectivefunc(Bul, 2)
  If degisken1 < degisken2 Then
   Min = degisken1
   tabu(k + 1, 1) = degisken1
   tabu(k + 2, 2) = degisken1
   swap = degisken1
  Else
   Min = degisken2
   tabu(k + 1, 1) = degisken2
   tabu(k + 2, 2) = degisken2
   swap = degisken2
  End If
  For i = 1 To sayi
   If objectivefunc(Bul, i) = swap Then
```

```
  y(k + 1, Bul) = i
  cc = y(k + 1, Bul)
 End If
Next i
End If
If sayi > 2 Then
 degisken1 = objectivefunc(Bul, 1)
 degisken2 = objectivefunc(Bul, 2)
 For x = 3 To sayi
  If objectivefunc(Bul, x) < degisken1 Then
   h = objectivefunc(Bul, x)
   If degisken1 < degisken2 Then
    degisken2 = degisken1
    degisken1 = h
   End If
  Else
   If objectivefunc(Bul, x) < degisken2 Then
    zu = objectivefunc(Bul, x)
    If degisken2 < degisken1 Then
     degisken1 = degisken2
     degisken2 = zu
    End If
   End If
  End If
 Next x
 If degisken1 < degisken2 Then
  Min = degisken1
  tabu(k + 1, 1) = degisken1
  tabu(k + 2, 2) = degisken1
  swap = degisken1
 Else
  Min = degisken2
  tabu(k + 1, 1) = degisken2
  tabu(k + 2, 2) = degisken2
  swap = degisken2
 End If
 If Min = tabu(k, 1) Or Min = tabu(k, 2) Then
  degisken1 = objectivefunc(Bul, 1)
  degisken2 = objectivefunc(Bul, 2)
  For x = 1 To sayi
   If tabu(k, 1) <> objectivefunc(Bul, x) Or tabu(k, 2) <> objectivefunc(Bul, x) Then
    If objectivefunc(Bul, x) <= degisken1 Then
     h = objectivefunc(Bul, x)
     If degisken1 < degisken2 Then
      degisken2 = degisken1
```

```
    degisken1 = h
   End If
  Else
   If objectivefunc(Bul, x) <= degisken2 Then
    zu = objectivefunc(Bul, x)
    If degisken2 < degisken1 Then
     degisken1 = degisken2
     degisken2 = zu
    End If
   End If
  End If
 End If
Next x
If degisken1 < degisken2 Then
 Min = degisken1
 tabu(k + 1, 1) = degisken1
 tabu(k + 2, 2) = degisken1
 swap = degisken2
Else
 Min = degisken2
 tabu(k + 1, 1) = degisken2
 tabu(k + 2, 2) = degisken2
 swap = degisken1
End If
End If
For i = 1 To sayi
 If objectivefunc(Bul, i) = swap Then
  y(k + 1, Bul) = i
  cc = y(k + 1, Bul)
 End If
Next i
End If
If sayi = 1 Then
 Bul = Bul + 1
 If Bul > RPcount Then
  Bul = Bul - RPcount
  cc = 1
 End If
End If
f = 0
For i = 1 To RPcount
 If i <> Bul Then
  s = objectivefunc(i, (y(k, i)))
 Else
  s = objectivefunc(i, cc)
```

```
 End If
 f = s + f
 totobjforkthiteration(k) = f
Next i
For d = 1 To RPcount
 If i <> Bul Then
  For n = 1 To 15
   sched(i, n) = sched(i, n)
  Next n
 End If
 If d = Bul Then
  For n = 1 To 15
   sched(d, n) = sched(i, n)
  Next n
 End If
Next d
If totobjforkthiteration(k) <= bestsofar Then
 bestsofar = totobjforkthiteration(k)
End If
End If
If (k <> 1 And flag = True) Then
 For i = k - 1 To sss Step -1
  If cu <> True Then
   If totobjforkthiteration(k) = totobjforkthiteration(i) Then
    flag = False
    v = k - i
    hh = k + v
    cu = True
   End If
  End If
 Next i
End If
If k = hh Then
 cu = False
 qq = 1
 For i = 0 To v - 1
  If totobjforkthiteration(k - v + i) <> totobjforkthiteration(k - 2 * v + i) Then
   qq = 0
  End If
 Next i
 If qq <> 0 Then
  Cycle = v
  flag = True
  sss = hh
  div = div + 1
```

```
Bul = Bul + 1
If Bul > RPcount Then
 Bul = Bul - RPcount
End If
sayi = 0
For e = 1 To numTbases
 If supportbases(Bul, e) <> Empyt Then
  sayi = sayi + 1
 End If
Next e
If sayi >= 2 Then
 degisken1 = objectivefunc(Bul, 1)
 degisken2 = objectivefunc(Bul, 2)
 For x = 3 To sayi
  If objectivefunc(Bul, x) < degisken1 Then
   h = objectivefunc(Bul, x)
   If degisken1 < degisken2 Then
    degisken2 = degisken1
    degisken1 = h
   End If
  Else
   If objectivefunc(Bul, x) < degisken2 Then
    zu = objectivefunc(Bul, x)
    If degisken2 < degisken1 Then
     degisken1 = degisken2
     degisken2 = zu
    End If
   End If
  End If
 Next x
 If degisken1 < degisken2 Then
  Min = degisken1
  tabu(k + 1, 1) = degisken1
  swap = degisken1
  dew = degisken2
 Else
  Min = degisken2
  tabu(k + 1, 1) = degisken2
  swap = degisken2
  dew = degisken1
 End If
End If
For i = 1 To sayi
 If objectivefunc(Bul, i) = dew Then
  y(k + 1, Bul) = i
```

```
  cc = y(k + 1, Bul)
 End If
Next i
If sayi = 1 Then
 Bul = Bul + 1
 If Bul > RPcount Then
  Bul = Bul - RPcount
  cc = 1
 End If
End If
f = 0
For i = 1 To RPcount
 If i <> Bul Then
  s = objectivefunc(i, (y(k, i)))
 Else
  s = objectivefunc(i, cc)
 End If
 f = s + f
 totobjforkthiteration(k) = f
Next i
For d = 1 To RPcount
 If i <> Bul Then
  For n = 1 To 15
   sched(i, n) = sched(i, n)
  Next n
 End If
 If d = Bul Then
  For n = 1 To 15
   sched(d, n) = sched(i, n)
  Next n
 End If
Next d
If totobjforkthiteration(k) <= bestsofar Then
 bestsofar = totobjforkthiteration(k)
End If
End If
End If
flag = True

If greedy = "Y" Then
If k <= RPcount Then
devam = True
For i = 1 To RPcount - 1
 If devam = True Then
  bg = 0
```

```
For p = 1 To RPcount - i
 If devam = True Then
  gg = True
  If tempVar(i, 3) = tempVar(i + p, 3) Then

   If devam = True Then
    If tempVar(i, 1) <> tempVar(i + p, 1) Then
     If devam = True Then
      bur = True
      For hm = k To i Step -1
       If tempVar(saz(hm - 1), 1) = tempVar(i + p, 1) Then
        bur = False
        If bur = False Then
         hm = i
        End If
       End If
      Next hm
      If i + p <= saz(k - 1) And bur = False Then
       p = p + 1
       bg = bg + 1
       gg = False
      End If
      If gg = True Then
       If tempVar(saz(k - 1), 1) <> tempVar(i + p, 1) Then
        If tempVar(i, 6) <> tempVar(i + p, 6) And tempVar(i, 1) <> tempVar(i + p, 1)
Then
         If tempVar(saz(k - 1), 1) <> tempVar(i + p, 1) Then
          If tempVar(i, 6) <> tempVar(i + p, 6) Then
           If tempVar(i + p, 2) = tempVar(i + p + 1, 2) Then
            fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
            ss = tempVar(i + p, 1)
            vv = 0
            r = i + p
            For ff = r To 1 Step -1
             If tempVar(ff, 1) = ss Then
              vv = ff
             End If
            Next ff
            gorev = tempVar(vv, 1)
            If tempVar(vv, 10) + fark < tl Then
             tempVar(i + p, 6) = tempVar(i, 6)

             While tempVar(vv, 1) = gorev
              tempVar(vv, 5) = tempVar(vv, 5) + fark
              tempVar(vv, 8) = tempVar(vv, 8) + fark
```

```
   tempVar(vv, 9) = tempVar(vv, 9) + fark
   tempVar(vv, 10) = tempVar(vv, 10) + fark
   vv = vv + 1
  Wend
  devam = False
  saz(k) = i + p
 End If
Else
 fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
 ss = tempVar(i + p, 1)
 vv = 0
 r = i + p
 For ff = r To 1 Step -1
  If tempVar(ff, 1) = ss Then
   vv = ff
  End If
 Next ff
 gorev = tempVar(vv, 1)
 If tempVar(vv, 10) + fark < tl Then
  tempVar(i + p, 6) = tempVar(i, 6)
  While tempVar(vv, 1) = gorev
   tempVar(vv, 5) = tempVar(vv, 5) + fark
   tempVar(vv, 8) = tempVar(vv, 8) + fark
   tempVar(vv, 9) = tempVar(vv, 9) + fark
   tempVar(vv, 10) = tempVar(vv, 10) + fark
   vv = vv + 1
  Wend
  devam = False
  saz(k) = i + p
 End If
End If
Else
 fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
 ss = tempVar(i + p, 1)
 vv = 0
 r = i + p
 For ff = r To 1 Step -1
  If tempVar(ff, 1) = ss Then
   vv = ff
  End If
 Next ff
 gorev = tempVar(vv, 1)
 If tempVar(vv, 10) + fark < tl Then
  While tempVar(vv, 1) = gorev
   tempVar(vv, 5) = tempVar(vv, 5) + fark
```

```
            tempVar(vv, 8) = tempVar(vv, 8) + fark
            tempVar(vv, 9) = tempVar(vv, 9) + fark
            tempVar(vv, 10) = tempVar(vv, 10) + fark
            tempVar(vv, 6) = tempVar(i, 6)
            vv = vv + 1
           Wend
           devam = False
           saz(k) = i + p
          End If
         End If
        End If
       End If
      Else
      End If
     End If
    End If
   End If
  End If
 Next p
End If
Next i
For n = 1 To RPcount
 For ii = 1 To 15
  bestSol(n, ii) = tempVar(n, ii)
  sched(n, ii) = tempVar(n, ii)
 Next ii
Next n

Else

For i = 1 To RPcount
 If numb(i) > 1 Then
  geci(1) = tempVar(i, 3)
  geci(2) = tempVar(i, 4)
  geci(3) = tempVar(i, 7)
  geci(4) = tempVar(i, 11)
  geci(5) = tempVar(i, 5)
  For zi = 1 To numb(i)
   If geci(1) <> supportbases(i, zi) And geci(1) <> supportbases(Bul, cc) Then
    tempVar(i, 3) = supportbases(i, zi)
    tempVar(i, 4) = DV(RPind(i, zi), 4)
    tempVar(i, 7) = DV(RPind(i, zi), 5)
    tempVar(i, 11) = DV(RPind(i, zi), 12)
```

```
tempVar(i, 5) = tempVar(i, 8) - (tempVar(i, 11) / tankers(3, 3))
devam = True
If devam = True Then
 bg = 0
 sal = False
 For p = 1 To RPcount - i
  If sal = False Then
   If devam = True Then
    gg = True
    If tempVar(i, 3) = tempVar(i + p, 3) Then
     If devam = True Then
      If tempVar(i, 1) <> tempVar(i + p, 1) Then
       If devam = True Then
        bur = True
        For hm = k - RPcount To 1 Step -1
         If tempVar(saz(k - 1), 1) = tempVar(i + p, 1) Then
          bur = False
          If bur = False Then
           hm = 1
          End If
         End If
        Next hm
        If i + p <= saz(k - 1) And bur = False Then
         p = p + 1
         bg = bg + 1
         gg = False
        End If
        If gg = True Then
         If tempVar(i, 6) <> tempVar(i + p, 6) Then
          If tempVar(i + p, 1) = tempVar(i + p + 1, 1) Then
           If tempVar(i + p, 2) = tempVar(i + p + 1, 2) Then
            fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
            If fark <> 0 Then
             ss = tempVar(i + p, 1)
             vv = 0
             r = i + p
             For ff = r To 1 Step -1
              If tempVar(ff, 1) = ss Then
               vv = ff
              End If
             Next ff
             gorev = tempVar(vv, 1)
             If tempVar(vv, 10) + fark < tl Then
              tempVar(i + p, 6) = tempVar(i, 6)
              While tempVar(vv, 1) = gorev
```

```
    tempVar(vv, 5) = tempVar(vv, 5) + fark
    tempVar(vv, 8) = tempVar(vv, 8) + fark
    tempVar(vv, 9) = tempVar(vv, 9) + fark
    tempVar(vv, 10) = tempVar(vv, 10) + fark
    vv = vv + 1
   Wend
   devam = False
   saz(k) = i + p
  End If
 End If
Else
 fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
 If fark <> 0 Then
  ss = tempVar(i + p, 1)
  vv = 0
  r = i + p
  For ff = r To 1 Step -1
   If tempVar(ff, 1) = ss Then
    vv = ff
   End If
  Next ff
  gorev = tempVar(vv, 1)
  If tempVar(vv, 10) + fark < tl Then
   tempVar(i + p, 6) = tempVar(i, 6)
   While tempVar(vv, 1) = gorev
    tempVar(vv, 5) = tempVar(vv, 5) + fark
    tempVar(vv, 8) = tempVar(vv, 8) + fark
    tempVar(vv, 9) = tempVar(vv, 9) + fark
    tempVar(vv, 10) = tempVar(vv, 10) + fark
    vv = vv + 1
   Wend
   devam = False
   saz(k) = i + p
  End If
 End If
End If
Else
 fark = ((tempVar(i, 5) + 10) - tempVar(i + p, 5) + bg * 10)
 If fark > 0 Then
  ss = tempVar(i + p, 1)
  vv = 0
  r = i + p
  For ff = r To 1 Step -1
   If tempVar(ff, 1) = ss Then
    vv = ff
```

```
                End If
               Next ff
               gorev = tempVar(vv, 1)
               If tempVar(vv, 10) + fark < tl Then
                While tempVar(vv, 1) = gorev
                 tempVar(vv, 5) = tempVar(vv, 5) + fark
                 tempVar(vv, 8) = tempVar(vv, 8) + fark
                 tempVar(vv, 9) = tempVar(vv, 9) + fark
                 tempVar(vv, 10) = tempVar(vv, 10) + fark
                 vv = vv + 1
                Wend
                devam = False
                saz(k) = i + p
               End If
              End If
             End If
            End If
           Else
          End If
         End If
        End If
       End If
      End If
     End If
     If saz(k) <> Empty Then
      sal = True
      p = RPcount - i
     End If
    End If
   Next p
  End If
 Else
  End If
  If sal = True Then
   i = RPcount
  End If
  If saz(k) = Empty Then
   tempVar(i, 3) = geci(1)
   tempVar(i, 4) = geci(2)
   tempVar(i, 7) = geci(3)
   tempVar(i, 11) = geci(4)
   tempVar(i, 5) = geci(5)
  End If
 Next zi
End If
```

```
Next i
'---------
'asagisi fizibilite
'---------
For i = 1 To RPcount
 ger = False
 For kl = i + 1 To RPcount
  If ger = False Then
   If tempVar(i, 6) = tempVar(kl, 6) And tempVar(i, 3) = tempVar(kl, 3) Then
    taf = kl
    If kl < RPcount Then
     ger = True
    End If
   End If
  End If
  If ger = True Then
   If tempVar(i, 1) = tempVar(taf, 1) Then
    If tempVar(i, 3) = tempVar(taf, 3) Then
     tempVar(taf, 6) = tempVar(taf, 6) + 1
     ger = True
    End If
   Else
    If tempVar(i, 3) = tempVar(taf, 3) Then
     If tempVar(taf, 5) - tempVar(i, 5) - 10 < 0 Then
      fark = tempVar(i, 5) + 10 - tempVar(taf, 5)
      ss = tempVar(taf, 1)
      vv = 0
      r = taf
      For ff = r To 1 Step -1
       If tempVar(ff, 1) = ss Then
        vv = ff
       End If
      Next ff
      gorev = tempVar(vv, 1)
      While tempVar(vv, 1) = gorev
       tempVar(vv, 5) = tempVar(vv, 5) + fark
       tempVar(vv, 8) = tempVar(vv, 8) + fark
       tempVar(vv, 9) = tempVar(vv, 9) + fark
       tempVar(vv, 10) = tempVar(vv, 10) + fark
       vv = vv + 1
      Wend
     End If
    End If
   End If
  End If
```

```
  If ger = True Then
   ger = False
   kl = taf
  End If
 Next kl
Next i

End If

'------------------
'asagisi ayni gorevlerde kuyruk numarasini esitliyor
'------------------
If saz(k) = Empty Then
 For mmm = 1 To RPcount - 1
  If devam = True Then
   For z = 2 To RPcount
    If devam = True Then
     If tempVar(mmm, 1) <> tempVar(mmm + z, 1) Then
      If tempVar(mmm, 3) = tempVar(mmm + z, 3) And tempVar(mmm, 6) =
tempVar(mmm + z, 6) Then
       If tempVar(mmm + 1, 1) <> tempVar(mmm + z + 1, 1) Then
        If tempVar(mmm + 1, 3) = tempVar(mmm + z + 1, 3) And tempVar(mmm + z - 1,
6) <> tempVar(mmm + z + 1, 6) Then
         fark = (tempVar(mmm, 5) + 10) - tempVar(mmm + z, 5)
         ss = tempVar(mmm + z, 1)
         vv = 0
         r = mmm + z
         For ff = r To 1 Step -1
          If tempVar(ff, 1) = ss Then
           vv = ff
          End If
         Next ff
         gorev = tempVar(vv, 1)
         If tempVar(vv, 10) + fark < tl Then
          tempVar(mmm + z + 1, 6) = tempVar(mmm + 1, 6)
          While tempVar(vv, 1) = gorev
           tempVar(vv, 5) = tempVar(vv, 5) + fark
           tempVar(vv, 8) = tempVar(vv, 8) + fark
           tempVar(vv, 9) = tempVar(vv, 9) + fark
           tempVar(vv, 10) = tempVar(vv, 10) + fark
           vv = vv + 1
          Wend
          devam = False
          saz(k) = mmm + z + 1
         End If
```

```
      End If
     End If
    End If
   End If
   If saz(k) <> Empty Then
    devam = False
   End If
  End If
 Next z
End If
Next mmm
End If
For n = 1 To RPcount
 For ii = 1 To 15
  bestSol(n, ii) = tempVar(n, ii)
  sched(n, ii) = tempVar(n, ii)
 Next ii
Next n


Else


For mj = 1 To adasize
 If k > 10 Then mj = 2
 If k > 20 Then mj = 3
devam = True
 For g = ada(mj, 2) To RPcount - 2

 hd = tempVar(g, 3)
  saz(k) = Empty
  For gh = 1 To RPcount - g
  If fdt = True Then
   saz(k - 1) = 1
   fdt = False
  End If
  If devam = True Then
   If saz(k - 1) > gh Then
    gh = saz(k - 1)
   End If
   If tempVar(g, 6) = tempVar(g + gh, 6) And tempVar(g, 3) = tempVar(g + gh, 3) And
tempVar(g, 1) = tempVar(g + gh, 1) Then
    x = tempVar(g, 3)
```

```
  For hj = 1 To numTbases
   If tankBases(hj, 1) = x Then
    nu = tankBases(hj, 5)
    If tempVar(g + gh, 6) + 1 <= nu Then
     tempVar(g + gh, 6) = tempVar(g + gh, 6) + 1
    Else
     tempVar(g + gh, 6) = tempVar(g + gh, 6) + 1 - nu
    End If
    saz(k) = g + gh
    gh = RPcount - g
    g = RPcount
    hj = numTbases
    devam = False
   End If
  Next hj
 Else
  If tempVar(g, 6) = tempVar(g + gh, 6) And tempVar(g, 3) = tempVar(g + gh, 3) And
tempVar(g, 1) <> tempVar(g + gh, 1) Then
   If devam = True Then
    ss = tempVar(g + gh, 1)
    r = g + gh
    For ff = r To 1 Step -1
     If tempVar(ff, 1) = ss Then
      vv = ff
     End If
    Next ff
    gorev = tempVar(vv, 1)


   If tempVar(g + gh, 1) = tempVar(g + gh + 1, 1) And tempVar(g + gh, 3) = tempVar(g
+ gh + 1, 3) And tempVar(g + gh, 6) = tempVar(g + gh + 1, 6) Then

     tempVar(g + gh + 1, 6) = tempVar(saz(k - 1), 6) + 1
     If tempVar(g + gh + 1, 6) > nu Then
      tempVar(g + gh + 1, 6) = tempVar(saz(k), 6) + 1 - nu
     End If
     saz(k) = g + gh + 1
     devam = False
     fark = tempVar(saz(k - 1), 5) + 10 - tempVar(vv, 5) '- tempVar(saz(k-1), 5)
     If tempVar(vv, 10) + fark < tl Then
      While tempVar(vv, 1) = gorev
       tempVar(vv, 5) = tempVar(vv, 5) + fark
       tempVar(vv, 8) = tempVar(vv, 8) + fark
       tempVar(vv, 9) = tempVar(vv, 9) + fark
       tempVar(vv, 10) = tempVar(vv, 10) + fark
```

```
      vv = vv + 1
     Wend


     End If

    Else
     ss = tempVar(g + gh, 1)
     r = g + gh
     For ff = r To 1 Step -1
      If tempVar(ff, 1) = ss Then
       vv = ff
      End If
     Next ff
     gorev = tempVar(vv, 1)
     fark = tempVar(g, 5) + 10 - tempVar(vv, 5) '- tempVar(saz(k-1), 5)
     If tempVar(vv, 10) + fark < tl Then
      While tempVar(vv, 1) = gorev
       tempVar(vv, 5) = tempVar(vv, 5) + fark
       tempVar(vv, 8) = tempVar(vv, 8) + fark
       tempVar(vv, 9) = tempVar(vv, 9) + fark
       tempVar(vv, 10) = tempVar(vv, 10) + fark
       vv = vv + 1
      Wend
      saz(k) = g + gh
      devam = False
     End If
    End If
   End If
  End If
 End If
End If
Next gh

If saz(k) <> Empty Then

 g = RPcount
 fdt = False

Else
 g = ada(mj + 1, 2)
 If g = -1 Then
  devam = True
  g = 0
 End If
```

```
  fdt = True


 End If
 Next g
 If saz(k) <> Empty Then
  mj = adasize
 End If
Next mj


For n = 1 To RPcount
 For ii = 1 To 15
  bestSol(n, ii) = tempVar(n, ii)
  sched(n, ii) = tempVar(n, ii)
 Next ii
Next n

'----
If k > RPcount Then
For i = 1 To RPcount
 ger = False
 For kl = i + 1 To RPcount
  If ger = False Then
   If tempVar(i, 6) = tempVar(kl, 6) And tempVar(i, 3) = tempVar(kl, 3) Then
    taf = kl
    If kl < RPcount Then
     ger = True
    End If
   End If
  End If
  If ger = True Then
   If tempVar(i, 1) = tempVar(taf, 1) Then
    If tempVar(i, 3) = tempVar(taf, 3) Then
     tempVar(taf, 6) = tempVar(taf, 6) + 1
     ger = True
    End If
   Else
    If tempVar(i, 3) = tempVar(taf, 3) Then
     If tempVar(taf, 5) - tempVar(i, 5) - 10 < 0 Then
      fark = tempVar(i, 5) + 10 - tempVar(taf, 5)
      ss = tempVar(taf, 1)
      vv = 0
      r = taf
      For ff = r To 1 Step -1
```

```
      If tempVar(ff, 1) = ss Then
       vv = ff
      End If
     Next ff
     gorev = tempVar(vv, 1)
     While tempVar(vv, 1) = gorev
      tempVar(vv, 5) = tempVar(vv, 5) + fark
      tempVar(vv, 8) = tempVar(vv, 8) + fark
      tempVar(vv, 9) = tempVar(vv, 9) + fark
      tempVar(vv, 10) = tempVar(vv, 10) + fark
      vv = vv + 1
     Wend
    End If
   End If
  End If
 End If
 If ger = True Then
  ger = False
  kl = taf
 End If
Next kl
Next i
End If
End If

If saz(k) = Empty Then
 For mmm = 1 To RPcount - 1
  If devam = True Then
   For z = 2 To RPcount
    If devam = True Then
     If tempVar(mmm, 1) <> tempVar(mmm + z, 1) Then
      If tempVar(mmm, 3) = tempVar(mmm + z, 3) And tempVar(mmm, 6) =
tempVar(mmm + z, 6) Then
       If tempVar(mmm + 1, 1) <> tempVar(mmm + z + 1, 1) Then
        If tempVar(mmm + 1, 3) = tempVar(mmm + z + 1, 3) And tempVar(mmm + z - 1,
6) <> tempVar(mmm + z + 1, 6) Then
         fark = (tempVar(mmm, 5) + 10) - tempVar(mmm + z, 5)
         ss = tempVar(mmm + z, 1)
         vv = 0
         r = mmm + z
         For ff = r To 1 Step -1
          If tempVar(ff, 1) = ss Then
           vv = ff
          End If
         Next ff
```

```
        gorev = tempVar(vv, 1)
        If tempVar(vv, 10) + fark < tl Then
         tempVar(mmm + z + 1, 6) = tempVar(mmm + 1, 6)
         While tempVar(vv, 1) = gorev
          tempVar(vv, 5) = tempVar(vv, 5) + fark
          tempVar(vv, 8) = tempVar(vv, 8) + fark
          tempVar(vv, 9) = tempVar(vv, 9) + fark
          tempVar(vv, 10) = tempVar(vv, 10) + fark
          vv = vv + 1
        Wend
        devam = False
        saz(k) = mmm + z + 1
        End If
       End If
      End If
     End If
    End If
    If saz(k) <> Empty Then
     devam = False
    End If
   End If
  Next z
 End If
Next mmm
End If
For n = 1 To RPcount
 For ii = 1 To 15
  bestSol(n, ii) = tempVar(n, ii)
  sched(n, ii) = tempVar(n, ii)
 Next ii
Next
```

BIBLIOGRAPHY

Air Force Regulation 55-47, Air Refueling Management

Airlift/Tanker Quarterly, Volume 8, Number 3: 10-14, Summer 2000

Airlift/Tanker Quarterly, Volume 8, Number 4: 26, Fall 2000

Airlift/Tanker Quarterly, Volume 6, Number 1: 19, Winter 2000

Ben-Daya, M. and Al-Fawzan, M. "A Tabu Search Approach for the Flow Shop Problem," ORSA Journal on Operations Research Society, Vol. 109: pp. 88-95, 1998

Battiti, R. "Reactive search: Toward self-tuning heuristics," Modern Heuristic Search Methods, Rayward-Smith (ed.), John Wiley and Sons Ltd: pp. 61-83, 1996

Battiti, R. and G. Tecchiolli. "The reactive tabu search," ORSA Journal on Operations Research Society, 6(2): pp. 126-140, 1994

Battiti, R. and G. Tecchiolli. "Training Neural Nets with the Reactive Tabu Search," IEEE, Transactions on Neural Networks, Vol. 6, No.5, pp. 1185-1200, September 1995

Carlton, W.B. and Barnes, J.W. 'A note on hashing functions and tabu search algorithms', European Journal of Operational Research Vol. 95: pp. 237-239, 1996

Committee on the Next Decade of Operations Research (CONDOR) "Operations Research: The Next Decade," Operations Research, Vol. 36: pp. 619-637, 1988

Congressional Budget Office, "Aerial Tanker Force Modernization," pp. 1-5, March 1982

Congressional Budget Office, "Modernizing the Aerial Tanker Fleet: Prospects for Capacity, timing, and Cost," pp. 2, September 1985

Gant, H.N. "Deterrence in the Face of Explosive Peace," Air University Press 1994

Glover, F. and M. Laguna. Tabu Search. Boston: Kluwer Academic Publishers, 1997

Glover, F. "Future Paths for Integer Programming and Links to Artificial Intelligence," Computers and Operations Research, Vol. 13: pp. 533-539, 1986

Glover, F. "Tabu Search: A Tutorial," Interfaces, Vol. 20: pp. 74-94, 1990

Hostler, H.C. "Air Refueling Tanker Scheduling," Thesis, AFIT/ENS, 1987

Logicon, "Combined Mating and Ranging Planning System Overview," Briefing, 1996

Lokketangen, A. and Glover, F. "Solving Zero-One Mixed Integer Programming Problems Using Tabu Search," European Journal of Operational Research, Vol. 106, pp. 624-658, 1998

Moore, J.T. and Hill, R.R. "Applications of Metaheuristics to Air Force Problems," Proposal, 2000

NASA Technical Memorandum, "Crew Factors in Flight Operations," 103852, December 1991

Pinedo, M. "Scheduling Theory, Algorithms, and Systems," Prentice Hall, 1995

Silver, E.A., Vidal R.V., and Werra D. "A tutorial on heuristic methods," European Journal of Operational Research, Vol. 5, pp. 153-162, 1980

Skorin-Kapov, J. "Tabu search applied to the Quadratic Assignment Problem," ORSA, Journal on Computing, Vol. 2, No.1 (May), pp. 33-45, 1990

Winston, W.L. "Operations Research Applications and Algorithms," third edition, Duxbury Press, International Thomson Publishing, pp. 778-781, 1994

Woodruff, D.L. and Zemel, E. "Hashing Vectors for Tabu Search," Annals of Operations Research, Vol. 41: pp. 123-137, 1993

Zanakis, S.H. and Evans, J.R. "Heuristic 'Optimization': Why, When, And How To Use It," Interfaces, Vol. 11, 1981

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
20-03-2001	Master's Thesis	Jun 2000 - Feb 2001

4. TITLE AND SUBTITLE
A REACTIVE TABU SEARCH METAHEURISTIC EXTENSION OF THE AIR REFUELING TANKER ASSIGNMENT PROBLEM

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Umit Hilmi TEKELIOGLU, 1st Lieutenant, TUAF

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENS)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/O1M-16

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ AMC/XPY
402 Scott Drive, Unit 3L3
Scott AFB, IL 62225/5307
DSN:576-5954

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Combined Mating and Ranging Planning System (CMARPS) is the system used by AMC to schedule air refueling for deployments from the continental US to other parts of the world. The tool developed by Capehart (2000) got results similar to CMARPS in less time. Capehart's tool allows AMC to input several receiver groups consisting of various aircraft types and numbers. Each receiver group contains a point of origin and destination, with the option of providing one waypoint along the path, a ready to load date (RLD) and required delivery date (RDD). The user is also able to specify the locations of military tanker aircraft. The main goal of this tool is to assign the tankers to the different refueling points of the receiver groups so that all receiver groups arrive before their RDD. Secondary goals include the reuse of tankers and limiting the total flight distance for all tanker aircraft. The main purpose of this research is to introduce a dynamic feature of tabu search, reactive tabu search, into the tool. This method changes tabu tenure when necessary in the hope of finding better solutions by diversifying the search to the unexplored areas of the solution space.

15. SUBJECT TERMS
Tabu Search, Heuristics, Metaheuristics, Tanker Scheduling, Assignment Problem

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. James T. Moore, AFIT/ENS JamesMoore@afit.af.mil
U	U	U	UU	113	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4337

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Form Approved
OMB No. 074-0188