A Tabu Search for Scheduling and Rescheduling Combat Aircraft

Kevin M. Calhoun

A TABU SEARCH FOR SCHEDULING AND RESCHEDULING COMBAT AIRCRAFT

THESIS

Kevin M. Calhoun, Captain, USAF

AFIT/GOR/ENS/00M-6

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503			
1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** March 2000	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A TABU SEARCH FOR SCHEDULING AND RESCHEDULING COMBAT AIRCRAFT			**5. FUNDING NUMBERS** EN – 00 - 186
6. AUTHOR(S) Kevin M. Calhoun, Captain, USAF			
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 P Street, Building 640 WPAFB OH 45433-7765			**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GOR/ENS/00M-6
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NI Attn: Mr. John McMann 4040 Fairfax Drive Suite 500 Arlington, VA 22203-1613 DSN: 426-9583			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** 99 - 045
11. SUPPLEMENTARY NOTES Prof. D. F. Deckro, ENS, DSN: 785-6565, ext. 4325 e-mail: richard.deckro@afit.af.mil			
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			**12b. DISTRIBUTION CODE** A

ABSTRACT (*Maximum 200 Words*)

Scheduling an air campaign is time and labor intensive. Exacerbating the problem, combat planners use manual methods to accomplish much of this daunting but critical task. While some work has been done to automate the process, the approaches used generate schedules that must undergo major modifications before they are "flyable." Planners, therefore, distrust the results and use the automated features of the software sparingly.

Additional problems arise due to the lag time between the publication of the Air Tasking Order (ATO) and the start of the ATO day. During this time, conditions change in the dynamic battlespace (aircraft break, runways are damaged, etc.). Current execution software performs a validation check against current conditions, but yields no replanning options.

This research explores using heuristics to determine good solutions for the initial air campaign plan and extends previous work by including air-tasking priorities. Additionally, the heuristics are adapted to generate replanning options for the ATO Execution Managers. Java was used for portability and for object reusability elsewhere in the planning hierarchy. Furthermore, the method may be applied to other areas in commercial, government, and military organizations. The heuristic can be modified for use in any enterprise where re-scheduling is common

14. SUBJECT TERMS ATO, Replanning, Project Scheduling, Resource Constrained Project Scheduling, Generalized Precedence Constraints, Multi-modal, Multi-modal Generalized Resource Constrained Project Scheduling, Critical Path Method, Re-scheduling, CTAPS, Prioritized Targets, Target Nomination List, Java, Heuristics, Tabu Search, SQL, Goal Programming			**15. NUMBER OF PAGES** 135 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

AFIT/ENS/GOR/00M-06

A TABU SEARCH FOR SCHEDULING AND RESCHEDULING COMBAT AIRCRAFT

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

Kevin M. Calhoun, B.S., M.Ed.

Captain, USAF

March, 2000

AFIT/GOR/ENS/00M-6

A TABU SEARCH FOR SCHEDULING AND RESCHEDULING COMBAT AIRCRAFT

Kevin M. Calhoun, B.S., M.Ed.
Captain, USAF

Approved:

[illegible]	20 Mar 00
Richard F. Deckro (Chairman)	date
[illegible]	20 Mar 00
James W. Chrissis (Member)	date
James T. Moore	20 Mar-00
James T. Moore (Member)	date
John C. Van Hove	15 MAR 00
John C. Van Hove (Member)	date

Acknowledgments

My gratitude is extended to the following people for their support, encouragement, and patience throughout my AFIT experience. To my Lord and Savior Jesus Christ, without whom nothing would be possible. To my wife, Karen, for taking care of my family, for taking up the slack during the countless nights I've been at school, and for loving me no matter how grumpy I got. To my advisor, Dr. Richard F. Deckro, for never giving up on me and for never just giving me the easy answer. To my committee, Dr. James T. Moore, Dr. James W. Chrissis, and Dr. (Capt.) John Van Hove, for their help and understanding along the way. Recognition is also in order for Lt. Chris Cullenbine, who went beyond the call of duty in his efforts to help a friend and fellow student. I wish to acknowledge the work of Lt. Robert Harder, whose Java savvy was useful in the extreme. Finally, I'd like to thank the rest of my fellow students who all have inspired me in some way. May God bless you all.

Kevin M. Calhoun

Table of Contents

List of Figures

List of Tables

ABSTRACT

Scheduling an air campaign is a time and labor intensive task. Exacerbating this task is the fact that combat planners still make use of manual methods to accomplish much of this daunting but critical effort. While some work has been done to automate the process, the heuristics used generate schedules that must undergo major modifications before they are "flyable." Combat planners, therefore, distrust the results and use the automated features of the software sparingly.

An additional problem arises due to the lag time between the publication of the Air Tasking Order (ATO) and the start of the ATO day. During this time period conditions in the dynamic battlespace can, and often do, change (aircraft break, runways may be damaged, close air support requests come in, for example). The execution software currently in use will do a validation check against the current conditions, but yields no options for replanning.

This research explores the use of tabu search (TS) to determine "good" solutions for the initial air campaign plan. It extends previous work by including air-tasking priorities. Additionally, this effort adapts the TS to focus on generating replanning and re-scheduling options for the ATO Execution Managers (or Combat Operations Division Duty Officers). The TS procedure was implemented in Java to be portable and to make the objects available for reuse and adaptation elsewhere in the planning hierarchy. Furthermore, the method that was developed in this research can be applied to other areas in commercial, government, and military organizations. Re-scheduling is a critical problem and the TS developed in this research can be modified for use in any enterprise where re-scheduling is common.

CHAPTER 1. INTRODUCTION

Even in today's technological, computer-driven environment, much of air battle planning is done manually. Despite this, US Joint and Combined Air Operations have accomplished amazing results. If the pace and size of the air battle increase, while the size of our force and tail numbers decrease, planners may have increasing difficulty meeting all the projected demands for air power. Even though some attempts have been made to automate portions of air campaign planning, the results have not been completely satisfactory. The planning process can be improved. The current Air Tasking Order (ATO) (a list of acronyms is provided in Appendix A) cycle is 72 hours, 48 of which are claimed by the planning phase, with the other 24 hours consumed by the execution phase [AFI13-1AOCV3, pp. 38-39, 1999]. Although much of this time is spent analyzing intelligence, reaching agreement on the Joint Integrated Prioritized Target List (JIPTL) and other necessary time-consuming tasks, there are aspects of the ATO process that can, and should, be streamlined through the use of computer resources. Automated support of the planning process would serve two vital objectives: 1) significantly shorten the ATO cycle [Gonzales, p. 28, 1996] and, 2) improve the solutions generated in the process by providing a list of good solutions that planners could use as a base.

1.1. Background and Purpose

During military operations such as Operation Desert Storm, the personnel working in the Guidance, Apportionment and Targeting (GAT) cell translate the Joint Forces Air Component Commander's (JFACC) guidance into the Master Air Attack Plan (MAAP). The MAAP contains mission information such as target data and weaponeering information. The MAAP is then passed on to the Combat Plans Division (CPD) where detailed mission information such as number and type of aircraft, mission numbers, and coordination are developed using the Advanced Planning System (APS) in the Contingency Theater Automated Planning System (CTAPS). Finally, the ATO is published and transmitted a minimum of 10 hours before the start of the effective period of the ATO [AFI13-1AOCV3, p. 39, 1999].

The units therefore should have at least that same 10-hour period for mission planning and preparation. Generally, however, that has not been the case. Often, missions are planned several hours before the actual production of the ATO. When the ATO is published, there are certain missions that may no longer be flyable. The execution management software, the Computer Assisted Force Management System for X-windows (CAFMSX), performs a validation check against current conditions and generates a text file containing all of the unflyable missions with the constraints that they have violated. The combat operations duty officers (DOs) must then manually replan these missions and targets. There are currently no software tools in the field that offer automated replanning options for the DOs [JAOSC Course Handout, pp. 8E2L1 7-11, 1998].

Units charged with interdiction missions are particularly susceptible to the consequences of target and timing changes. The GAT cell frequently makes changes to the current ATO, often at the last minute. The GAT, in trying to adjust to continually changing battlespace conditions, view these last-minute changes as the best way to maximize the available sorties to strike higher-priority targets. However, pilots of the affected aircraft are generally of the opinion that the quantity and late timing of the changes resulted in reduced effectiveness and increased vulnerability to threats and fratricide [Haas, p.30, 1998, Cohen, p.230-232, 1993]. Missions must be rescheduled and re-coordinated with their packages, tankers and other support sorties. Again, this re-scheduling task has been primarily accomplished through manual means.

Gonzales, in a study for the RAND Corporation, suggests two ways to make the planning cycle more responsive [Gonzales, p.25-27, 1996]. First, shorten the planning cycle. Second, allow changes to the plan only at designated points in the cycle time line (so-called *time fences* in industrial production scheduling). A combination of these two suggestions appears to be the best overall choice. This research focuses on the first suggestion, shortening the planning cycle. RAND's stated goal was to ultimately reduce the planning cycle from 48 hours to 24 hours, thus allowing changes to be incorporated into the next day's ATO. In this way, an attack strategy would still be responsive and the number of last-minute changes would be minimized. If such a reduction could be accomplished, it would only be necessary to carry out one planning process each day (tomorrow's) instead of the two parallel processes (tomorrow and the day after) required under the current system. The research in this thesis is a step in that direction.

Gonzales estimated that MAAP production time could be reduced from 11 hours to five hours with automation [Gonzales, p.28, 1996]. A start has been made in this area with the advent of the JFACC Planning Tool (JPT) which uses the Conventional Targeting and Effectiveness Model (CTEM) to help develop the draft MAAP. Two Numbered Air Forces (NAFs) and the Air Ground Operations School employ JPT [Abt, pp.15-17, 1997]. Current AF doctrine, however, calls for MAAP production to be done manually [JAOSC Course Handout, p. 3G 5-7, 1998].

As previously stated, the output from the MAAP (Target Planning Worksheets or TPWs) is given to the CPD where the Advanced Planning System (APS) is used to develop the mission details. APS possesses the ability to auto-plan combat missions. However, since the greedy heuristic used for this function in APS is only marginally successful [Van Hove, p. 4, 1998], planners generally opt to plan the missions themselves. An example of the APS auto-plan method is given in Chapter 2. This thesis, therefore, is concerned with improving the automated solutions obtained when building the mission details from the TPWs and on automating replanning options. A heuristic is developed to obtain an initial feasible solution for the air combat scheduling problem which is then improved upon by using a technique called tabu search (TS).

1.2. Scope

The scope of this research is limited to scheduling the attack aircraft for suppression of enemy air defenses (SEAD) and interdiction (INT). Operationally, each of these missions may also require support such as escorts (ESC missions), electronic counter-measures (ECM missions), airborne command, control, and communications (ABCCC), and aerial refueling (AR). Other direct attack mission types such as close air support (CAS) and offensive and defensive counter-air (OCA and DCA) are not directly addressed in this research; however, the procedure developed herein may be extended to include these missions.

This research focuses on air campaign planning and replanning. This overall problem can be formulated as a project scheduling problem (PSP). In order to obtain an initial solution to the problem, a heuristic is developed based on key concepts and heuristics from scheduling theory. A tabu search (TS) procedure that uses the result from the initial solution heuristic and improves upon it is used to solve the PSP. The TS procedure is tested by solving similar PSP problems with known optimal solutions. Once tested, the TS is applied to the combat planning and replanning problem; it generates options for combat planners and operations duty officers to schedule or re-schedule attack assets to targets. A method to re-schedule *ad hoc* combat missions during the ATO day is then developed.

A distinction is made between replanning and re-scheduling. Replanning, as defined in this thesis, refers to that aspect of combat operations concerned with "fixing" the schedule before the start of the flying day. Re-scheduling refers to re-assigning missions to targets, as circumstances warrant, during the execution of the Air Battle Plan (ABP) (i.e. during the flying day).

Since scheduling air assets against targets is similar to project planning, the methodology developed here applies, in general, to program management as it relates to project scheduling. Of particular interest in the industrial sector, though, is the technique developed for re-scheduling projects; a critical operational concern to which little previous work has been directly applied.

1.3. Objectives

The TS procedure developed for this research takes advantage of the structure of the force-level air mission planning, replanning, and re-scheduling problems. This research represents a two-pronged approach. First, it continues work done by Van Hove and Koewler in finding better solutions to the initial ABP by applying a TS procedure to the problem. The research then extends the effort of previous research by capturing target priorities through the use of a goal programming (GP) model.

Secondly, a TS procedure for replanning is developed. Replanning has two objectives. The first objective concentrates on the replanning that must be done prior to the start of the ATO day. It suggests new ABPs based on the resources, assets and targets currently available. In many respects the TS for replanning is similar to that used to generate the initial plan. The second objective was to modify the TS to perform quick re-scheduling of missions or packages during the execution of the ABP. For both objectives, this approach applied a lexicographic goal programming formulation which maximizes the number of high priority targets that are scheduled and then minimizes the number of missions to be re-scheduled.

In all cases, whether planning, replanning or re-scheduling, the TS generates a list of options for the planner to review. The planner may then select the best solution for the current operational situation from the list. This process uses the options developed by the heuristic, coupled with the decision maker's unquantifiable experience and knowledge, to develop a plan. It is acknowledged that the methodology developed here is simply an aid to be used by, not a replacement for, experienced combat planners. No computer tool can incorporate in its approach the non-quantifiable insight or intuition of an experienced human combat planner. The goal of this research was to rapidly provide improved solutions to support the ultimate decision-maker.

1.4. Assumptions

In order to develop the TS procedure in this thesis, the following assumptions were made:

1. Since the missions to be planned came from the Target Planning Worksheets (TPWs) generated by the MAAP, the necessary combat resources (planes, people, fuel, bombs and bullets) are in-theater and available.
2. Scheduling the attack missions first and then forming packages is suitable.
3. There are tankers available to refuel the strike packages as necessary.
4. Airborne Elements of the Theater Air Control System (AETACS) and the Ground Theater Air Control System (GTACS) are in place and functioning.
5. The duration of a mission can be assigned an approximate value based on aircraft fuel burn rates, flight profile, munitions carried (Standard Conventional Load, SCL) and base and target locations. Mission duration consists of the sum of the travel time of the outbound leg and the inbound leg, plus the turn time.
6. The airfield can accommodate the solution options generated.
7. If a choice must be made between a high priority target and a lower priority target, the high priority target must be scheduled.
8. When finding a re-scheduling solution, a *minimum* of six hours notice must be given to the pilot and support personnel for mission planning and re-arming [Cohen,1993].
9. Sortie allocations, SCL, target priorities, and probability of kill calculations have been specified in the MAAP.
10. The combat airspace has already been deconflicted.
11. A target that must be attacked before other targets (a predecessor) must be of the same target priority classification, or higher.

1.5. Approach

To best exploit the advantages of object-oriented programming, the heuristic procedures developed in this thesis are written in Java. This makes the procedure *portable* so that it can run on any platform—PC, Sun, Silicon Graphics, and so forth. Although the workstations in the Aerospace Operation Center (AOC) have been primarily Unix (Sun UltraSPARCs), a push is underway to migrate to PCs (Windows NT) [TBMCS JPMR, 1997].

The data necessary to build an ABP is stored in the Air Campaign Database (ACDB), an Oracle database, for this reason the procedures make Structured Query Language (SQL) calls to the ACDB (see Appendix B for more on SQL). The use of ANSI-compliant SQL augments the portability of these procedures. These queries pull only the attributes from different tables in the ACDB that are necessary to solve the planning problem (the ATO contains only a small subset of the information in the ACDB). For this study, the desired data is stored in a Microsoft Access database for ease of implementation on a PC. Since an Access database can be easily converted to an Oracle database, the transition to an AOC-like environment (UltraSPARC) can be made readily.

As noted earlier, the TS procedure results are compared for accuracy to the lower bounds of a test set of scheduling problems. CPU processing times for the solutions are reported for each problem to demonstrate how processing times grow with the size of the problem. Inter-procedure comparison of processing times is of limited value because of computer platform. Once the TS procedure was validated against these test cases, it was modified and applied to sample air campaign planning, replanning and re-scheduling problems. The quality of the solutions generated for these problems is measured using heuristic testing procedures developed by Barr, *et al* [Barr, *et al*, 1995].

1.6. The ATO Process

This section will familiarize the reader with the ATO production cycle. It is included in order to give the reader an idea of how complex the process can be so that he or she can appreciate the value of automating part of the procedure. Section 1.7 presents a description of the type of solution to expect from the APS auto-plan feature to reinforce the need for an application that can produce quality solutions to the air campaign planning problem..

The ATO is a written plan for the employment of the air assets of the deployed forces. The lead-time required for development of an ATO is at least 48 hours. ATOs are normally executed during a 24-hour period of time. The start time and end time for a particular ATO can vary and is usually specified in the Operations Order (OPORD). A typical time line for ATO production is shown in Figure 1 while the ATO cycle is represented by Figure 2 where the starting point for a particular ATO is generally considered to be at the top node, Strategy Development. The production of an ATO is a step-by-step process wherein each step must be completed before the next one begins.

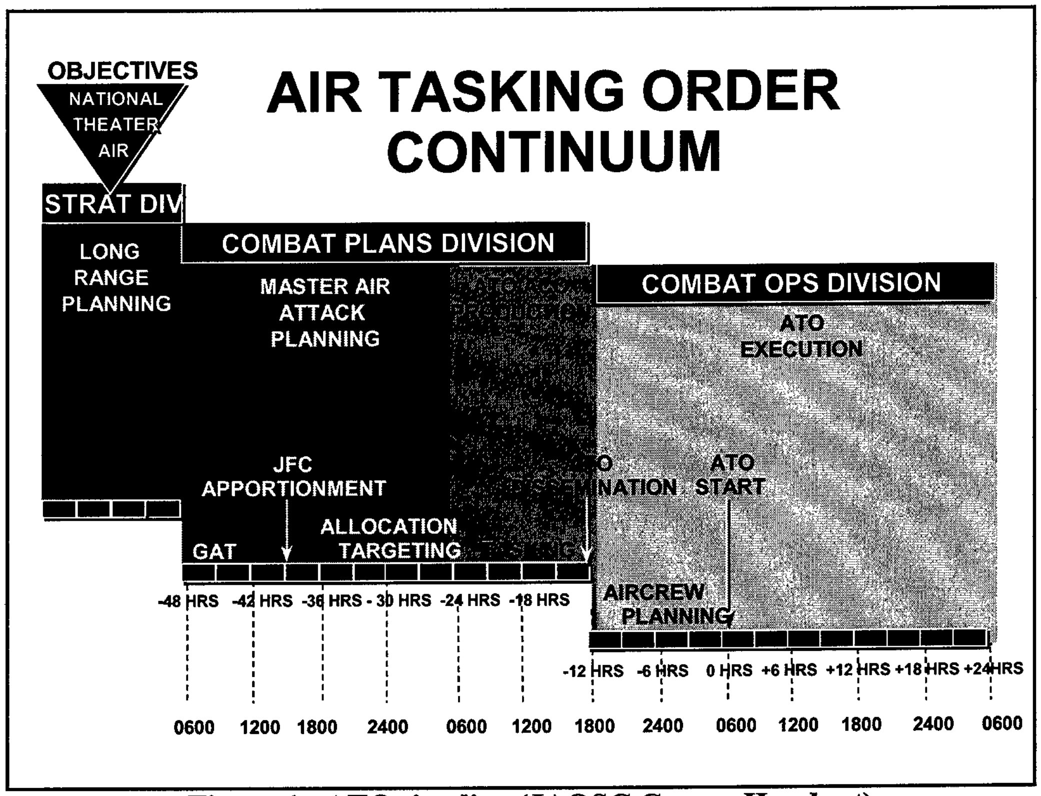


Figure 1. ATO timeline (JAOSC Course Handout)

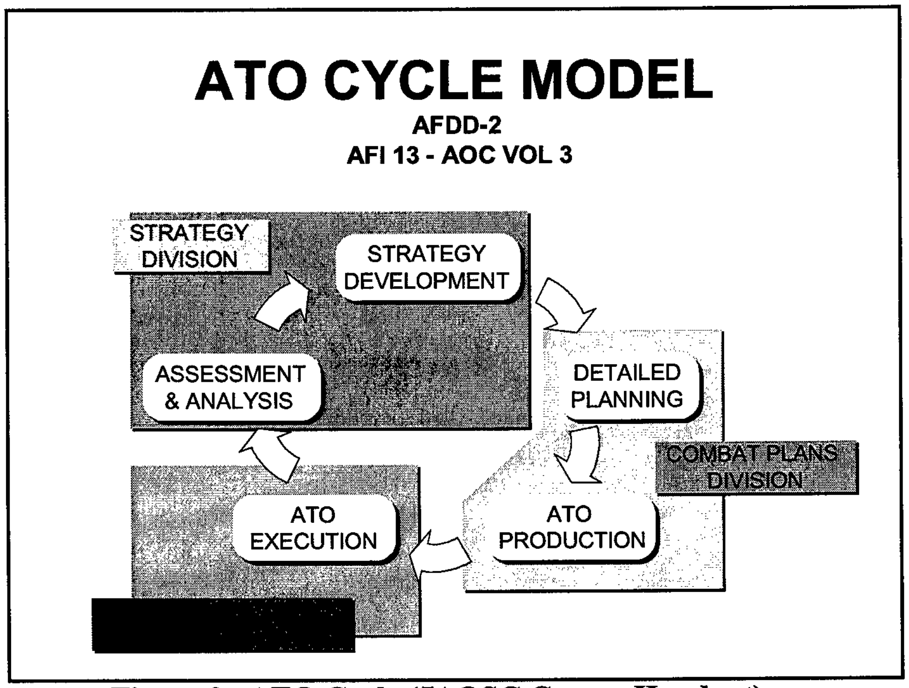


Figure 2. ATO Cycle (JAOSC Course Handout)

1.6.1. Strategic Development

The work of developing an ATO begins in the Strategy Division (SD). The SD is involved in developing a part of an ATO referred to as Strategy Development. The task of the SD is to ensure that the work being done by the rest of the ATO developers is in compliance with the guidelines of the Joint Force Commander (JFC) and the other components (land, naval, air, and special ops). The SD continually assesses how well the air component is achieving the tasks and objectives assigned it by the JFC. They also look at phases in the conflict and timing. Key questions such as the following are considered:

Have we achieved aerospace superiority?
Have we met our strategic attack goals?
Do we need to concentrate on interdiction or close air support (CAS) as the battle develops?

The Contingency Theater Automated Planning System (CTAPS) application called the JFACC Planning Tool (JPT) assists the members of the division in resolving the answers to these questions.

The output of the Strategy Division is specific guidance for target development in the form of specific air tasks with accompanying measures of merit that will be implemented by the planners in the next step of the ATO development. This specific guidance is not developed in a vacuum; it is fully coordinated with the other components and the JFC's staff so that the subsequent work on the ATO is synchronized with, and not in conflict with, the battle plans of the other components.

1.6.2. Detailed Planning

Detailed Planning (DP) follows Strategic Development in the planning cycle and contains three major phases: Guidance, Apportionment, and Targeting (GAT) development, Weaponeering and Allocation, and the development of the Master Air Attack Plan (MAAP). Members of the Combat Plans Division perform DP. Initial steps in the planning process are the responsibility of the MAAP Core Team. Combat Plans is organized into teams of workers with each team adding to the ATO as it makes its way through the process towards dissemination to the units who ultimately execute it.

The first phase of Detailed Planning is the development of Guidance, Apportionment, and Targeting by members of the MAAP Core Team. An understanding of target generation is necessary to fully understand the work done by the GAT planners. The service components each derive their operational objectives from the JFC's objectives. Air component planners use the JFACC Planning Tool (JPT) to match up potential targets with national, strategic, and tactical objectives. The planners then examine the target database. Generally the target database for the Joint Task Force's (JTF) area of responsibility (AOR) is a subset of the target database for the entire theater. The JFC's intelligence staff normally maintains the database for the JTF AOR. This database contains all possible targets that may be attacked in the AOR.

Representatives from each service component examine the database with their operational objectives in mind. This allows for the identification of specific types of targets in the database and target sets that, if attacked and destroyed or damaged, would allow the component to achieve its objectives. Next, each component examines the target sets that it must attack or destroy and verifies that it has the means to do so.

The GAT planners normally meet from 0600 to 1200 hours to begin constructing the ATO. All components produce their nominations for ATO targeting which are then grouped into categories of missions: strategic attack (SA), interdiction (INT), counter air

(CA) or offensive counter air (OCA), and close air support (CAS). These lists of targets, called *candidate target lists* (CTL), are then prioritized.

During prioritization, members of each service component explain why their targets are important and deserve to be placed high on the list. Ultimately, however, the GAT inspects the CTLs and determines priorities based upon the guidance of the JFC. Reviewing and ordering the lists normally takes most of the morning.

The results of the GAT are then briefed to the JFACC and, if accepted, go on to the JFC for final approval. The targets, once approved, become the JFC's Joint Integrated Prioritized Target List (JIPTL).

The next step in the cycle is force application and weaponeering. This is the process of determining the quantity and type of lethal or non-lethal weapons required to achieve a specific level of damage to a given target. The weaponeers consider target vulnerability, weapons effect, munitions delivery accuracy, damage criteria, probability of kill, weapons reliability, and operational capabilities and limitations [AFI 13-1AOCV3, p. 67, 1999]. The end result of the weaponeers' work is a Target Planning Worksheet (TPW) for each target on the Target Nomination List (TNL) that is produced from CTAPS. The TPW identifies the desired mean point of impact (DMPI), its measured location, and as many aircraft and weapons options as possible. The TPWs are then forwarded to the planners who build the MAAP.

The other components notify the JFACC about the number and type of sorties they will make available for JFACC tasking. All of the JFACC's sorties are then "allocated." It is the management of these additional available sorties that is one of the jobs of the JFACC. Any naval air assets that are not required for naval missions such as sea surveillance, anti-submarine warfare, counter-shipping, fleet defense, and so on, are made available to the JFACC for tasking. Through this practice, aerospace power is most effectively employed and the use of valued assets is kept at a high level.

Finally, the planners stabilize on an effective number of targets in each mission category that can be struck with the available assets. The draft plan is then re-examined to determine if it will have the effect on the battlefield that the JFC intended when he apportioned the missions. The list of targets included in this draft plan is called the Target Nomination List (TNL).

Note that units have not been assigned to specific missions. This is an initial allocation, not a final allocation. In the Master Air Attack Planning process, assets often change missions from the original allocation plan for a variety of reasons. Only after the MAAP has been developed has the final allocation of aircraft been established. At that point a SORTIEALOT (sortie allotment) message is transmitted that notifies the components and units about the employment of their assets. CAS planning, a separate but related issue, is not addressed in this thesis.

The next step in the MAAP process is the most complex stage of ATO development. The MAAP is the plan that contains key information that forms the foundation of the joint ATO. The purpose of the MAAP is to translate the approved JIPTL into a specific air attack plan that serves as the basis for the ATO [Joint Pub 3-56.1, 1994]. Master air attack planning typically lasts from 1800 to 2100 until 0600 hours. For this reason, it is often referred to as the "Night Targeting Cell" or simply the "Night GAT".

Coordination is the key to putting together an effective plan. The development of a package to hit OCA/INT targets will generate a requirement for suppressing enemy air defenses (SEAD missions), airspace that may need to be deconflicted and defensive counter air (DCA) for escort. For example, Reconnaissance (RECCE) and Special Operations may want to attach missions to the package to take advantage of the SEAD and mutual support.

There is no correlation between the priority of targets and the sequence of the packages. The first package may hit relatively low priority targets while the highest priority targets may not be struck until late in the day. Note that at this time, specific Times On Target (TOT) have not yet been determined. These times are determined by developing a spreadsheet that tracks aircraft from different bases as they turn throughout the day. As the spreadsheet is developed, planners must bear in mind that aircraft have varying distances to travel to get to the target area. A map with the proposed airspace for the day will show the proposed routing aircraft will take to reach their target area. Additionally, aircraft travel at different speeds, have different loiter times in the target area or on-station, and have different turn-around times after landing. A notional example of such a flowchart or spreadsheet is given in Figure 3. This results in the specific TOTs that are assigned to the targets.

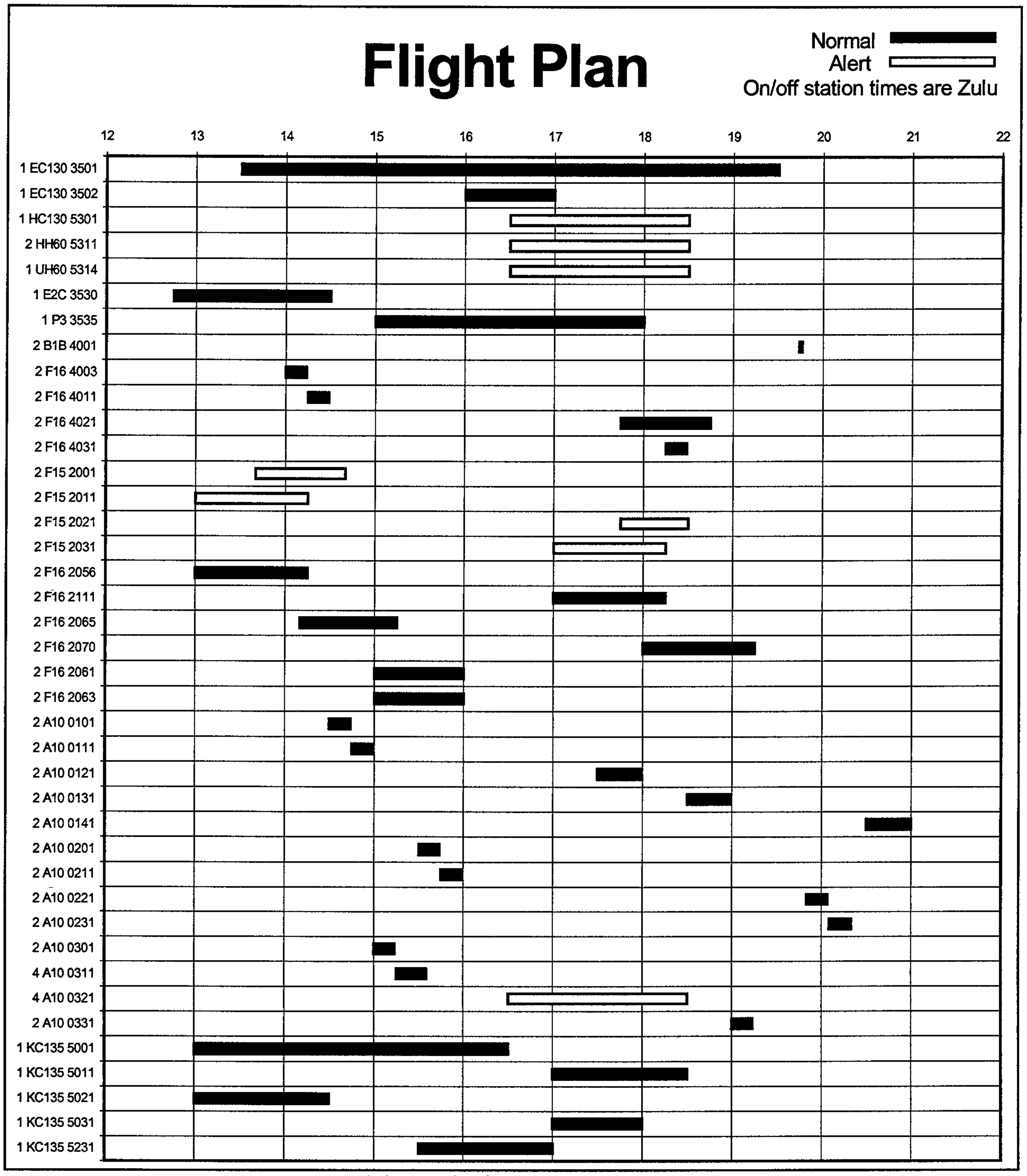


Figure 3. Sortie Flow

Finally, the planners must coordinate their missions with other elements of the MAAP team. SEAD, DCA, airspace, and placement of surveillance aircraft must all be coordinated. Master Air Attack Planning requires a great deal of coordination. In addition to the MAAP for INT, OCA, SA, and CAS, and the Airspace Plan for the daily ATO, several other plans are fleshed out during master air attack planning. The air defense plan, communications plan, tanker plan, and command and control plan, to name a few, are finalized at this time.

It is important to note that as master air attack planning ends, all of these plans exist primarily on paper. In the next phase, ATO Production, these plans are put into CTAPS and detail added to allow mission planning by the tasked units.

1.6.3. ATO Production

The main tool used during ATO Development is the CTAPS Advanced Planning System (APS). All data generated during Master Air Attack Planning is input to APS. At this time units are actually tasked with specific missions. Call signs and identification friend or foe (IFF) codes are assigned to missions as well. Specific tankers are paired with aircraft that require refueling in particular tanker tracks at explicit times. Normally the ATO Development process takes approximately 12 hours, typically beginning 24 hours prior to the execution of the ATO.

Once all missions are coordinated, the APS Air Battle Plan (ABP) is sent to the Computer Assisted Force Management System (CAFMS) in Combat Operations. It is placed into the United States Message Text Format (USMTF) for an ATO. It is then disseminated to the appropriate units. Normally the units are given 10 hours for crew planning prior to the first time on target that signals the start of ATO execution [AFI 13-1AOCV3, 1999].

1.6.4. ATO Execution

As the ABP travels from APS to CAFMS, it leaves the Combat Plans Division and arrives in the Combat Operations Division. Personnel in Combat Operations are responsible for monitoring the execution of the ATO during the phase of the ATO cycle called ATO Execution.

The ATO Execution cycle begins, typically about 0600, with the following inputs: the ATO, Airspace Control Order (ACO), and weather, intelligence, and friendly force status briefings. Wing Operations Centers (WOCs) will inform Combat Operations of takeoff and landing times as they occur, and mission success as it becomes known. Personnel in Combat Ops monitor intelligence from all sources and stay in close touch with liaisons from other services as well as allies and coalition partners.

At this point, if everything went exactly according to plan, Combat Ops would have little to do—the ATO would simply be flown as scheduled. However, changes in the enemy and friendly situations dictate changes to the ATO. Friendly airbases might be attacked, or the weather may worsen. As a result, Combat Operations generates multiple changes to the ATO and ACO, launches aircraft from alert onto immediate tasked missions, and diverts aircraft from their preplanned missions onto *ad hoc* missions when very high priority targets appear on the battlefield. Personnel in Combat Operations constantly modify and change the plan developed in Combat Plans to "optimize" the effect of aerospace power on the battlefield.

1.7. Advanced Planning System

APS assigns squadrons to targets in a *greedy* fashion; it regards the initial assignment of aircraft to targets as a set-covering problem [Van Hove, p. 4, 1998]. At each decision point, a greedy heuristic assigns the best available choice. APS simply examines each target, in priority order, assigns the best available resources (nearest squadron possessing appropriate assets) to it, builds a mission with that resource/target pairing and schedules the mission at the first available time in the ATO day. APS allows the user to define what makes a resource "best;" it may be maximize PK or minimize refueling requirements, for example. For the sample problem in the following paragraph, "best" is defined in terms of proximity. If a necessary resource for a particular target is not available, no mission is planned for that target. This approach was selected for APS because it takes less time than optimal-seeking methods [Van Hove, p. 4, 1998]. It is unlikely that the proposed missions represent all considerations in the allocation of available resources; hence the need for combat planners to make major adjustments to the proposed schedule in order to ensure all targets are assigned appropriate attack missions.

Consider the following simple problem illustrated by Figure 4, with target requirements and available base assets contained in Tables 1 and 2. Note that this straightforward example does not consider priorities or precedence relationships.

Table 1. Sample Problem: Target Requirements

Target	Aircraft Type	# of Assets	Aircraft Type	# of Assets	Aircraft Type	# of Assets
1	Aircraft 2	4				
2	Aircraft 2	2	Aircraft 1	4	Aircraft 3	4
3	Aircraft 2	4	Aircraft 1	6		

Table 2. Sample Problem: Base Assets

Base	Aircraft Type	Available	Aircraft Type	Available	Aircraft Type	Available
1	Aircraft 2	5	Aircraft 1	8		
2	Aircraft 2	2	Aircraft 1	4	Aircraft 3	4

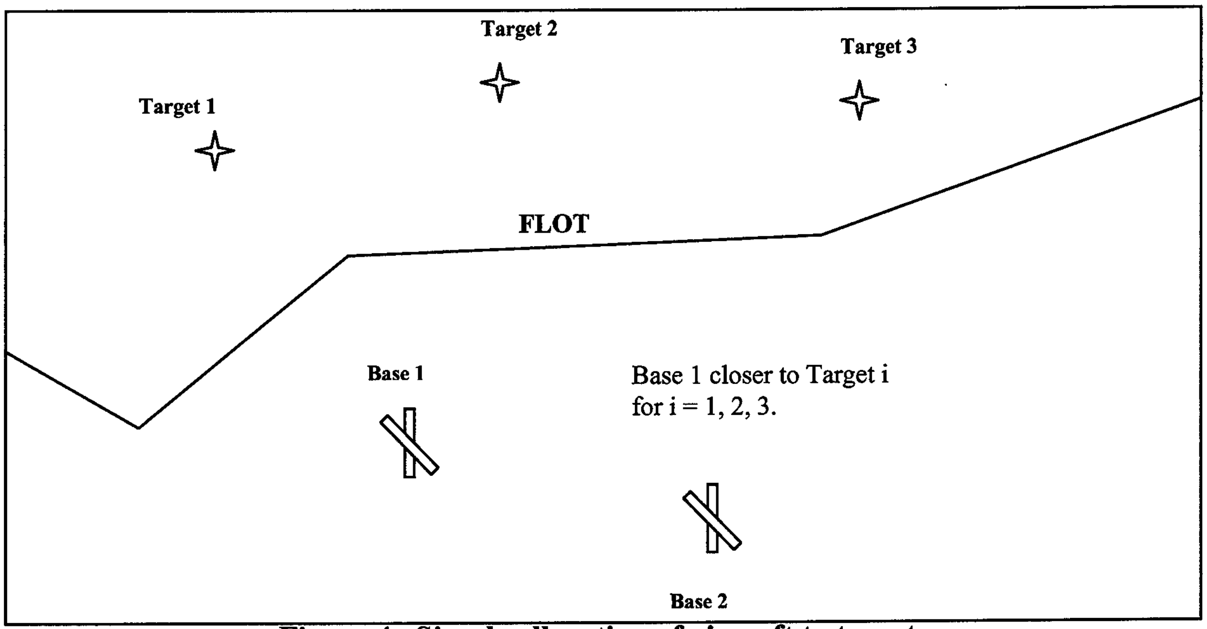


Figure 4. Simple allocation of aircraft to targets

If proximity is defined as "best," APS's greedy heuristic would schedule four of Aircraft Type 2 from Base 1 against Target 1 and then schedule four of Aircraft Type 1 from Base 1 against Target 2. When APS considers Target 3, there are not enough assets at either base to schedule against it. Hence, no mission would be scheduled against Target 3. An obvious solution to this small example would be to schedule four of Aircraft Type 3 from Base 2 against Target 2, six of Aircraft Type 1 from Base 1 to Target 3, and four of Aircraft Type 2 from Base 1 to Target 1. Another solution would schedule two Aircraft Type 2 from Base 2 to Target 2, four Aircraft Type 2 from Base 1 to Target 1, and six Aircraft Type 1 from Base 1 to Target 3. While this scheduling problem was simple enough to fix by inspection, the array of choices encountered with hundreds of targets and thousands of sorties can become complex. Additionally, any application that attempts to solve this scheduling problem must be work quickly, given the number of missions that must be scheduled.

1.8. Summary

This chapter introduced the problem addressed in this research and the approach that was used to solve it. Chapter 2 presents a review of the literature pertaining to project management as it relates to scheduling. A summary of goal programming is followed by an introduction to TS concepts. Chapter 3 details the development of the TS procedure used to solve the GP models for air combat planning, replanning and re-scheduling problems. Chapter 4 presents the results of using the TS procedure in a case study that demonstrates the utility and unique capabilities that come with using a TS to explore the solution space of a large problem. Chapter 5 provides a summary of the research, the significant contributions to both military and civilian enterprises, and recommendations for future work.

CHAPTER 2. LITERATURE REVIEW

This research presents a fast but effective approach for assigning allocated attack aircraft to targets. Before describing the methodology employed, a review of the pertinent literature is appropriate. Selected topics from scheduling theory are presented to lay a foundation. The application developed in this research uses a goal programming (GP) model similar to that in the Conventional Targeting Effectiveness Model (CTEM), therefore a description of GP appears. The search method employed to solve the resulting GP, tabu search (TS), is reviewed next. Finally, to enhance the portability of the application developed in this research, the TS code is written in Java. Accordingly, a section on object-oriented programming in general and Java in particular rounds out the chapter. The chapter ends with a summary of the key points from the literature.

2.1. Scheduling Theory

This section introduces concepts from scheduling theory, especially as it relates to project management and the targeting process. "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, p.1, 1995]." The scheduling process exists in virtually all operational settings. It is key in the industrial sector with such items as utilization of manufacturing and production systems to information-processing systems. It is also spread throughout transportation and distribution systems [Pinedo, p.1, 1995]. Scheduling is widely used in military settings such as weapon system development (acquisition) and flight scheduling.

The resources, tasks, and objectives that make up a schedule may take on a variety of forms. *Resources* may be machines in a workshop, runways at an airport, aircraft in a squadron, and so forth. *Tasks* may be operations on an assembly line, stages in a construction project, or attacking targets on a Target Nomination List (TNL) for example. *Objectives* include the minimization of the completion time of the last job (makespan), minimization of the maximum tardiness (worst violation of the due dates), and minimization of the total number of late tasks, to name a few [Pinedo, p.1, 1995].

A *project* is a systematic enterprise designed to accomplish some specific non-routine or low-volume task [Shtub, et al, p.1, 1994]. For the purpose of this research, a project is a finite set of activities that must be scheduled in agreement with certain precedence requirements between activity pairs and with limited resources. Project management, then, is the process of planning, scheduling, and overseeing the activities of a project. In the context of this thesis, the project is the air tasking order [Van Hove, 1998, Koewler, 1999].

This section is intended to accomplish several goals. First, it introduces the reader to key concepts from scheduling theory. Secondly, it lays the foundation for a heuristic (described in Chapter 3) to obtain an initial solution for the ABP. This is done by drawing analogies between air combat planning, project scheduling, and job shops. This initial solution is then used by the TS as a starting point to search the solution space for better solutions. Third, the section reviews the integer linear programming (ILP) model developed by Van Hove to produce an optimal solution to the air combat-planning problem. This model is introduced to demonstrate the vastness of the problem in terms of the number of variables and constraints. The problem can take hours to formulate and solve [Van Hove, p.142, 1998] for a mathematically optimal solution, justifying the need for a quicker heuristic technique, albeit with the loss of guaranteed optimality.

2.1.1 Gantt Charts

One of the most widely used project management tools, the Gantt chart, is a bar chart that graphically represents the relationship of activities over time [Shtub, *et al*, p.302, 1994]. Normally, each resource on the vertical axis is unique, for example if a problem had three machines, the Gantt chart would have three rows of activities and a different machine would service each row. Examination of a single row gives users an intuitive feel for the resource usage for that particular machine.

Figure 3, Section 1.6.3, uses specialized Gantt charts to display sortie flow. Each individual bar represents a mission, where the horizontal axis represents time and the vertical axis shows the aircraft involved in each mission. Here, however, the same resources might fly missions represented in different rows. APS has the capability to generate the sortie flow for an ATO. The sortie flow resembles the grease boards and spreadsheets that planners use today for scheduling missions.

2.1.2 Parallel Machine Models

A machine can be thought of as a finite resource required for completing a task, such as a drill press in a job shop, cashiers in a checkout line, or attack aircraft in theater. In scheduling theory, the simplest model is that of the single machine. Analysis of single machine models has led to heuristics for more complicated machine environments [Pinedo, p.26, 1995]. However, in most real-world settings the occurrence of resources in parallel is common [Pinedo, p.61, 1995]. When parallel machines are present, job *j* requiring processing on a single machine, may be processed on any one of the machines in the shop. For example, if the job is to destroy a bridge, it can be accomplished by a number of different methods (various combinations of aircraft and bombs, missiles, naval gunfire, or even a ground-based demolition team), but, assuming a successful strike, the bridge need only be destroyed once. Parallel machines may be identical (jobs are processed at the same rate regardless of machine chosen) or unrelated (process time depends on which machine is selected).

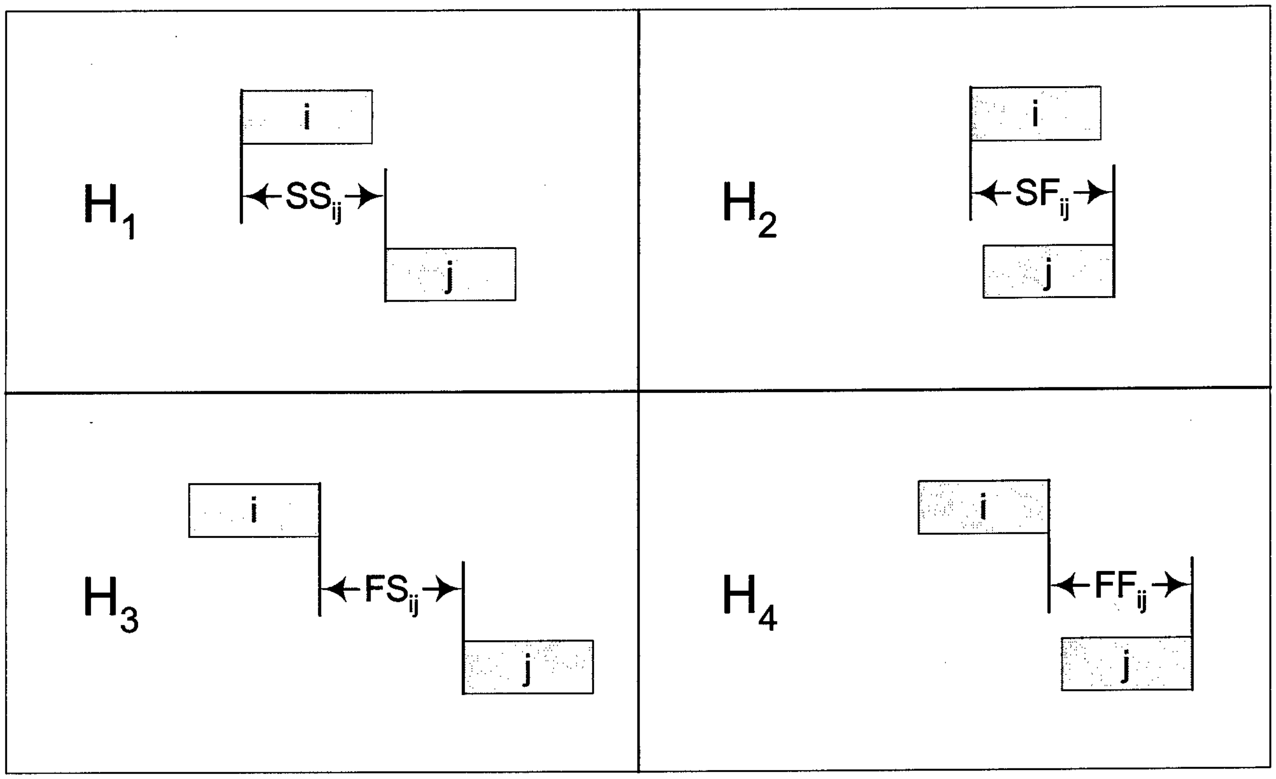
One of the most common objectives when working with parallel machine models is that of minimizing the makespan, or completion time of the last job. Often schedulers must deal with balancing the load across the machines in parallel; by minimizing the makespan a good balance is ensured [Pinedo, p.61, 1995]. Such is the case with ATO planning.

Scheduling parallel machines may be considered a two-step process. First, determine which jobs should be allocated to which machine. Second, determine the sequence of jobs on each machine, subject to any precedence constraints [Pinedo,p.62, 1995]. The analogy to ATO planning is apparent.

2.1.2.1 Precedence Constraints

Precedence constraints define timing requirements between activity pairs within projects. The most common type of precedence constraints are of the finish-start variety and are used to specify that a predecessor activity must end before its successor activity may start. Other common types of precedence constraints are start-finish, start-start, and finish-finish [Shtub, et al, p.321, 1994]. More complex activity timing requirements can be expressed by generalized precedence constraints which dictate a minimum lag time between an endpoint of a predecessor activity and an endpoint of a successor activity.

Figure 5 illustrates the four different types of generalized precedence constraint types using Gantt charts where:



H_1 models a start-start activity pair relation with a lag of SS_{ij}

H_2 models a start-finish activity pair relation with a lag of SF_{ij}

H_3 models a finish-start activity pair relation with a lag of FS_{ij}

H_4 models a finish-finish activity pair relation with a lag of FF_{ij}.

Figure 5. Generalized Precedence Constraints [Koewler, p.14, 1999]

At times, the quantity of precedence constraints among activities in a project may make the project hard to explain verbally or via a mathematical model. Therefore graphical representations of precedence constraints are frequently used. [Shtub, *et al*, p.321, 1994]. One way to represent precedence constraints is by an *activity on the arc*

(AOA) diagram (Figure 6). In an AOA representation, a node designates an event in the network and an arc is directed from node *i* to node *j* if and only if event *i* must be completed before activities leaving node *j* can begin. The duration of the activity is indicated on the arc. The boldface arrows denote the *critical path* in Figure 6. The AOA model is usually associated with *critical path method* analysis (see next section) and is the basis for most computer implementations [Shtub, *et al,* p.338, 1994].

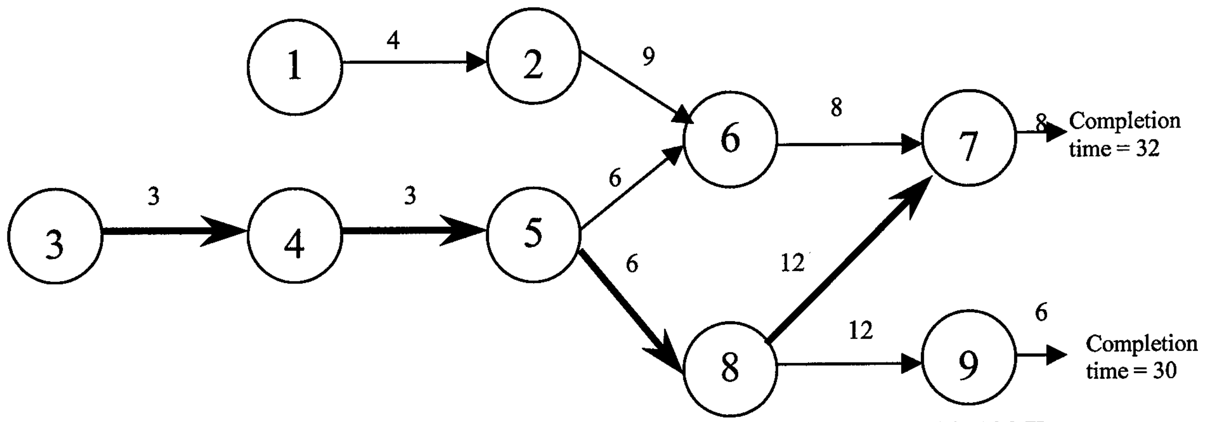
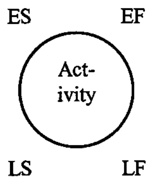


Figure 6. AOA precedence constraint graph [Pinedo, p. 66, 1995]

Alternatively, precedence constraints may be represented by an *activity on the node* network (AON). In an AON representation, a node designates an activity in the network and may display information about the activity such as duration, early start (ES), early finish (EF), late start (LS), and late finish (LF). Arcs depict precedence relationships. A typical node in an AON network would be:



Depending on the project parameters, some of the information displayed on the node may not be required. The advantage of using an AON network is that the calculations for project completion times may be performed and displayed directly on the nodes by using the Critical Path Method (next section) forwards and then backwards [Shtub, *et al*, pp. 337-338, 1994]. The complete AON network representation for the above AOA network is presented in the next section (Figure 7).

2.1.2.2 Critical Path Method

The Critical Path Method (CPM) for project scheduling uses either an AON or an AOA network for graphically portraying the relationships between the tasks and milestones in a project. Dupont and the UNIVAC division of Remington Rand developed the CPM to schedule maintenance shutdowns in chemical processing plants. CPM assumes the processing and set-up times are deterministic or "known" [Shtub, et al, p.306, 1994].

When the number of resources are unlimited, or at least as large as the number of jobs, the CPM technique yields a schedule with an optimal makespan. The algorithm is simply:

1. Schedule the jobs one at a time starting at time 0.
2. Whenever a job has been completed, start all jobs for which all predecessors have been completed (i.e. all schedulable jobs).

The *critical path* is the set of jobs that cannot be postponed without delaying the earliest finish of the whole project. These jobs are called critical jobs while jobs not on the critical path are called slack jobs. The length of the critical path (or any other path) is equal to the sum of the durations of every activity on it. If the earliest finish equals the due date, then duration of the entire project is equal to the length of the critical path

[Pinedo, p.65, 1995]. A problem where only the precedence constraints and the makespan are considered is called a classic project scheduling problem (PSP).

2.1.2.3 Critical Path Method Example

Consider nine jobs (activities) with no resource constraints [Pinedo, pp. 65-66, 1995], with the processing times as given below and with precedence constraints as shown in Figure 6:

Jobs	1	2	3	4	5	6	7	8	9
P_j	4	9	3	3	6	8	8	12	6

The makespan is calculated by using the CPM algorithm. To find the critical path, apply the CPM algorithm backwards. Start at the known makespan and work toward time 0, while adhering to the precedence relationships. In this manner all jobs are completed at their latest possible completion times and started at their latest possible starting times. If there is no due date, the jobs whose earliest possible starting times are equal to their latest possible starting times are the critical jobs and make up the critical path [Pinedo,p.65, 1995]. The remaining jobs have *slack*, that is there are periods of time where the resources associated with these jobs are available. *Total slack* is the time that the completion of an activity can be delayed without delaying the end of the project. Total slack for an activity, *i,* is calculated as $LF_i - EF_i$ (or equivalently, $LS_i - ES_i$) [Shtub, *et al*, p. 333, 1994].

Assuming no due date, the makespan, early start, early finish, late start and late finish for the project can be calculated directly on an AON network using a forward and a backward pass of the CPM algorithm (Figure 7).

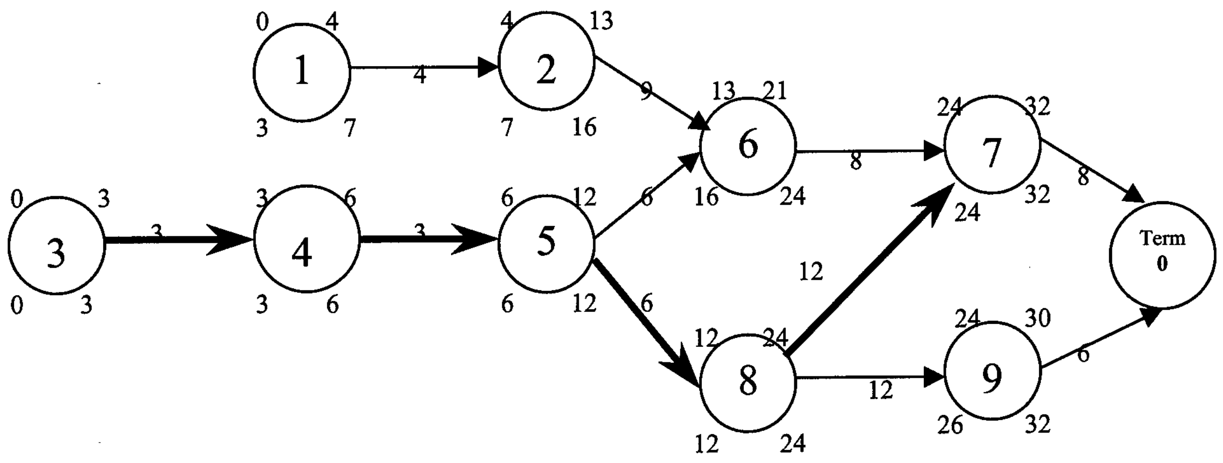


Figure 7. AON network for CPM example displaying ES,EF,LS,LF

The AON network provides the earliest completion times, C_j', and the latest completion times, C_j'', for each job as in Table 3. The makespan is 32 and is equal to the completion time of job 7. The jobs whose earliest completion time equals their latest completion time are the critical jobs and make up the critical path. The critical path therefore is the chain 3—4—5—8—7 and is depicted in Figure 7 by the bold arrows.

Table 3. Early/Late Completion Times and Slack for CPM Example

Jobs	1	2	3	4	5	6	7	8	9
C_j'	4	13	3	6	12	21	32	24	30
C_j''	7	16	3	6	12	24	32	24	32
slack	3	3	0	0	0	3	0	0	2

A linear programming (LP) model may be used to find the critical path (and hence the optimal makespan) for a PSP. Ahuja's LP formulation of the longest path network flow problem is given below [Ahuja, 1993].

Maximize $$\sum_{(i,j)\in P} \tau_i y_{ij} \qquad (1)$$

Subject to $$\sum_{i\forall(i,j)\in P} y_{ij} - \sum_{k\forall(j,k)\in P} y_{jk} = \begin{cases} -1 & \text{if } j = 1 \\ 0 & \forall\, j \in A - \{1, n\} \\ 1 & \text{if } j = n \end{cases} \qquad (2)$$

$$y_{ij} \geq 0 \;\forall\; (i,j) \in P \qquad (3)$$

$$y_{ij} \in \{0,1\} \qquad (4)$$

A is the set of all *n* activities in the project and *P* is the set of all *(i,j)* activity pairs where *i* must be completed before *j* can begin. Each y_{ij} decision variable represents the quantity of flow along the arc from activity node *i* to activity node *j* and τ_i is the completion time of activitiy *i*. The model sends one unit of flow from source to sink in the network representation of the PSP. The objective function, Equation (1), finds the longest path through the network by maximizing the sum of the weights of the arcs that make up this path. In other words, it finds the string of sequential activities that has the longest duration—the critical path. The constraints in Equation (2) maintain conservation of flow while those of Equation (3) restrict flow to be non-negative [Van Hove, p.11, 1998]. For this formulation, there must be one source (an activity with no predecessors) and one sink (an activity with no successors). If this is not the case, dummy nodes may be included in the model to satisfy the assumption. To illustrate, Figure 5 is modified below with S_0 and T_0 as the dummy source and sink, respectively.

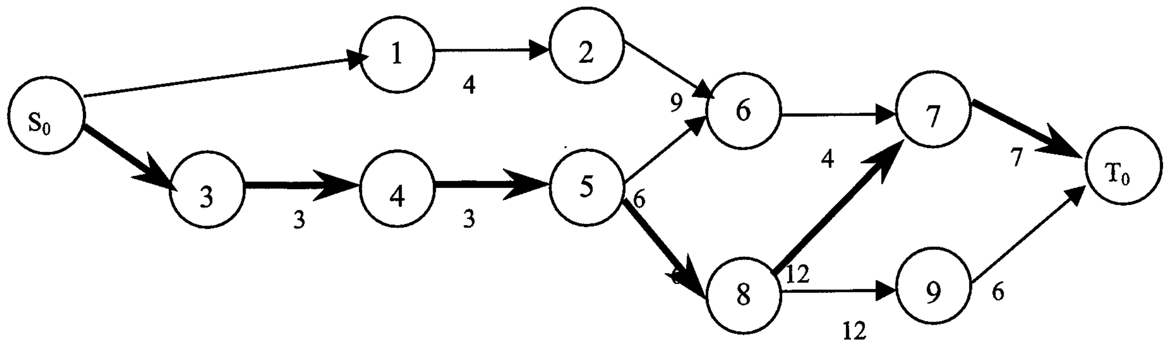


Figure 8. AON graph with dummy source and sink nodes (critical path bold)

2.1.3 Resource Constraints

In the above problem, the assumption was that there was adequate equipment so no job had to wait for a machine. The simple PSP model does not account for limited resources that are used by the activities. Hence, an optimal solution to a PSP may be infeasible if other resource constraints are present. An expansion of the PSP model developed to handle limited resources is the Resource Constrained Project Scheduling Problem (RCPSP). The formulation of the RCPSP model in Figure 9 was developed by Pritsker, Watters, and Wolfe [Pritsker *et al*, pp.93-101, 1969] and adapted by Van Hove [pp.12-13, 1998].

Parameters:	
A	the set of all activities
K	the set of all resources
P	the set of all activity precedence pairs
n	the last activity in the network
g	the project deadline
τ_i	the duration of activity i
e_i	the earliest completion time for activity i
l_i	the latest completion time for activity i
r_{ik}	the amount of resource k required by activity i
R_{jk}	the amount of resource k available in period j
Variables:	
x_{it}	equals 1 if activity i finishes in period t; 0 otherwise

$$\text{Minimize } \sum_{t=e_n}^{l_n} t x_{nt} - \sum_{t=e_1}^{l_1} t x_{1t} \tag{4}$$

$$\text{Subject to } \sum_{t=e_n}^{l_n} t x_{jt} - \sum_{t=e_1}^{l_i} t x_{it} \geq \tau_n \quad \forall (i,n) \in P \tag{5}$$

$$\sum_{i \in A} \sum_{t=j}^{j+\tau_i-1} r_{ik} x_{it} \leq R_{jk} \quad \forall k \in K \text{ and j = 1,...,g} \tag{6}$$

$$\sum_{t=e_i}^{l_i} x_{it} = 1 \quad \forall i \in A \tag{7}$$

$$x_{it} \in \{0,1\} \quad \forall i \in A \text{ and t = 1,...,g} \tag{8}$$

Figure 9. RCPSP Formulation

Since it is now necessary to have a series of binary decision variables for each activity to account for per period resource consumption, there is a significant increase in the number of decision variables in the RCPSP formulation vs. the PSP model. Recall that the PSP formulation only required a single continuous decision variable for each activity [Van Hove, p. 13, 1998].

The objective function, (4), minimizes makespan and the precedence constraints are stated in (5). Both (4) and (5) are more complex here than their counterparts in the PSP model (finding the critical path is the dual of minimizing the makespan). The constraints in (6) ensure that resource use stays within limits for each resource available in each period. The constraints represented by (7) allow each activity to be completed in only one of the possible time periods. Neither (6) nor (7) are represented in the classic PSP [Van Hove, p.13, 1998]. Equation (8) forces each decision variable to be binary (0 or 1).

2.1.4 Unrelated Parallel Machines

Thus far the parallel machines discussed were assumed to be identical. That is, job *j* is processed in the same amount of time without regard to which machine it is assigned. In an air combat environment, this would be analogous to a situation where all attack aircraft are based at the same location and are all the same aircraft types, regardless of SCL, fly at the same speed with an identical weapons capability. This would imply that all aircraft would "process" a target in the same amount of time, assuming identical bases. Clearly, the situation described above does not hold for any realistic combat scenario.

Instead, the combat situation corresponds to an environment where the machines have different performance profiles against different targets. That is, machine 1 may be able to process job 1 in a short time but may need a long time to process job 2, while machine 2 may take a long time on job 1 but may be quite speedy on job 2. Plainly, if Base 1 is much closer to Target 1 than to Target 2 then flight times of identical aircraft to Target 1 will be shorter than to Target 2. In scheduling theory, this case is called unrelated machines in parallel. For *m* machines, the notation is R_m [Pinedo,p.81, 1995].

2.1.5 Scheduling Heuristics

The RCPSP integer linear programming (ILP) model from §2.1.3 will yield an optimal solution if one exists. However, depending on the number of variables involved, it can be time-consuming to solve. Therefore, decision makers are often interested in techniques that will yield good, but not necessarily optimal, solutions quickly. These techniques are called heuristics. In scheduling theory, these heuristics are often referred to as priority rules [Pinedo, p. 142, 1995].

A priority rule frequently used when jobs are subject to arbitrary precedence constraints and arbitrary job processing times is the *largest number of successors* (LNS) rule. This means that the job that has the largest number of total successors in the precedence constraints graph has the highest priority and would be scheduled first, subject to any additional constraints [Pinedo, p.71, 1995].

When job *j* can only be processed on a subset of the available parallel machines, other dispatching rules are used. One such rule is the *least flexible job first* (LFJ) rule. Every time a machine is freed, the LFJ rule selects the job that can be processed on the smallest subset of machines, with ties broken arbitrarily.

When multiple machines are freed simultaneously, the LFJ rule does not specify which machine to consider first. One may expect that if a number of machines are free at the same time, it may be advantageous to consider the *least flexible machine* (LFM) rule. This rule assigns a job to the machine that can process the smallest subset of remaining jobs [Pinedo, p. 71, 1995].

A combination of the preceding two rules gives priority to the least flexible jobs on the least flexible machines. That is, for each time, t, the least flexible machine would be assigned the least flexible feasible job. This heuristic (dispatching rule) is abbreviated as the LFM-LFJ rule [Pinedo, p. 72, 1995]. The heuristics discussed in this section were all developed with the objective of minimizing the makespan of the project, but of course, no guarantee of optimality should be inferred.

2.2.6 The Multi-Modal Resource Constrained Project Scheduling Problem

The RCPSP formulation may be extended to include the situation where the activities can be completed in one of a number of possible execution modes. The amount and type of resources consumed and the processing time depend on the mode selected for the activity. Van Hove [p.19, 1998] adapted Boctor's multi-modal model [Boctor, p. 350, 1996].

Parameters	
A	the set of all activities
K	the set of all resources
P	the set of all activity precedence pairs
M_i	the set of all execution modes for activity i
τ_{im}	the duration of activity i in mode m
e_{im}	the earliest completion time for activity i
l_{im}	the latest completion time for activity i
r_{imk}	the amount of resource k for activity i in mode m
R_{jk}	the amount of resource k available in period j
Variables	
x_{imt}	$= 1$ if activity i finishes in period t using mode m

Minimize
$$\sum_{m \in M_n} \sum_{t=e_n}^{l_n} t x_{nmt} \tag{9}$$

Subject to:
$$\sum_{m \in Mj} \sum_{t=e_j}^{l_j} (t - \tau_{jm}) x_{jmt} - \sum_{m \in M_i} \sum_{t=e_i}^{l_i} t x_{imt} \geq 0 \quad \forall (i,j) \in P \tag{10}$$

$$\sum_{i \in A} \sum_{m \in M_i}^{l_j} \sum_{t=j}^{j+\tau_i-1} r_{imk} x_{imt} \leq R_{jk} \quad \forall (j,k) \tag{11}$$

$$\sum_{m \in M_i} \sum_{t=e_i}^{l_i} x_{imt} = 1 \quad \forall i \in A \tag{12}$$

$$x_{imt} \in \{0,1\} \quad \forall (i,m,t) \tag{13}$$

Figure 10. Van Hove's adaptation of Boctor's MMRCPSP Formulation

Again, the objective function, (9), models the goal of minimizing the makespan. The significant difference between this model and that for the RCPSP is the decision variable, x_{imt} vs. x_{it}. This binary variable equals one if and only if activity i is executed in mode m and completed in time t.

2.2.7 The Generalized MMRCPSP

In the IP formulations thus far, the precedence constraints were strictly start-to-end. However, a PSP may employ another type of precedence constraint or, indeed, a combination of types. The start-to-end constraints are not flexible enough to model mission sequencing in air campaign planning. For example, start-to-end constraints force mission j to wait until all aircraft employed in mission i have landed and their turn time has expired (here activity j follows i). Certainly, this type of constraint is inadequate. Generalized precedent constraints, by contrast, may be used to enforce any timing requirement called for in an operational scenario and are therefore required. The generalized MMRCPSP has the abbreviation, MMGRCPSP.

Consider the following diagram (Figure 11). If Target 1 were a Surface-to-Air Missile (SAM) site, mission requirements may be such that Target 1 must be attacked before subsequent missions that strike deep interdiction targets (Targets 2 and 3) may cross the Forward Line of Own Troops (FLOT). This requirement may mean that the missions to Targets 2 and 3 may have to delay their take off. This concept, minimum lag time, is more fully covered in §3.3 along with its counterpart, maximum lag time. Suffice it to say generalized precedence constraints must be included in the formulation.

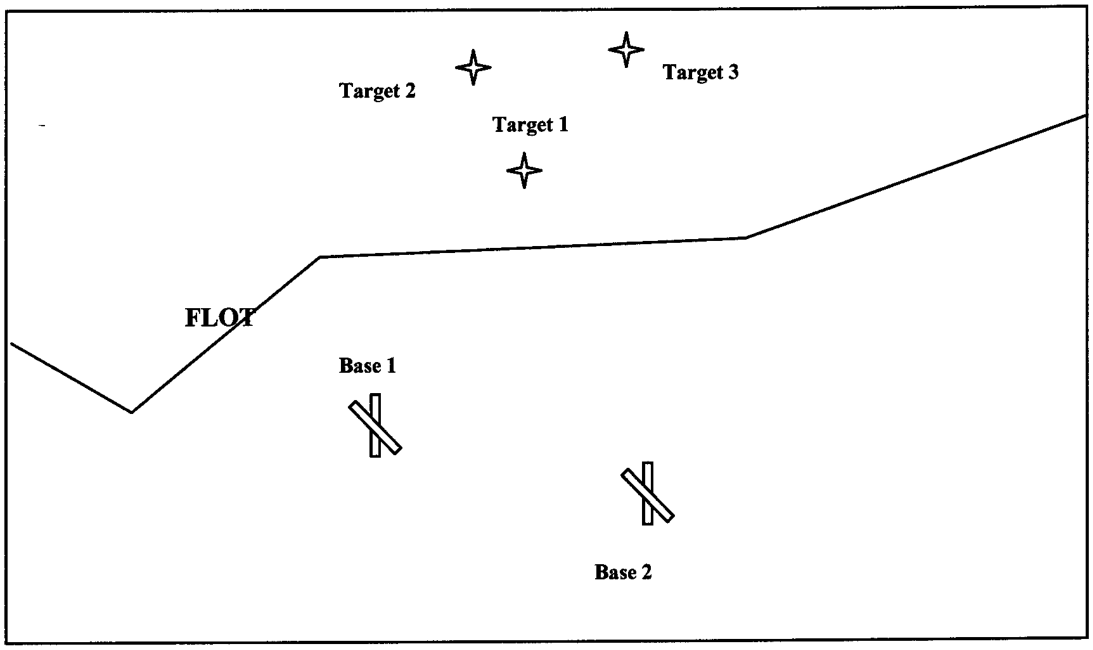


Figure 11. Scenario for Generalized Precedence

2.2.7.1 Doubly Constrained Resources

Before continuing with the model formulation, the reader should have a clear understanding of the connection between scheduling theory terminology and that of air campaign planning. These associations are provided in Table 4.

Table 4. Terminology Associations [Koewler, p. 16, 1999]

Scheduling	Air Campaign Planning
Activities or jobs	Targets or strike mission assigned to a target
Mode	Weaponeering and base selection
Processing time	Mission duration
Precedence Constraint	Mission timing requirement
Resource	Squadron or unit
Asset	An individual aircraft

In air campaign planning, the number of allocated aircraft assigned to a particular squadron equals the limit on how many aircraft may be tasked during the same time period. However, this is not the limit on how many aircraft may be tasked from this squadron throughout the day. Generally, each aircraft in a unit can fly more than one mission in an ATO day—provided that the crews, fuel, and ordnance are available. Associated with each unit is a turn rate—the number of sorties an individual aircraft may fly per day. The number of sorties a unit may fly per day is equal to the turn rate multiplied by the number of allocated aircraft in the unit. For example, suppose a squadron had 16 F-15Es allocated for interdiction with a turn rate of 2.5. Obviously, at most 16 aircraft may be tasked at any one time while the turn rate indicates that 16 x 2.5 = 40 sorties may be flown throughout the day.

Let K be the set of units in the problem and $k \in K$. The renewable aspect of k is the limit R_k, the actual number of allocated aircraft assigned to the unit. Again, this means that at any given time during the planned ATO day, no more than R_k units of resource $k \in K$ may be in use [Van Hove, p. 43, 1998].

Now let tr_k be the turn rate for unit k. Then N_k, the number of sorties that unit k can generate in an ATO day, is given by $N_k = R_k \times tr_k$ truncated to the nearest integer. The nonrenewable aspect of resource k is the limit dictated by N_k [Van Hove, p. 43, 1998].

Each unit in the air campaign planning model is a *doubly constrained resource*. In other words, the assets associated with the resource are both renewable and nonrenewable.

2.2.7.2 Multi-Modal Activities

The TNL determines the activities for an air campaign planning scenario. The TNL lists the JFACC-approved targets, along with weaponeering options for each target. The information associated with an option includes the number and type of aircraft, SCL, and the probability of kill (PK) or expected percent damaged (PD). A valid execution mode for a target (activity) is given by the number and type of aircraft together with a unit possessing the indicated type of aircraft.

The mission duration for an execution mode depends on the distance between the target and the base and on the speed of the aircraft associated with the mode. A mission route consists of several legs. For this research, a route consists of the following legs: take-off to FLOT, FLOT to target (ingress), target to FLOT (egress), FLOT to landing. Of course, mission planners at the Wing Operations Center would construct more detailed routes. Since the AOC personnel generally do not know the exact routes the aircraft will fly, these straight-line route legs can be used for ATO planning estimates. The estimated time to complete each of the legs is then calculated using a nominal airspeed for each aircraft type. The sum of these route times plus the aircraft turn-around (prep time for the next mission) makes up the mission duration (processing time). If necessary, a time window may be established during which a mission has exclusive access to a target. This window is bounded by a time on target (TOT) and a time off target (TFT). If this window were included, the mission duration would increase by the length of this interval.

2.2.7.3 MMGRCPSP Formulation

In the formulation of the MMGRCPSP problem (Figure 12), the objective function, (14), again minimizes the makespan of the schedule. Equation (15) enforces the sortie rate limit (nonrenewable resource), while equation (16) enforces the limit on the number of aircraft tasked per time period (renewable resource). The binary decision variable, x_{imt}, is equal to one if activity i starts in period t and is executed in mode m. Equation (17) enforces the lag times between predecessor/successor activities. The binary decision variable x_{jnt} selectively enforces the applicable constraint and relaxes the redundant constraints. As the delta ($\Delta_{ijm'n}$) can be positive or negative, both minimum and maximum precedence constraints can be expressed by using this constraint twice, once for precedence pair (i,j) and once for precedence pair (j,i) [Van Hove, 2000].

The model generates an optimal schedule in terms of makespan. It does not, however, model target priority classifications, nor does it model maximum lag times or replanning. Van Hove solved a relatively small 10-job problem to optimality in less than one second, while the optimize-r, CPLEX, took almost 9 hours to solve. Van Hove's case study sample problem (before preprocessing) required more than six million binary decision variables and 80,000 constraints [Van Hove, p. 143, 1998]. Using his hybrid integer decomposition approach with an objective of minimizing makespan, Van Hove was able to solve this problem to optimality in less than an hour [Van Hove, p. 142, 1998]. By contrast, solving this same problem to optimality would have required over a century using the CPLEX optimizer [Van Hove, 2000]. Van Hove's hybrid integer program decomposition approach increased the size of problems that can be realistically solved to optimality from 10 activities(targets) to 100 activities (targets). There are, however, operational problems that exceed this 100-target limit.

Parameters:

A	the set of all activities
d	the index of terminal activity
e_{im}	the earliest completion time for activity *i in mode m*
l_{im}	the latest completion time for activity *i in mode m*
M_i	the set of all execution modes for activity *i*
τ_{im}	the duration of activity *i* in mode *m*
S_i	the set of generalized successors of activity *i*
Δ_{ijmn}	the minimum lag between the start time of activity *i* in mode *m* and the start time of activity j $\in S_i$ in mode *n*
K	the set of all double constrained resources
r_{imk}	the amount of resource *k* required by activity *i* when being executed in mode *m*
R_k	the per period availability of resource *k*
N_k	the total amount of resource *k* available
g	the deadline for the project under consideration

Variables:

x_{imt}	= 1 if activity *i* starts in period *t* and is executed in mode *m*

Minimize $$\sum_{m \in M_d} \sum_{t=e_{dm}}^{l_{dm}} \left(t + \tau_{dm}\right) x_{dmt} \qquad (14)$$

Subject to: $$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=e_{im}}^{l_{im}} r_{imk} x_{imt} \le \mathrm{N}_k \quad \forall k \in K \qquad (15)$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=j-\tau_{im}+1}^{j} r_{imk} x_{imt} \le R_k \quad \forall\, k \in K,\ \mathrm{j} = 1 \ldots \mathrm{g} \qquad (16)$$

$$\sum_{n \in M_j} \sum_{t=e_{jn}}^{l_{jn}} \left(\Delta_{ijm'n} - t\right) x_{jnt} + \sum_{t=e_{im'}}^{l_{im'}} t x_{im't} \le \sum_{m \in (M_i \backslash m')} \sum_{t=e_{im}}^{l_{im}} g x_{imt} \qquad \forall\, \mathrm{i} \in \mathrm{A},\ \forall\, \mathrm{j} \in \mathrm{S}_i,\ m' \in \mathrm{M}_i \qquad (17)$$

$$\sum_{m \in M_i} \sum_{t=e_{im}}^{l_{im}} x_{imt} = 1 \quad \forall i \in A \qquad (18)$$

$$x_{imt} \in \{0,1\} \quad \forall (i,m,t) \qquad (19)$$

Figure 12. Van Hove's complete MMGRCPSP model formulation [Van Hove, p. 49]

2.3. Goal Programming

Goal programming (GP) was developed to solve multiple objective decision making problems (MODMP), has been successfully used to solve numerous applications of real-world problems, and has been accepted as a basic mathematical programming method for solving MODMP [Baykasoglou, p. 960, 1999]. GP first appeared in the nineteen fifties to obtain "constrained regression" estimates for an executive compensation problem [Charnes, Cooper, & Ferguson, 1955]. The purpose of GP is to simultaneously satisfy several goals relevant to the decision making problem [Romero, p. 2, 1991].

The first step in formulating a GP model is establishing a set of goals. Each goal is a combination of an attribute together with its corresponding aspiration level. Whether this aspiration level may be required to be satisfied exactly, surpassed, or not exceeded must be indicated [Romero, p. 2, 1991]. To accomplish this, deviational variables need to be introduced into the model. A positive deviational variable, d^+, indicates the number of units by which a goal's aspiration level has been surpassed. A negative deviation, d^-, indicates the number of units by which a goal's aspiration level is lacking. Clearly, at least one of the variables must be zero; if the goal is surpassed, the negative deviational variable must be zero and vice versa. If the goal is met exactly, both d^+ and d^- are zero [Romero, p. 2, 1991].

Algebraically, the *i*th goal is expressed as:

$$f_i(\mathbf{x}) - d_i^+ + d_i^- = b_i$$

where $\mathbf{x}$ is the vector of decision variables and b_i is the aspiration level for the *i*th goal, d_i^+ is the over-attainment of b_i, and d_i^- is the underachievement of goal *i*. If it is desirable that the ith goal meet or exceed the aspiration level ($f_i(\mathbf{x}) \geq b_i$), then d_i^- must be as small as possible (minimize d_i^-). Alternatively, if it is preferred that goal attainment level not be exceeded, ($f_i(\mathbf{x}) \leq b_i$), then d_i^+ must be as small as possible (minimize d_i^+). If the ith goal should be achieved exactly ($f_i(\mathbf{x}) = b_i$), then both deviational variables must be as small as possible (minimize $d_i^+ + d_i^-$). While a single goal could be expressed as an inequality without deviations, deviations are required to trade off between goals if all goals cannot be simultaneously satisfied.

The two most common types of GP are preemptive weighted GP and non-preemptive weighted GP. Preemptive weighted GP models (commonly referred to as lexicographic goal programming (LGP)) satisfy goals in ordinal order of importance. Non-preemptive weighted GP models may have weights, developed by the decision-maker, assigned to the goals and are referred to as weighted goal programming models (WGP). In WGP models, all goals are considered simultaneously [Baykasoglu, p. 960, 1999]. There are other GP variants, but WGP and LGP contain the most common, basic features of GP. However, because of the difficulties associated with assigning weights to target priorities, a preemptive GP is appropriate for this research.

2.3.1 Lexicographic Goal Programming

In LGP, once goals, target levels, and the deviational variables to be minimized have been determined by the decision-maker, the specific level of priority must be assigned to each goal or group of goals. In other words, the decision-maker must rank his goals from the most important (goal 1) to the least important (goal *n*).
An objective function must be determined for the LGP model. The objective function coefficient for the variable representing goal *i* is the preemptive weight P_i, and it is assumed $P_1 >>> P_2 >>> P_3 >>> \ldots >>> P_n$. Because of the nature of LGP, P_i is "infinitely" greater than P_{i+1} [Deckro and Hebert, p. 150, 1988]. This weighting ensures that the goal program first tries to satisfy the decision-maker's most important goal. Then, among all points that satisfy goal 1, the decision-maker tries to come as close as possible to satisfying goal 2, and so forth. The process continues until the only way to draw closer to

satisfying a goal is to increase the deviation from a higher-priority goal [Winston, p. 778, 1994].

2.3.2 Goal Interval Programming

One variant on LGP is called Goal Interval Programming (GIP). Often, standard GP techniques using preemptive weightings and deviations from a target value are overly constrained in their requirements [Charnes *et al*, p. 351, 1976]. LGP models are incapable of considering trade-offs involving the satisfaction of goals that are associated with different priority levels [Deckro and Hebert, p. 150, 1988]. Replacing an original stipulated goal with a *goal interval* is one way to allow a decision maker to assess the feasibility of an inter-goal trade-off. The GIP model makes it possible to focus on one activity at a time while allowing for departures from goals (or goal intervals) within limited ranges [Charnes *et al*, p. 357, 1976].

In GIP, a goal is considered to be satisfied if the deviation of the corresponding variable from the desired level is within a pre-determined range [Charnes *et al,* 1976]. GIP would be appropriate for the air campaign planning problem especially in the realm of intra-class trade-offs. Targets are usually assigned broad target priority classifications (i.e. Priority 1, 2, 3, 4 for example). Within these classifications are sub-categories (1A, 1B, 1C, and so on). A planner may be willing to roll a target with a 2B classification into a future ATO if it meant picking up, say, four 2C targets for today's ATO.

2.3.3 Conventional Targeting Effectiveness Model

The Conventional Targeting Effectiveness Model (CTEM) was developed for HQ USAF/XOOC (Checkmate) for analysis of current operations [Cotsworth, 1993]. CTEM is a conventional weapons version of the Arsenal Exchange Model (AEM) which has been the standard force structure analysis tool for some time [Yost, p. 55, 1996]. CTEM is typically used as a preemptive goal program where targets are grouped into classes that must be attacked in a certain priority order [Yost, p. 55, 1996]. Solution times for CTEM range from one to three hours, depending on scenario. Experience with CTEM and AEM show that a goal-orientation is much easier for users than a target-value orientation [Yost, p. 58, 1996].

CTEM is now used to aid in MAAP production within the JFACC Planning Tool (JPT). A great deal of aggregation and several assumptions must be made in order for the problem to be solvable by CTEM in a reasonable amount of time [Haas, p. 57, 1998]. If input data needs to be corrected, the entire run of the model must be redone and this may be time prohibitive under air campaign planning conditions [Haas, p. 59, 1998].

Based on the precedent set by CTEM, a LGP model for detailed mission planning seems appropriate. Additionally, a method that can yield good, quick solutions to the model would be of great benefit. Indeed, such a method may well serve in MAAP production as a proxy for solving the CTEM model to optimality, especially under conditions where a rerun of the CTEM model is required.

2.3.4 An Appropriate Heuristic for GP Models

There are pitfalls with using deterministic search algorithms (like simplex and branch and bound, for example) for solving GP models. Like all methods, they may encounter difficulties with local optima when searching for a global or near-global optimum solution. Moreover, since they always move in an improving direction it may be difficult to escape the trap of local optima. They also require the analytic form of the model, without which solving the problem is impossible. In real-world problems, however, the analytic form of some objectives may not be available [Baykasoglu, p. 960, 1999].

Stochastic search methods such as genetic algorithms, simulated annealing and tabu search, on the other hand, have built-in mechanisms that offer a more robust methodolgy for trying to escape local optima. They are generally problem independent and can handle any kind of constraint or objective function, and do not require an exact analytic form [Baykasoglu, p. 960, 1999]. The TS solution method involves working with neighborhoods (more than one solution at a time) which gives it the ability to deal with multiple goals (objectives) easily [Baykasoglu, p. 960, 1999]. The next section presents TS in some detail.

2.4 Tabu Search

Tabu search (TS) is a meta-heuristic procedure for solving large combinatorial optimization problems. Meta-heuristics guide other heuristic procedures to escape the trap of local optima. TS has proven to be extremely successful in solving to optimality or near-optimality a variety of classical and practical problems [Glover, p. 74, 1990].

2.5.1 Tabu Search Description

Tabu search uses *recency* (short-term memory) and *frequency* (long-term memory) to search a solution space efficiently by prohibiting the search from remaining in regions found to be locally optimal and forcing the exploration of other regions (perhaps even infeasible) not yet encountered. Specifically, the search examines a neighborhood of a given solution. This neighborhood is defined to be the set of all solutions one move away from the current solution. The move must be defined for each type of problem. For example, in a traveling salesman problem (TSP), a move may be defined as swapping the order of two nodes [Nowicki and Smutnicki, p. 800, 1996]. The appropriate move for a given problem is usually established through experimentation [Glover and Laguna, p. 80, 1993]; several different moves may be tried to see which one (or which combination) produces the best results.

The search utilizes the *k* most recent moves to classify moves that are *tabu* (restricted*)*, which keep the search from undoing the *k* previous moves (where *k* is some pre-defined integer). Once a move is made, reversing that move is classified as tabu and recorded on the *tabu list* (of length k). The tabu list keeps track of recent moves and is an example of what is meant by recency or short-term memory. Moves classified tabu may still be selected if they pass what is known as the *aspiration criteria.* An aspiration criterion is defined as a condition that must be met in order to allow a tabu move to take place. Permitting a tabu move to be chosen only if the move gives the best solution found so far is an example of an aspiration criterion [Glover and Laguna, p. 76, 1993].

The tabu search may (unknowingly) encounter the optimal solution for a problem in a shorter time than classic methods. However, the TS, by itself, neither guarantees optimality nor can it usually tell if the best solution obtained is optimal. An exception to the latter is Nowicki and Smutnicki's algorithm (labeled TSAB) for job shops that can detect an optimum under certain specific conditions [Nowicki and Smutnicki, p. 803, 1996]. Likewise, if TS were applied to a GP model and all of the goals were satisfied, then optimality will have been achieved and the TS would recognize this as well by simply checking the current solution against the target deviation vector. A typical TS, even if it finds the optimal solution, continues to search until a pre-specified stopping rule is met (exception: TSAB). Candidate stopping rules include:

1. The time limit has expired.
2. The maximum number of iterations has been reached.
3. The current solution is within some ε (epsilon) of a lower bound (upper bound for a maximization problem).
4. There has been no improvement in the solution for *n* iterations.

As long as the set number of iterations or the time limit is established at a reasonable value, the TS generally yields an excellent solution quickly [Nowicki and Smutnicki, p.799, 1996]. It is possible to integrate TS with algorithms that guarantee optimality (such as branch and bound) and this is an avenue of research worthy of investigation [Glover, pp. 82-83, 1990], but not within the scope of this research.

2.5.1.1 Tabu Tenure

Tabu tenure is the number of iterations that a move is classified as tabu (also the *length* of the tabu list). The tabu tenure that achieves the best results is experimental; different values are tried and compared. Glover suggests that in several applications of TS, a tabu list size of seven represents a highly effective value and that the best tabu tenures consistently fall in the interval from five to 12 [Glover, p. 83, 1990].

2.5.1.2 Intensification

TS uses flexible memory for *intensification* and *diversification*. 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 (local or even global). Reduction of the tabu tenure is a basic intensification strategy. An intensification phase is initiated after a pre-set number of iterations. A record of the best (elite) solutions found so far is kept and the TS re-starts (with a shorter tabu tenure) from each of these in turn. The justification for re-starting from these best solutions is simple—they have a high probability of being close to a local (and perhaps global) optimum. The search may not have found these optima because the moves that would have led in the direction of the optima may have been on the tabu list. By reducing the tenure, some of those moves may drop off the tabu list, thus allowing the search to proceed towards the nearby local optima (if it exists). After a certain number of non-improving iterations, the intensification phase begins anew from the next solution on the elite solution list. Once the intensification phase has been completed, the TS may terminate, start a diversification phase, or continue searching with its original tabu tenure from the current solution or from the last solution encountered before the intensification phase was begun [Glover and Laguna p. 47-50, 1997].

2.5.1.3 Diversification

At times, the TS continually returns to an attractive region of the solution space (the trap of local optima). This means that the TS has reached an attractive area of the solution space and the moves selected tend to keep the TS in this region. Quite possibly, when a tabu move has achieved its tenure and comes off the list, it is immediately chosen as the next move. The TS can use one of many diversification strategies to escape this trap.

If a diversification phase were desired, the simplest strategy would be to lengthen the tabu tenure, which may cause the search to move to unexplored regions. Often, an attractive variable, once in the solution, remains in the solution. It may never be selected to be dropped since it may never be the choice for "least un-improving move." However, if the tabu tenure were longer such that all of the other decision variables currently in the solution were tabu, then this attractive variable may be considered for leaving and one of the less frequently used variables may enter. This is an example of using recency for diversification.

Another simple diversification strategy would employ frequency or long-term memory. Keeping track of the number of iterations that the decision variables are in the solution and then penalizing the objective function for keeping an attractive variable in the solution is one way to diversify. Alternatively, the TS could track the number of times a variable has been toggled during the search. It could then penalize the objective function for selecting a variable that has been often added or dropped, thus allowing less frequently selected variables to be chosen. Diversification generally employs long-term memory to expand the search to an unexplored area. Other diversification strategies are available such as path relinking and strategic oscillation, but like all TS strategies and parameters, experimentation is in order to find the best combination for a problem instance. At the completion of the diversification phase, the tabu tenure will revert to its original value and the search proceeds as before [Glover, p. 47-50, 1997].

2.5.2 Benefits of Using Tabu Search

The advocates of TS suggest it makes use of memory, both long and short term, to maximize the depth (intensification) and breadth (diversification) of the neighborhood search. Additionally, intelligent use of candidate lists can speed up the search by forcing an evaluation of a fractional subset of the neighborhood. The tools at the disposal of an effective tabu search will, in most cases, find a very good solution (often better than the accepted standard for a class of problems) in a relatively short time [Glover, p. 75, 1990].

TS was chosen as the method for this research because it has been shown to be effective for solving many types of combinatorial optimization problems (of which the problem being addressed by this research is one). For example, Laguna, Barnes, and Glover developed a TS application that obtained optimal solutions to all test problems for which optimality could be verified. Moreover, their application obtained solutions within a few percent of an optimality bound for larger problems that branch-and-bound methods could not solve in a reasonable time [Laguna, Barnes, and Glover, 1989]. Since tasking aircraft to targets can involve hundreds or even thousands of targets and aircraft, the problem can be very large. Bearing in mind the size and type of the problem and TS's performance record, research using TS for this type of problem seems appropriate.

Scheduling provides one of the most fruitful areas for modern heuristic techniques in general and for tabu search in particular [Glover, p. 127, 1993]. Nowicki and Smutnicki tested their TSAB application for job shop scheduling against a benchmark set of problems and compared their results to those of a shifting bottleneck heuristic and simulated annealing. Their procedure outperformed the other methods with respect to (adjusted) CPU time and the quality of the generated makespans [Nowicki and Smutnicki, p. 808. 1996].

2.5.3 Steps in Applying a Basic Tabu Search

Figure 13, taken from Glover (1990) illustrates the basic steps involved in a TS. These steps are common to all tabu searches; however the details (candidate lists, tabu tenure, aspiration criteria and so forth) will differ depending on the problem and on the phase of the search (such as diversification or intensification). Custom procedures not listed here are used to tailor the TS to specific problems.

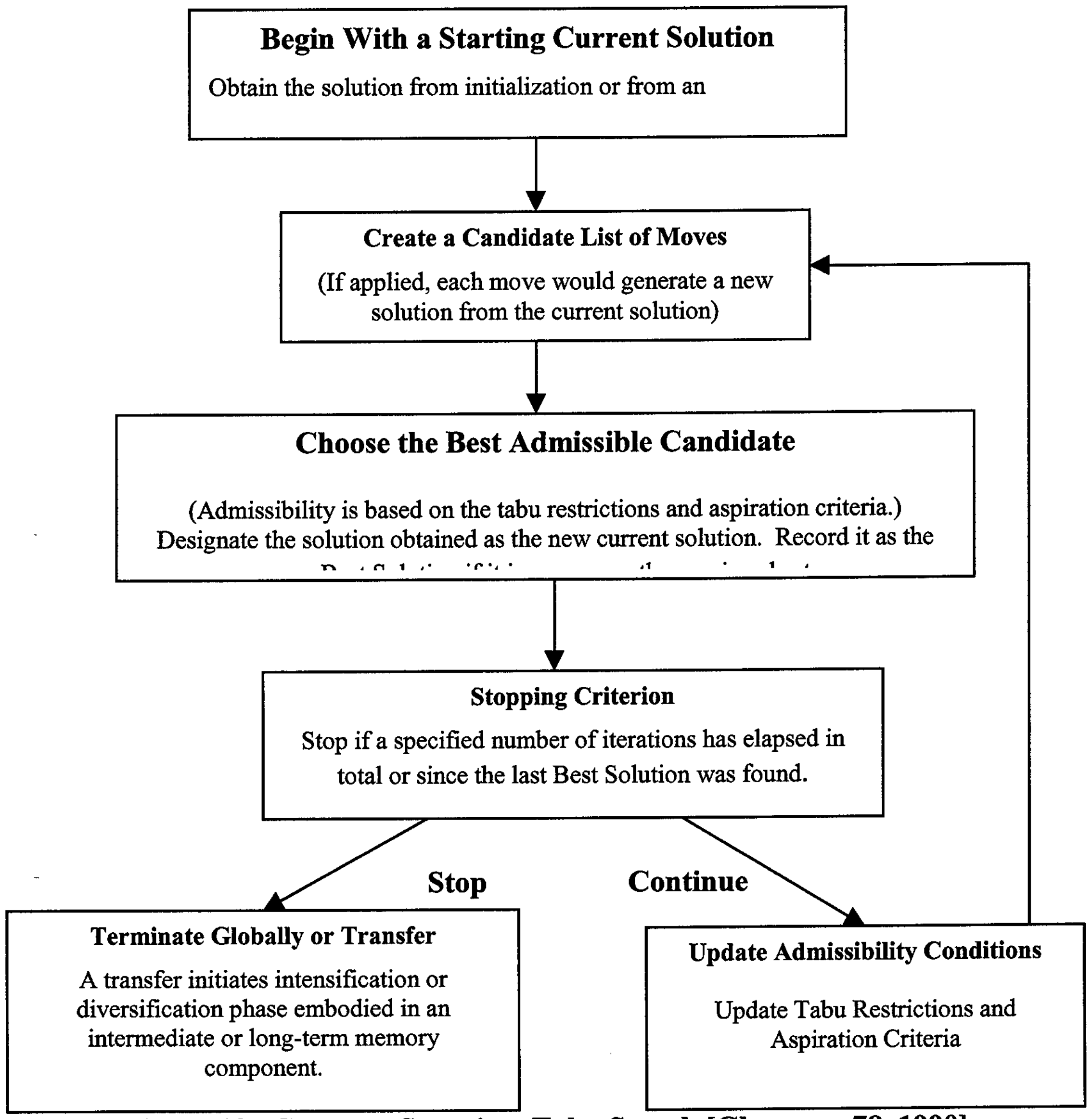

Begin With a Starting Current Solution
Obtain the solution from initialization or from an
Create a Candidate List of Moves
(If applied, each move would generate a new solution from the current solution)
Choose the Best Admissible Candidate
(Admissibility is based on the tabu restrictions and aspiration criteria.)
Designate the solution obtained as the new current solution. Record it as the
Stopping Criterion
Stop if a specified number of iterations has elapsed in total or since the last Best Solution was found.
Stop
Continue
Terminate Globally or Transfer
A transfer initiates intensification or diversification phase embodied in an intermediate or long-term memory component.
Update Admissibility Conditions
Update Tabu Restrictions and Aspiration Criteria

Figure 13. Common Steps in a Tabu Search [Glover, p. 78, 1990]

2.5.4 A Simple Example of Tabu Search

TS has been used successfully on various optimization problems including traveling salesman problems, vehicle routing problems, and job shop scheduling [Glover, p. 75, 1990]. This section demonstrates how TS works with a simple job shop-scheduling problem. Advanced principles of TS are not demonstrated through this example.

Consider a shop with one machine and four jobs [Pinedo, pp. 151-152, 1995]. The processing time, due date and weight for each job is given in Table 5. The objective is to minimize the total weighted tardiness (in scheduling notation this is a $1 \mid\mid \Sigma w_i T_i$ problem).

Table 5. Single Machine Example: Data

Jobs	1	2	3	4
p_j (processing times)	10	10	13	4
d_j (due dates)	4	2	1	12
w_j (Weights)	14	12	1	12

The neighborhood for this problem contains all schedules that can be obtained from the current schedule through adjacent pairwise interchanges. The tabu list is a list of pairs of jobs *(i,j)* that were swapped within the last two moves (i.e. tabu tenure, *k,* is two) and cannot be swapped again unless the swap satisfies the aspiration criterion or it has left the tabu list. The aspiration criterion is the value for a move that would allow the tabu restriction to be overridden. Initially, the tabu list is empty.

As a first schedule, the sequence $S_1 = 2,1,4,3$ is randomly chosen; the corresponding value of the objective function is 500 (see Table 6). The aspiration

Table 6: Single Machine Example: First Schedule

Jobs (S_1)	Days late	Contrib. to Obj. Func.
2	8	96
1	16	224
4	12	144
3	36	36
Total		500

criterion for a move on the tabu list is that its objective function value be less than the best known objective function value (in this case, the best objective function value obtained so far is 500). There are three schedules in the neighborhood of S_1, 1,2,4,3; 2,4,1,3; and 2,1,3,4. The respective values of the objective functions are 480, 436, and 652. Selection of the best non-tabu sequence, with a value of 436, results in S_2 = 2,4,1,3. The aspiration criterion is now 436 (see Table 7). The tabu list is updated and contains (1,4).

Table 7. Single Machine Example: Second Schedule

Jobs (S_2)	Days late	Contrib. to Obj. Func.
2	8	96
4	2	24
1	20	280
3	36	36
Total		436

There are three schedules in the neighborhood of S_2, 4,2,1,3; 2,1,4,3; and 2,4,3,1. The respective values of the objective functions are 460, 500, and 608. The best move at this point is the first one. Note that the second schedule (move) is tabu since the swap (1,4) is on the tabu list. Therefore, even if it was the best move, we could not use it unless the aspiration criterion was met and we could override the tabu restriction. The selection of the first move increases the current objective function value (see Table 8). Regardless, S_3 = 4,2,1,3 and the tabu list now consists of (1,4) and (2,4).

Table 8: Single Machine Example: Third Solution

Jobs (S_3)	Days late	Contrib. to Obj. Func.
4	0	0
2	12	144
1	20	280
3	36	36
Total		460

There are three schedules in the neighborhood of S_3, 2,4,1,3; 4,1,2,3; 4,2,3,1. The respective values of the objective functions are 436, 440, and 632. The best move is the first one, but this move is on the tabu list and the aspiration criterion has not been met. We must therefore choose the second move (see Table 9). S_4 is 4,1,2,3 and the tabu list contains (2,4) and (1,2).

Table 9. Single Machine Example: Fourth Schedule

Jobs (S_4)	Days late	Contrib. to Obj. Func.
4	0	0
1	10	140
2	22	264
3	36	36
Total		440

The neighborhood of S_4 consists of 1,4,2,3; 4,2,1,3; and 4,1,3,2 with respective values of 408, 460, and 586. The best move is the first one (see Table 10), which is not tabu and, in any case, meets the aspiration criterion. S_5 is 1,4,2,3 and the tabu list consists of (1,2) and (1,4). In this case, S_5 is optimal. Generally, the TS generally does not know when a solution is optimal. The search would continue in this fashion until one of the stopping rules is fulfilled.

Table 10: Single Machine Example: Final Schedule

Jobs (S_5)	Days late	Contrib. to Obj. Func.
1	6	84
4	2	24
2	22	264
3	36	36
Total		408

For this simple problem, complete enumeration of all the possible combinations would not be prohibitive. Since there are only four jobs, each with only one operation, and each job must be processed on one machine, the total number of combinations is 4! or 24. Here a heuristic search, such as TS, would not be necessary. But suppose each job had three operations with each operation to be processed on one of two machines. Now there are up to $[(4*3/2)!]^2$ or 518,400 feasible schedules to enumerate and evaluate [*Scheduling Theory Course,* 1999]. For even relatively small job shop-scheduling problems, complete enumeration may be cumbersome. In such a setting TS, as well as other search procedures, which evaluate some small fraction of the possible combinations, are often advantageous.

2.6 Object-Oriented Programming and Java

Objects are essentially reusable software *components* that model items in the real world. Most software developers find that using a modular, object-oriented design and implementation approach can make software development more productive than with previous popular programming techniques such as structured programming because object-oriented programs are easier to understand, correct, and modify [Deitel and Deitel, p.12, 1998]. Many different programming languages have been developed that are object-oriented. Among these are Smalltalk and Java, while the popular language C^{++} is a hybrid—it is possible to program in a structured programming style or an object-oriented style or in a combination of both [Deitel and Deitel, p. 12, 1998].

Java was selected as the object-oriented language for this research for several reasons. First, Sun has announced its Java Database Connectivity (JDBC) standard. This is intended for developers of heavy-duty database applications (see §2.3). Second, because Java was developed to fully utilize and exploit the Internet and the World Wide Web, it was purposely built to be platform independent. A Java application will run just as readily on a Sun SPARCstation as it will on a PC. Third, Java is based on C and C^{++} and, as such, programmers who are familiar with those languages can understand and modify Java applications with minimal cross training.

2.7 Chapter Summary

This chapter began by providing background on the ATO process. Some key elements of scheduling theory and project management were outlined including network diagrams, the critical path method, scheduling heuristics. A discussion of various project scheduling problems was followed by the formulations of mathematical models used to solve them to optimality. Selected concepts of goal programming were presented and then a detailed look at the meta-heuristic, tabu search, was provided.

CHAPTER 3. METHODOLOGY

This chapter details how the topics and methods from Chapter 2 were applied to the interdiction aircraft assignment problem. An explanation of the heuristic used to provide an initial solution to the problem is provided. The goal programming model and formulation for the planning and the replanning problems are then presented. The chapter finishes with a detailed look at the TS application developed to provide combat planners and operations duty officers with scheduling and re-scheduling options.

3.1 Initial Solution Heuristic

As this research focuses on improving the APS solution, it is appropriate to investigate generating a better initial solution that considers precedence and flexibility. Recall the *largest number of successors* (LNS) rule from Section 2.2.5. Under this rule, the job with the most successors has the highest priority [Pinedo, p. 69, 1995]. For example, a Surface-to-Air Missile (SAM) site has to be downgraded before any succeeding missions could ingress the immediate area. This mission obviously would have a high priority under the LNS rule.

Recall the LFM-LFJ heuristic (Section 2.2.5) [Pinedo, pp. 71-71, 1995] and consider the earlier example from Section 2.2.7. If the aircraft are the "machines" and the targets are the "jobs", then the heuristic would select Aircraft Type 3 from Base 2 first (it is the least flexible; it can only be used against one target) and assign it to Target 2. Aircraft Type 1 from Base 1 would be selected next and assigned to Target 3. Finally Aircraft Type 2 from Base 1 would attack Target 1.

A comment is required concerning tie breaking—suppose Aircraft Types 1 and 2 could both be used against all three targets. Once Aircraft Type 3 has been scheduled, either type could be considered next. One tie-breaking rule might be to choose the aircraft type that has the most available assets in-theater. For this example, there are twelve Aircraft Type 1 and seven Aircraft Type 2, so a planner would schedule Aircraft Type 1 next. Another possibility for a tie-breaker is to choose the most rested unit, or other considerations and commander's preferences.

This tasking of air assets is closely related to a machine shop with the following features: unrelated machines in parallel, precedence relationships, and each job can only be processed on a subset of the available machines. The objective is to minimize the makespan (the amount of time it takes to process all the jobs). This suggests a heuristic to generate an initial solution for tasking allocated attack aircraft to targets. The heuristic is a combination of the LNS rule and the LFM-LFJ rule (referred to as the LNS-LFM-LFJ rule). The LNS-LFM-LFJ rule provides a sound "first cut" initial solution.

Before the steps of the heuristic can be outlined, there is another important consideration, that of *lag times*. In the earlier example, a SAM site should be downgraded before subsequent targets in the area (or deeper targets where the assigned missions must fly within range of the SAM) can be attacked. For this research, predecessor missions must attack before successor missions are permitted to cross the FLOT (crossing the FLOT is known as egress on the outbound leg and ingress on the inbound leg of the mission). This gives rise to a minimum lag time which may constrain the take-off time for successor missions. This concept is illustrated with a simple example (Table 10).

Table 11

	Take-off to FLOT (minutes)	FLOT to Target
Predecessor (Mission #1)	60	15
Successor (Mission #2)	85	15
Successor (Mission #3)	55	15

Table 11 contains sample missions and mission times. If Mission #1 takes off at 0000, it would strike its target at 0115 (75 minutes later). Missions 2 and 3, since they are successors to Mission #1, cannot ingress until this time. The time from base to FLOT for Mission #2 is 85 minutes, so no lag would be necessary. However, Mission #3 would need to delay its takeoff by 20 minutes so that it would ingress no earlier than (NET) 0115.

To be more precise, the above example is a demonstration of generalized precedence constraints. To ensure that the successor missions do not ingress before the predecessor hits its target, two generalized precedence constraints must be defined, one between each predecessor/successor pair, (1,2) and (1,3). Let S_i be the start time for mission *i*. The constraints may be formulated as follows:

$$S_2 \geq S_1 - 10$$

$$S_3 \geq S_1 + 20.$$

Negative 10 and 20 are the notional lag values associated with the general precedence constraints. Suppose Mission 1 takes off at 0800. The constraints on the successors are found as follows:

$$S_2 \geq S_1 - 10 \rightarrow \quad S_2 \geq 0800 - 10 \rightarrow \quad S_2 \geq 0750$$

$$S_3 \geq S_1 + 20 \rightarrow \quad S_3 \geq 0800 + 20 \rightarrow \quad S_3 \geq 0820.$$

Observe that the start times of the successors are constrained by the start time of the predecessor. Mission 2 is constrained to take off no sooner than 10 minutes prior to the start time of Mission 1, while Mission 3 is constrained to start no sooner than 20 minutes after the start of Mission 1. The formulation of these generalized precedence constraints can get considerably more complex with the addition of multi-modal activities.

Consider a successor target that must be attacked no later than (NLT) a specific number of minutes after its predecessor. For example, suppose a SAM site can become operational again an hour after being struck with a particular weapon. In this case, a successor must egress within at most 60 minutes of the TOT of the predecessor. This illustrates the concept of a maximum lag time that may either delay the take-off of the predecessor or hasten the take-off of the successor. Taken together, the minimum and maximum lag times make up a time window for the successor mission.

The technique developed in this research incorporates lag times to model staggered missions and suggests take-off times. This automated feature is not currently available within CTAPS. The suggested times may be changed by the unit as long as campaign objectives are met.

If a target with many successors were assigned a mission with a late TOT, then a condition known as a *bottleneck* might occur. That is, all of the successor missions would be "on hold" until this target is attacked. For this reason, the initial solution heuristic assigns missions with short base-to-target duration times to targets with many successors. Finally, to be a good initial solution for the GP TS, the heuristic strives for a solution that satisfies the goals of the GP. Therefore, the heuristic presented below considers the targets in priority order.

Step 1. Starting with the highest priority unscheduled targets, order the appropriate modes for each target by the LNS rule.

Step 2. Within priorities sort the modes first by LFM (least flexible mission) then by LFJ (least flexible job or target).

Step 3. Further sort the modes by ascending mission duration times. This ensures that the modes with the quickest completion times are paired up with the targets that have the most successors. This step will eliminate most bottlenecks.

Step 4. Schedule each target in the resulting order (from steps 1, 2, and 3).

Step 5. After scheduling the highest priority unscheduled targets, repeat steps 1, 2, 3, & 4 for the next highest priority targets. Continue until all targets are scheduled or until there are no more assets available.

This heuristic is a "pseudo-GP" as it maximizes coverage of each target priority (goal) in descending order. In other words, the heuristic attempts to schedule missions against all first priority targets before moving on to second priority targets. It then attempts to schedule all second priority targets before considering third priority targets. Finally it tries to schedule all third priority targets. It provides a feasible, good starting solution for the TS application of the GP model. It does not explicitly consider makespan. Makespan is implicitly captured via the LFM-LFJ heuristic which is a heuristic for minimizing makespan. The sorting of mission duration times was done to attempt to eliminate situations where missions are waiting for long periods for predecessors to attack their targets (a bottleneck). This simple heuristic corresponds with the common sense planners have used for years

3.2 GP Model for the Air Campaign Planning Problem

For the examples tested in this research, there are three target priority categories: Priority 1 (P1), Priority 2 (P2), and Priority 3 (P3), although the procedure developed in this thesis is not limited to three priority classes. The commander may desire more target priority classifications or sub-classifications, which can be incorporated into the model. Moreover, the commander may re-arrange the goals so that they correspond to his operational objectives (makespan may be more important than target coverage, for instance). The target priority classifications may be preemptive or non-preemptive. Likewise the sub-classifications within a priority class may be preemptive or be differentially weighted.

For this GP model, the three priorities are assumed to be preemptive. In other words, the most important goal is to schedule missions against the Priority 1 (P1) targets, to the exclusion of all other considerations. A schedule that assigns missions to all of the P1 targets and none of the P2 or P3 targets is considered "better" than a schedule that assigns missions to all of the P2 and P3 targets and all but one of the P1 targets (see Table 5, §3.5). The objective function for the LGP model of this research minimizes the deviation from complete target coverage for each of the target priority classifications in order, and then minimizes the makespan of the schedule (see Figure 14). The objective function, Equation (20), minimizes the deviation from the goals in lexicographic order. It ensures that P_i targets are considered before P_{i+1} targets are considered ($i = 1,2,3$) and that complete target coverage is attempted before trying to minimize the makespan. Equations (21), (22), and (23) give the constraints on the deviations, d_i. It is possible that all of the x_{imt} variables equal zero for activity i, making the d_{xi} variable equal to one. This condition would occur for targets that have no missions assigned to them; obviously no resources would be assigned to these targets. The d_i variables are therefore constrained to be equal to the number of priority i targets that do not have missions scheduled against them in the optimal solution. The rest of the model is identical to Van Hove's

MMGRCPSP formulation (Figure 12) except that his objective function becomes the constraint in Equation (24) that defines Goal 4, minimizing the makespan of the generated schedule.

Table 12. Parameters and Variables for the MMGRCPSP GP Formulation

Parameters:	
A_1	the set of all priority one targets
A_2	the set of all priority two targets
A_3	the set of all priority three targets
A	the set of all activities ($A_1 + A_2 + A_3$)
d	the index of the terminal activity
e_{im}	the earliest completion time for activity *i in mode m*
l_{im}	the latest completion time for activity *i in mode m*
M_i	the set of all execution modes for activity *i*
τ_{im}	the duration of activity *i* in mode *m*
S_i	the set of generalized successors of activity *i*
Δ_{ijmn}	the minimum lag between the start time of activity *i* in mode *m* and the start time of activity $j \in S_i$ in mode *n*
K	the set of all double constrained resources
r_{imk}	the amount of resource *k* required by activity *i* when being executed in mode *m*
R_k	the per period availability of resource *k*
N_k	the total amount of resource *k* available
g	the deadline for the project under consideration
Variables:	
x_{imt}	= 1 if activity *I* starts in period *t* and is executed in mode *m, 0 otherwise*
P_i	The weight of goal *i,* with $P_i >>> P_{i+1}$
d_i	# of Priority *i* targets not covered, $i = 1,2,3$
d_4	The makespan of the schedule

Minimize $\quad P_1d_1 + P_2d_2 + P_3d_3 + P_4d_4$ (20)

Subject to: $\quad d_{1i} + \sum_{m\in M_i}\sum_{t=e_{im}}^{l_{im}} x_{imt} = 1 \quad \forall i \in A_1$ and $d_{1i} \in \{0,1\}\ \forall i \in A_1$ and $d_1 = \sum_{i\in A_1} d_{1i}$ (21)

$d_{2i} + \sum_{m\in M_i}\sum_{t=e_{im}}^{l_{im}} x_{imt} = 1 \quad \forall i \in A_2$ and $d_{2i} \in \{0,1\}\ \forall i \in A_2$ and $d_2 = \sum_{i\in A_1} d_{2i}$ (22)

$d_{3i} + \sum_{m\in M_i}\sum_{t=e_{im}}^{l_{im}} x_{imt} = 1 \quad \forall i \in A_3$ and $d_{3i} \in \{0,1\}\ \forall i \in A_3$ and $d_3 = \sum_{i\in A_1} d_{3i}$ (23)

$$\sum_{m\in M_d}\sum_{t=e_{dm}}^{l_{dm}} (t+\tau_{dm})x_{dmt} - d_4 = 0 \tag{24}$$

$$\sum_{i\in A}\sum_{m\in M_i}\sum_{t=e_{im}}^{l_{im}} r_{imk}x_{imt} \le N_k \quad \forall k \in K \tag{25}$$

$$\sum_{i\in A}\sum_{m\in M_i}\sum_{t=j-\tau_{im}+1}^{j} r_{imk}x_{imt} \le R_k \quad \forall k\in K, j = 1 \ldots g \tag{26}$$

$$\sum_{n\in M_j}\sum_{t=e_{jn}}^{l_{jn}} (\Delta_{ijm'n} - t)x_{jnt} + \sum_{t=e_{im'}}^{l_{im'}} t x_{im't} \le \sum_{m\in(M_i\setminus m')}\sum_{t=e_{im}}^{l_{im}} g x_{imt} \quad \forall\, i\in A,\ \forall\, j\in S_i, \tag{27}$$

$$\sum_{m\in M_i}\sum_{t=e_{im}}^{l_{im}} x_{imt} = 1 \quad \forall i \in A \tag{28}$$

$$x_{imt} \in B \tag{29}$$

Figure 14. The MMGPRCPSP Goal Programming Model

3.3 Tabu Search for the Air Campaign Planning Problem

TS implementation is problem-specific. The details of the TS for this research are provided in this section.

3.3.1 Solution Representation

Each legitimate combination of aircraft number, aircraft type, base (unit), and target was given a code in the notional database. For example, 2T1B11 means 2 Type 1 aircraft from Base B (here aircraft type also determines the unit) against target 11. These solution codes were sorted by the SQL query that created them in accordance with the initial solution heuristic. This pre-sorting allows the heuristic to perform quickly since no sorting is done while the application is running. These codes are representations of the decision variables in the model formulation.

Associated with this resulting column of codes is a matrix of information (Table 13) which is the primary output of the application. The first column in the matrix consists of zeroes and ones; if combination *i* is in the solution, then element *i* in this *solution vector* is set to one. On the other hand, if combination *i* is not in the solution, then element *i* in the solution vector is set to zero. It is this solution vector that is acted upon by the TS move. The remaining columns in the solution matrix disclose the target priority classification, and convey mission timing information such as take off time, leg completion times, and mission completion time. The columns associated with timing are updated with every move.

Table 13 shows a few rows of the solution matrix. For example, the first row states that 4 aircraft of notional mode Type 1 is taking off at time zero from base B. It is scheduled to hit target #26, a Priority 1 target, at time 28. The aircraft associated with the mission will be available for assignment again at time 107. The last column tells which targets are covered. Since the mission in the first row attacks target #26, rows corresponding to all missions that could attack target #26 have the value in this column set to one (observe, for example, row 2). The matrix contains other columns such as time off target and landing time, that, for the sake of brevity, are not shown in the example.

Table 13. **Partial Example of Solution Matrix**

Solution Vector	Solution Code	Take Off	Time on Target	Completion Time	Target Priority	Is Target Scheduled? 1 if yes, 0 if no
1	4T1B26	0	28	107	1	1
0	4T1A26	0	0	0	1	1
1	2T1A53	0	36	123	1	1
0	2T1B53	0	0	0	1	1
0	4T2B26	0	0	0	1	1
0	4T2C26	0	0	0	1	1
1	4T1A54	10	46	133	1	1
0	4T1B54	0	0	0	1	1
1	2T2B27	3	29	126	1	1
0	2T2C27	0	0	0	1	1
0	2T2B54	0	0	0	1	1
0	2T2C54	0	0	0	1	1
1	4T1A3	0	24	99	1	1
0	4T1B3	0	0	0	1	1
1	4T1B80	0	60	171	1	1
0	4T1A80	0	0	0	1	1
1	2T1A7	0	21	93	1	1
0	2T1B7	0	0	0	1	1
0	4T2B3	0	0	0	1	1
0	4T2C3	0	0	0	1	1
1	2T2C76	0	36	143	1	1
0	2T2B76	0	0	0	1	1

3.3.2 Move Definition

Glover states that a good TS move for a particular application can only be determined through experimentation [Glover and Laguna, p.80, 1993]. One common move type is a *pairwise interchange* (also known as a *swap* or *2-opt* move). For a problem with binary decision variables, this move consists of turning off one variable (setting it to 0 if it was 1) while turning on a different variable (setting it to 1 if it was 0).

Another move type, and the one chosen for this research, is the *toggle*. Toggling consists of taking one of the elements in the binary solution vector and switching it to the opposite position; this may be the easiest type of move to implement. Employing a toggle will allow targets that were not initially scheduled to rotate into the ATO.

To demonstrate the toggle move, consider the solution vector (1,0,0,1,1) to a problem with five binary decision variables. The neighbors to this solution using the toggle move are (0,0,0,1,1), (1,1,0,1,1), (1,0,1,1,1), (1,0,0,0,1) and (1,0,0,1,0). It is easy to see that there are *n* neighbors for any solution of an *n*-variable problem. While the toggle move may seem simplistic, successive toggles achieve results similar to other move types. For example, two applications of the toggle move, where one move toggles a variable off (sets it to 0) and the subsequent move switches another variable on, has the same result as a swap. The drawback to using the toggle move is that it takes more iterations to achieve the same results as a more complex move. A move is chosen from a neighborhood if it is the most improving move or the least unimproving move and it is not on the tabu list (unless it meets the aspiration criterion).

3.3.3 Illustrative Neighborhood for Air Campaign Scheduling

To demonstrate the evaluation of a neighborhood of a solution for the GP model of the air campaign scheduling problem, suppose Table 14 displays five neighbors to some current solution. The goal deviations for each priority are given in the table. Solution #1, with a goal deviation vector of (1,0,0,1000) is the least desirable choice because it leaves one P1 target unassigned. Since the remaining neighbors each have a P1 deviation of zero, P2 deviations are examined. Only solutions 4 and 5 cover all of the P2 targets, and these are examined for the best P3 deviation. Solution #5 has the best P3 deviation and thus would be selected as the new solution.

Table 14. Illustration of Improving Solutions for a GP Model

Solution #	Goal 1 Deviations/ (P1 Targets Covered)	Goal 2 Deviations/ (P2 Targets Covered)	Goal 3 Deviations/ (P3 Targets Covered)	Makespan of Schedule
1	1/(59)	0/(25)	0/(15)	1000
2	0/(60)	25/(0)	15/(0)	1000
3	0/(60)	1/(24)	0/(15)	1000
4	0/(60)	0/(25)	14/(1)	1000
5	0/(60)	0/(25)	5/(10)	1100

Direction of Improving Solutions

If the best known solution had a deviation vector of, say, (0,0,4,1000) then no new best solution is recorded. On the other hand, if the best known deviation vector was (0,0,5,1200), then the makespans (goal 4) of the solutions would be compared. Since the new solution has a smaller makespan, then a new best known solution would be recorded.

3.4 *Implementing TSACP*

This section describes the TSACP application starting with the initial solution heuristic and continuing through an iteration of the tabu search. Flowcharts are provided that portray the overall process as well as details on the specific components.

Before the initial solution method is invoked, the application makes its calls to the database to retrieve and store the data required to solve the planning problem instance. The data is stored as objects which are easily passed from method to method, as necessary, so that multiple calls to the database are unnecessary. After the planning data is retrieved, a dialog box appears and the user selects from either the planning or the replanning applications. If the user selects "Replan" then an additional call is made to the database to retrieve the solution vector for the current plan. This solution vector is used for comparison against the replanning solution vector (discussed more fully in Sections 3.4 and 3.5).

3.4.1 Initial Solution Application

After all of the essential information is retrieved and stored, the *Initial Solution* method is invoked. The primary features of this method are: 1) a loop that repeats until either all of the targets have a mission scheduled against them or no further missions can be planed with the remaining resources and 2) a *timeWindow* array for each unit of length equal to the absolute latest completion time of the ATO and 3) the array of solution codes discussed in Section 3.3.1. Each element of *timeWindow* is initially set equal to the number of attack aircraft allocated to the unit.

Within the method are a series of loops that check to see if, for an element *i* of the array of solution codes, the target associated with code *i* has a mission assigned to it. If no mission is assigned to that target, a check is made first to see which, if any, predecessor missions are already scheduled. If there are any, then *earliestStartTime* for the mission under consideration is set to the maximum of the start time of all assigned predecessors plus their associated lag times. A check for available aircraft times is then made against *timeWindow.* This check determines at what point (starting at *earliestStartTime*) there are sufficient consecutive elements (equal to the duration of the mission plus turn time) of the array such that the number of aircraft available is greater than or equal to the number of aircraft required for the mission. If a time window is found that does not violate any successor constraints, the mission is scheduled with a take-off time equal to the index of the first element *t* of the time window. The mission leg times are set equal to $t + d_{leg}$ (where d_{leg} is the leg duration for the mission as specified in the database). If no suitable window is found, the mission is not assigned and code $i + 1$ is inspected in like manner. Figure 15 is a flowchart illustrating this process.

Notional example:

Unit 1A has 4 attack aircraft assigned to it, its initial *timeWindow* array is given in the second row of the following table:

Time period	0	1	2	3	4	5	6	7	8	9	10	11	12	…
Available a/c	4	4	4	4	4	4	4	4	4	4	4	4	4	…

The unit is considered for assignment to some target, *a*. If this target has no predecessors, the mission can start at time 0. If the mission has a completion time of 4 and requires two aircraft, then *timeWindow* becomes:

Time period	0	1	2	3	4	5	6	7	8	9	10	11	12	…
Available a/c	2	2	2	2	4	4	4	4	4	4	4	4	4	…

Next, the unit is considered for assignment to target *b* which is a successor to *a*. The *a/b* mode combination selected has a minimum lag of 2 and 2 aircraft are required to fly the mission, which has a completion time of 4. The resulting *timeWindow* becomes:

Time period	0	1	2	3	4	5	6	7	8	9	10	11	12	…
Available a/c	2	2	0	0	2	2	4	4	4	4	4	4	4	…

Examination of the array shows that no missions requiring 3 or 4 aircraft can take off before time period 6 (the first element of the array is time 0). Additionally, missions requiring 1 or 2 aircraft can take off at time period 4 and, if the completion time is by the end of the second time period, such a mission could take off at time 0, assuming no precedence relationship violations.

The *timeWindow* array is used only to keep track of the renewable resource, per time period aircraft utilization. The non-renewable resource, total sorties, is tallied elsewhere. It is assumed for this research that a unit will not be considered for assignment if it has flown all of its allocated sorties. Other mission attributes, such as leg times, are recorded in the solution matrix. The output of this method is a feasible solution to the planning (replanning) problem.

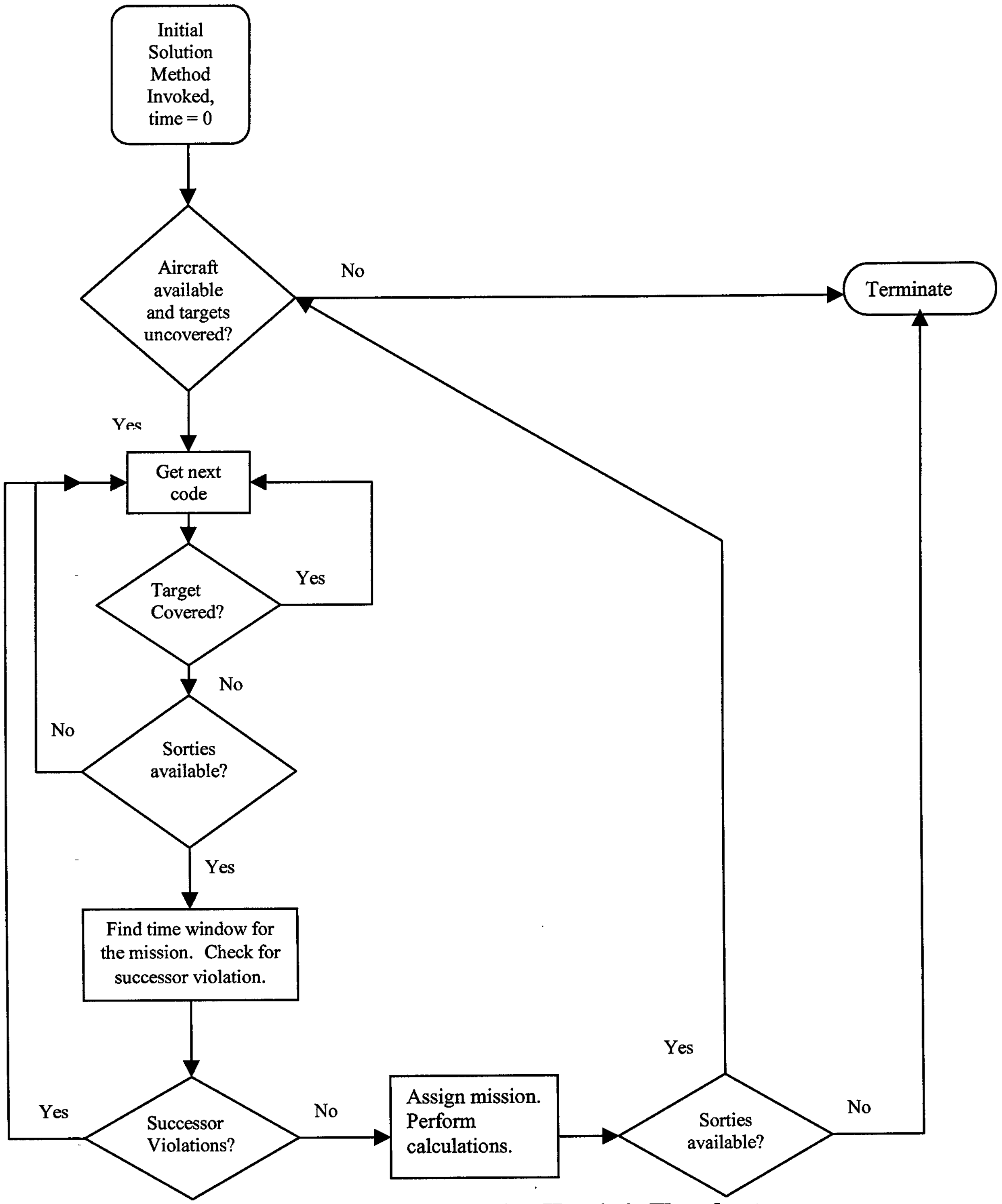

Initial Solution Method Invoked, time = 0
Aircraft available and targets uncovered?
No
Terminate
Yes
Get next code
Target Covered?
Yes
No
No
Sorties available?
Yes
Find time window for the mission. Check for successor violation.
Yes
Yes
Successor Violations?
No
Assign mission. Perform calculations.
Sorties available?
No

Figure 15. Initial Solution Heuristic Flowchart

Initial Solution Evaluation

The *Evaluate* method is used to appraise the output of the *Initial Solution* method to see how well it achieved the goals of the objective function. When the *Initial Solution* was invoked, an array, *numTarg* initialized at zero, was created with length equal to the number of targets in the problem instance. When a mission is scheduled, the element of *numTarg* corresponding to the target associated with the mission is set to 1.

To judge how well the initial solution covers the targets, the *Evaluate* method simply scans the *numTarg* array for zeros. When a zero is encountered, the priority of its associated target is obtained and the deviation vector of the corresponding goal is incremented by one. For example, if a zero is encountered at $i = 20$, then the priority classification of target # 19 is retrieved (the first element of an array is zero, hence target # equals i - 1). If target # 19 is Priority k, then the deviation vector for goal k is incremented.

Goal four in the planning problem is to minimize makespan. To determine the makespan, the *Evaluate* method passes the "Mission Duration" column of the solution matrix to the *Maxim* method. This method returns the value of the largest integer in the array.

The output of the *Evaluate* method is an array of four integers. For instance, if the initial solution covered all of the P1 targets, all but one of the P2 targets, and all but five of the P3 targets, while achieving a makespan of 550 minutes, then the objective function vector would be [0,1,5,550].

3.4.2 Tabu Search Walkthrough

The initial solution heuristic provides a robust solution that can be used on its own, but is intended as a good starting point for TSACP. Parameters for TSACP such as tabu tenure and total number of iterations must first be specified. The aspiration criterion is the best objective function value found. The initial solution is passed to the tabu search engine and TSACP starts.

3.4.2.1 Move Manager

The first consideration for TSACP is the move neighborhood. In other words, a decision must be made concerning which of the missions in the current solution are eligible for toggling; the set of missions that are considered eligible is called the *candidate list*. By restricting the candidate list to those missions that stand a good chance of improving the objective function, the neighborhood of the current solution will be smaller, there will be fewer moves to evaluate in each iteration, and the tabu search will be faster.

The *Move Manager* method selects the candidate list. This method considers missions associated with only one specific target category at a time for inclusion on the candidate list. The target category under consideration alternates at each iteration; only P1 targets may be considered at iteration 1, P2 targets for iteration 2, and P3 targets for iteration 3, then the cycle repeats. The reasoning behind this strategy is: if targets from all categories were included on the candidate list and an unimproving move was necessary (a mission must be turned off), then only missions associated with P3 targets would be selected. The tabu search would never select a mission to a P2 target or P1 target to toggle off if a mission to a P3 target was available. In essence, the candidate list would then only consist of all missions to P3 targets and only those P1 and P2 targets that do not currently have missions assigned to them. This would result in little or no inter-modal swaps and an overly restrictive neighborhood.

The Move Manager selects missions to P1 targets for inclusion on the candidate list if one of the following criteria is met: either no mission is currently assigned to the associated target (a mission must be toggled on, thereby improving the objective function) or, if the mission under consideration is an assigned mission (and would be toggled off), its mission completion time is equal to the makespan of the schedule. A move that selects a mission to a P1 target to be toggled off is the worst kind of unimproving move (the deviation variable for goal one would increase by one). Therefore allowing this type of move to be included in the candidate list must require a very good reason. Selecting (for toggling off) a mission to a P1 target only if it is associated with the makespan decreases the size of the candidate list while allowing for the possibility to decrease the makespan. The decision maker can change the candidate list inclusion criteria if a goal interval is specified, for example.

Selecting move candidates among missions to P2 and P3 targets is less complex. A mission is placed on the candidate list if either the target associated with the mission has no mission assigned to it (the mission may be toggled on) or if the mission is an assigned mission and would be toggled off.

3.4.2.2 Move

Each element of the candidate list is operated upon by the *Move* method. First, the mission is toggled (turned off if it was previously on and vice versa). Next, the *Solution Modifier* method is called. This method's primary components are two IFblocks; if the mission is set to "on" the statements of the corresponding IF-block are executed, otherwise the statements of the other IF-block are performed.

This method performs quickly because there are no loops within it. It only calculates the marginal change resulting from the move under consideration. For example, consider a potential move where a mission is toggled off; its corresponding element in the solution vector is changed to zero. When this occurs, each of the corresponding elements in the timing columns are also set to zero. Additionally, the *timeWindow* array for the unit is updated. The final array in the example in Section 3.4.1, was:

Time period	0	1	2	3	4	5	6	7	8	9	10	11	12	...
Available a/c	2	2	0	0	2	2	4	4	4	4	4	4	4	...

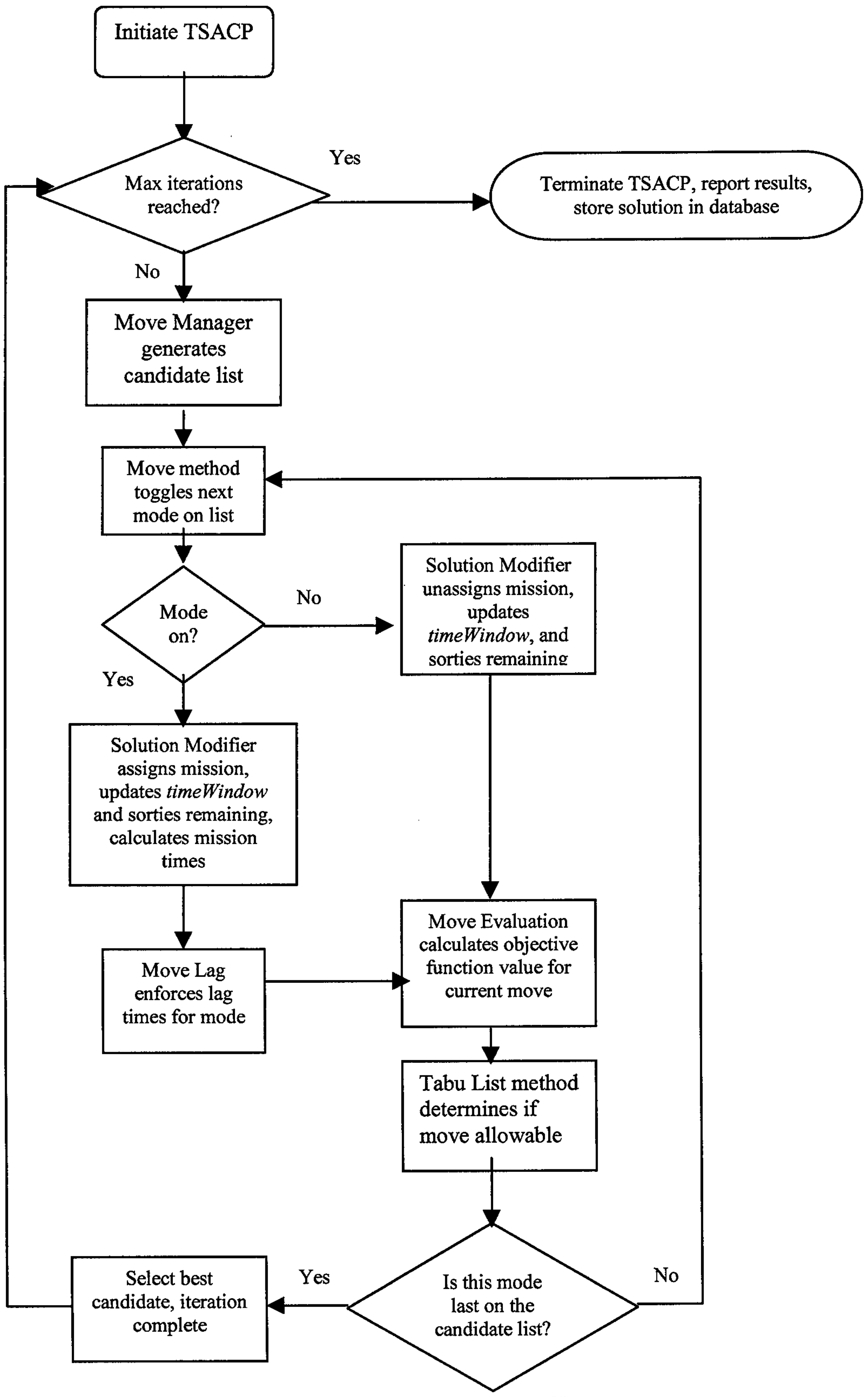
If the mission to target *a* was toggled off, the array would become:

Time period	0	1	2	3	4	5	6	7	8	9	10	11	12	...
Available a/c	4	4	2	2	2	2	4	4	4	4	4	4	4	...

Finally, the number of sorties flown by the unit is decremented.

For the circumstance where the mission was toggled on, the method checks to see if there are enough sorties left in a unit to fly that mission. If not, the method skips to the next mission on the list. If so, an attempt is made to locate a time window for the mission in a similar fashion as was done in the *Initial Solution* method. The difference during the tabu search is that if successor violations exist, the time window is assigned to the mission, a penalty is placed upon the objective function, and the next move is evaluated. This is done to keep the search from "bogging down" by looking overly long for time windows for any one move.

If a move is chosen that does violate any successor constraints, the successors that are in violation are the only moves allowed on the candidate list until all violations are reconciled. For example, say elements 5, 12, and 21 of the solution code array are successor missions and all three are violating precedence constraints. On the next three moves, only these three elements will be on the candidate list. Once these are all dropped from the schedule, TSACP can proceed choosing its candidates as before. Figure 16 is a flowchart that illustrates this process.



Initiate TSACP
Max iterations reached?
Yes
Terminate TSACP, report results, store solution in database
No
Move Manager generates candidate list
Move method toggles next mode on list
Mode on?
No
Solution Modifier unassigns mission, updates *timeWindow*, and sorties remaining
Yes
Solution Modifier assigns mission, updates *timeWindow* and sorties remaining, calculates mission times
Move Lag enforces lag times for mode
Move Evaluation calculates objective function value for current move
Tabu List method determines if move allowable
Select best candidate, iteration complete
Yes
Is this mode last on the candidate list?
No

Figure 16. TSACP Flow Chart

3.4.2.3 Move Evaluation

When the *Move* method toggles a mission to a P1 target on, a flag *changeInP1* is set to negative one. Similar flags are set to one for missions to P2 and P3 targets. If a mission was toggled off, then the value of the *changeInP*$_i$ variable is set to one. The value of the appropriate variable is sent to the *Evaluate Move* method; the value is then added to the corresponding element of the current objective function vector. For example, suppose the current solution has an objective function vector [0,1,5,550] and the current move consists of a mission to a P3 target being toggled off. *changeInP3* is set to one and *Evaluate Move* adds *changeInP3* to the third element of the objective function vector giving the new objective function vector [0,1,6,550]. The makespan of the current move is evaluated as before by sending the mission completion time column of the solution matrix to the *Minim* method.

After the current move is evaluated, it needs to be inspected by the *Tabu List* method to determine if it is an allowable move. In other words, is the move tabu? If so, is the aspiration criterion met? If the move is deemed admissible, the index of the move and its objective function value are stored until all the moves on the candidate list are operated upon and evaluated in like manner. When this is accomplished, the best move is selected and a new iteration is begun. Figure 17 is a flowchart that illustrates the move evaluation method while Figure 18 is a flowchart of the overall process discussed in this section (3.4).

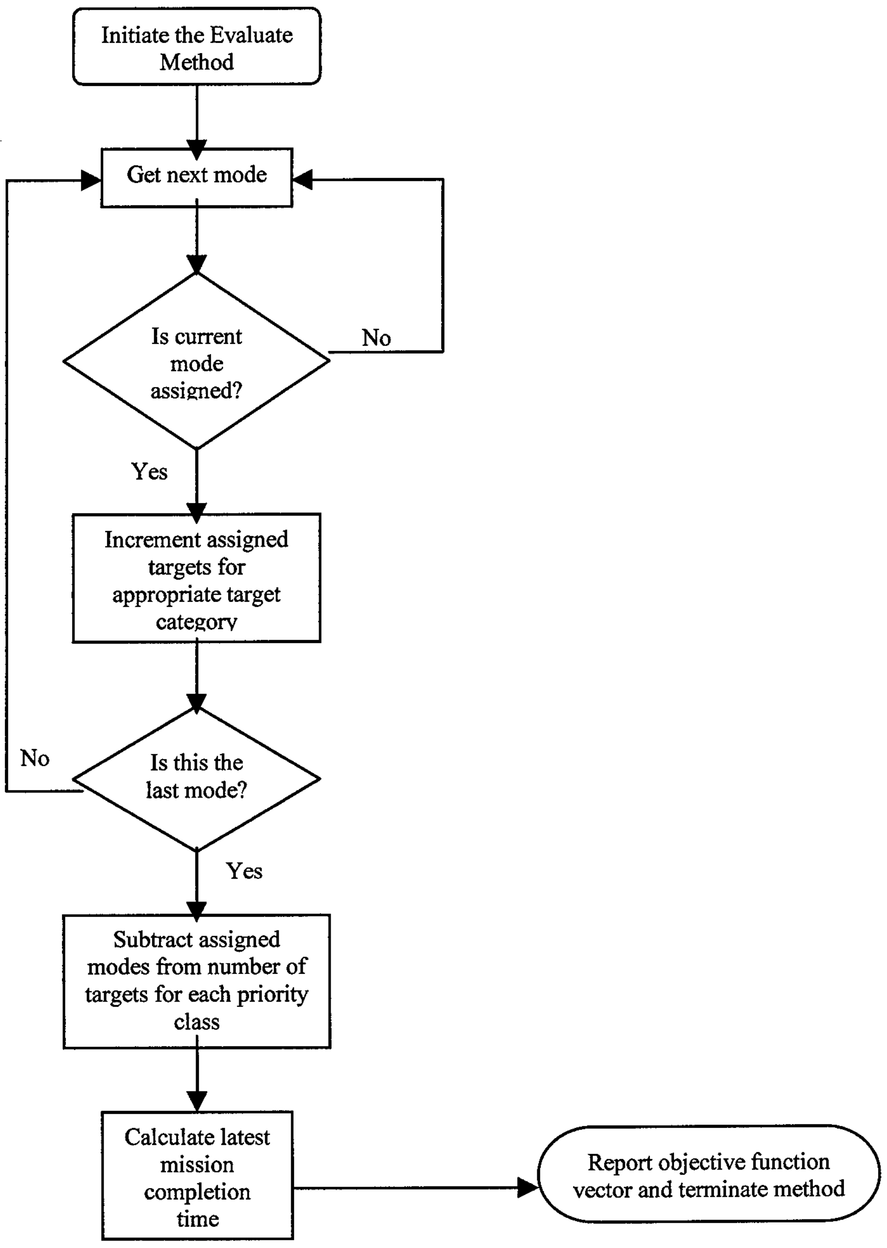

Initiate the Evaluate Method
Get next mode
Is current mode assigned?
No
Yes
Increment assigned targets for appropriate target category
Is this the last mode?
No
Yes
Subtract assigned modes from number of targets for each priority class
Calculate latest mission completion time
Report objective function vector and terminate method

Figure 17. Flow Chart for Evaluate Method

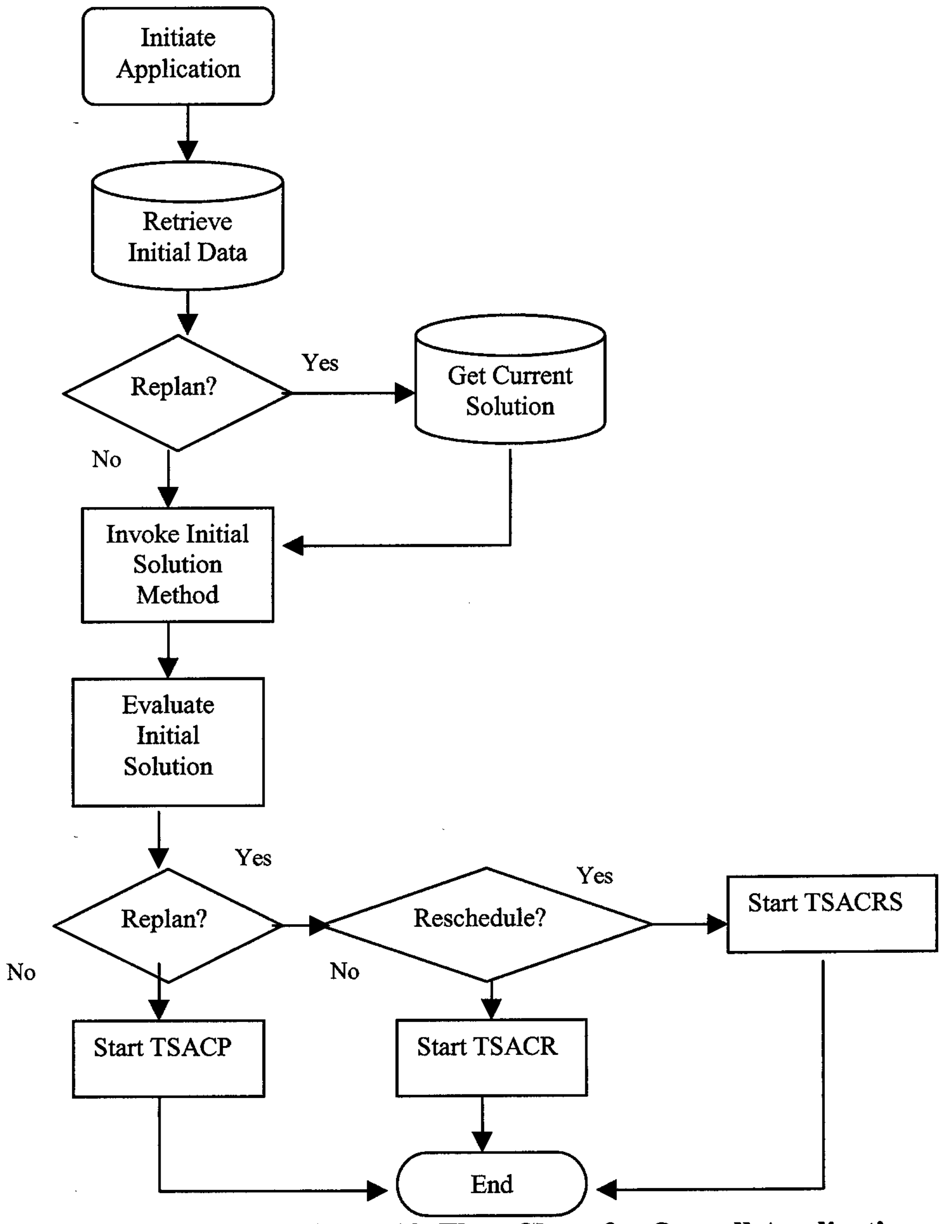


Figure 18. Flow Chart for Overall Application

3.5 Replanning

Replanning often occurs during the ATO process. The ABP is handed over to the Combat Operations Division approximately 10 hours before the start of the ATO day [AFI 13-1AOC V3, 1999]. This time is required for crew planning before the start of ATO execution. Should any changes occur to the ATO during this time, the missions affected by the changes may have to cope with a shortened mission preparation time. Therefore, the impact of any change should be minimized.

Suppose, for example, a high priority target originally scheduled for tomorrow must be attacked today instead. Assume that there are no more available sorties nor are there missions available that can "squeeze in" another target. In this case, a lower priority target may have to be rolled into the next day's ATO to free up assets to attack this new target.

3.5.1 LGP Model for Replanning

A slightly modified LGP model is appropriate for replanning. Consider the LGP for planning. It consisted of four goals:

Goal 1	Maximize P1 target coverage
Goal 2	Maximize P2 target coverage
Goal 3	Maximize P3 target coverage
Goal 4	Minimize makespan of the schedule

These goals, however, are not sufficient for the replanning application. An additional goal has been inserted between goal 3 and goal 4 (although it could be inserted wherever the commander feels it is appropriate). In order to minimize the impact of a change to the ATO, the number of missions affected should be minimized.

For example, suppose that by reassigning one mission (and its support package) currently scheduled against a P3 target, the new P1 target can be attacked. This might be operationally more desirable (although not necessarily mathematically more desirable) than a replanning schedule that shuffles several missions around while trying to minimize makespan. Therefore, a goal designed to minimize the number of missions changed is added as goal 4 (with the former goal 4 becoming goal 5). As the primary goals of target coverage remain unchanged, the heuristic developed for the planning application will be used to find the initial replanning solution. The heuristic would, in essence, find a new initial plan but with the current target and resource sets. Additionally, the planner may decide to lock in critical missions that employ extremely valuable or sensitive assets so that they cannot be changed (such as B-2 missions).

Goal 1	Maximize P1 target coverage
Goal 2	Maximize P2 target coverage
Goal 3	Maximize P3 target coverage
Goal 4	Minimize number of changed missions
Goal 5	Minimize makespan of the schedule

Baykasoglu stated that in real-world problems, the analytic form of some objectives may not be available [Baykasoglu, p. 960, 1999] or may be too time-consuming to be worthwhile. Constructing the analytical form for the new goal is an example of such a case. The mathematical model would require an initial vector of special ordered sets (x_{imt} variables) to use in its formulation of this new goal four. Such an objective would be difficult to formulate comprehensively.

However, if an LGP formulation were available for the replanning problem, it would be similar to that for the planning problem. The only difference would be the inclusion of a new variable in the objective function to account for the insertion of the goal to minimize the number of changed missions. Since reassigning one mission may be operationally desirable, this deviation variable has a target value of 1 and so the constraint, $d_4 \geq 1$ (or $1 - d_4 \leq 0$) would be included to define this deviational variable. This target value may be set by the decision-maker at a level of his or her choosing or it could be parameterized (i.e. a Goal Interval Program approach may be used). This means that solutions may be determined with varying values for d_4 to see if an acceptable trade-off of changed missions vs. makespan exists.

3.5.2 Tabu Search for Replanning

Since the theoretical formulation of the LGP model for replanning is similar to that for planning, the same move definition was used for the replanning TS, the toggle move. Likewise the tabu tenure and aspiration criterion are identical. The only change that needed to be made to the TSACP application to obtain the Tabu Search for Air Campaign Replanning (TSACR) was to incorporate the fifth deviational variable and its corresponding constraint.

When TSACP was used to solve each planning problem instance, a copy of the best solution obtained was saved to the database. For TSACR, an initial solution based on the current scenario situation is obtained using the LNS-LFM-LFJ heuristic. This initial solution is evaluated just as in TSACP, but with one difference. The solution vectors from the TSACP best solution and the TSACR initial solution are compared to one another. Each time the value of element *i* in one vector differs from the value of element *i* in the second vector, the value of the deviational variable associated with goal 4 is incremented by one. As the TSACR progresses, at each iteration every neighbor to the current solution is likewise compared to the TSACP best solution; the TSACR attempts to minimize this value as well as the values for the other goals.

To illustrate this vector comparison, consider Table 15. The first column in this notional table consists of the binary solution vector from the saved TSACP best solution for a problem instance, while the second column is the solution vector for some neighbor to the current solution of the replanning problem. As the two vectors are compared, the cumulative value of the deviational variable for goal 4 is displayed in column 4 of the table.

Table 15. TSACP vs. TSACR solution vector comparison

TSACP best solution	TSACR current solution	Comparison	Cumulative value of goal four
1	1	equal	0
0	0	equal	0
0	0	equal	0
1	1	equal	0
0	1	unequal	1
1	0	unequal	2
0	0	equal	2
0	1	unequal	3
0	0	equal	3
1	1	equal	3
1	0	unequal	4
0	0	equal	4
0	0	equal	4
1	0	unequal	5

This approach allows for a rapid comparison of the replanning solution against the original plan. By minimizing the value of the deviational variable associated with goal four, the operational impact to the schedule is decreased. If this number is small, most of the missions in the ATO can fly as scheduled.

Only the differences between the TSACP and TSACR methods will be discussed. After the user selects "Replan" from the initial dialog box, the solution vector of the current solution is retrieved from the database and a Boolean variable in the *main* method is set to true. This Boolean variable tells the *Initial Evaluation* and the *Evaluate* methods that a new goal is to be inserted into the objective function vector. A conditional FOR-loop now is executed. Within this FOR-loop each element of the two solution vectors are compared and the goal four deviation vector is incremented by one every time the two respective elements are unequal. Goals one, two, and three are identical to TSACP; goal four of TSACP now becomes goal five of TSACR. All of the other methods discussed in Section 3.4 function as before.

3.6 Rescheduling

Rescheduling is defined in this research as re-assigning missions, as circumstances warrant, while the Air Battle Plan (ABP) is already in the state of execution. It differs from replanning by the amount of time planners have available to find a solution and, subsequently, by the amount of time that the pilots and crews have available to rearm (if necessary), develop new detailed mission routes and times, and view the imagery associated with a new target. This means that an application for rescheduling must work rapidly. The solution obtained should, if possible, impact the ATO to an even lesser extent than that for replanning.

The rescheduling application developed for this research reduces operational impact by liberally using a feature mentioned in §3.5—"locking in" missions. While a combat planner may selectively lock in specific missions, for this research all remaining assigned missions to priority one and two targets were locked, thus allowing only missions currently assigned to priority three targets to be considered for re-assignment. Should the magnitude of the rescheduling situation be such that reassigning all of the missions assigned to priority three targets not be sufficient to cover all priority one targets, some or all of the priority two targets could be unlocked.

The rescheduling problem is solved by using a modified TSACR, Tabu Search for Air Campaign Rescheduling (TSACRS). By locking in a quantity of missions, the value of the deviational variable corresponding to goal four is already quite small; thereby presenting a good starting solution for a tabu search for rescheduling. In addition, the candidate list becomes quite small, thus allowing the tabu search to function extremely rapidly. This extra speed may well be necessary in a combat rescheduling environment.

3.7. Summary

This chapter presented the methodology for solving the air campaign planning problem. Chapter 4 explains how the methodology was tested and the results of the testing. The chapter also contains a case study problem that resembles an air campaign planning scenario. The case study was used to demonstrate not only how the planning application would be used in an operational environment, but also to demonstrate how the replanning application would be used.

CHAPTER 4. Results

This chapter presents the outcome of applying TSACP to several MMGRCPSP problems having known optimal or lower bound solutions. In addition, the generation of a sample combat planning problem is described in detail and the results of applying the TSACP and TSACR applications are provided.

4.1 Problem Sets

The problem sets used for testing TSACP were generated using the ProGen problem generator [Kolisch, *et al*, 1992, 1995]. After the problem sets were generated, they were put in a format that could be imported into the Access database and used by TSACP. A total of five 10-job problems, three 30-job problems, three 60-job problems, and two 90-job problems were used for testing.

The test problems are analogous to the air campaign planning problem; jobs are targets, resources are units, and activities are missions. Each of the jobs in the 10-job problems could be executed in three different modes; each mode uses different quantities and types of resources and has different processing times. The 30- 60- and 90- job problems consisted of jobs that are executed in a particular mode from a choice of one, two, or three possible modes. Again, each mode uses different resources and processing times. The doubly constrained resources among the problem sets each carried varying quantities of renewable and non-renewable assets. The problems within each set also differed in their network configuration and complexity. The number of successors and predecessors for each job (target) were varied among the problems to give an array of network structures.

The lag times between a predecessor/successor pair depends on the mode combination being used. For example, predecessor job J1 executed in mode A followed by successor job J2 executed in mode A would have a different lag time than predecessor job J1 executed in mode A followed by successor job J2 executed in mode B. Random maximum (negative value) or minimum (positive value) lag times were generated for each predecessor/successor mode combination. The lag times were uniformly distributed, randomly generated integers restricted to be in the range from negative one-half the duration of the predecessor activity to the duration of the predecessor activity. For example, if the predecessor activity had a duration of 10, then the lag times associated with this activity would be in the interval [-5, 10].

Due to the computational burden coupled with limited time and resources, optimal solutions were unavailable for all but one (Van Hove's small 10-job problem, pp. 73 –75) of the 10-job problems and none of the larger problems. However, relaxed optima, using only the non-renewable resource constraint, were obtained for the remaining 10-job problems and for one of the 30-job problems. The true optima for the doubly constrained problems cannot be less than the relaxed solution, hence the relaxed optima serve as lower bounds on the fully constrained problems. For the first phase of testing of the initial solution heuristic and the general tabu search heuristic, goals were not considered. The objective in these tests was to minimize makespan. Goals are considered in the case study, however.

The remaining 30-job, 60-job, and 90-job problems had no obtainable lower bounds within the limited time and resources available. Results of TSACP for these jobs are reported in this research for two primary reasons: first, to determine how problem size affects solution time and second, to measure the performance of the initial solution heuristic vs. the tabu search.

4.2 TSACP Results

Barr, *et al*, suggest the following experimentation steps for heuristics [Barr *et al*, page 11-28, 1995]:

1. Define the goal of the experiment.
2. Choose measures of performance and factors to explore.
3. Design and execute the experiment.
4. Analyze the data and draw conclusions.
5. Report the experiment's results.

The goal of the testing for this research is to show that the tabu search and initial heuristic perform quickly and provide reasonable solutions when compared to a lower bound on the makespan for the problem. Barr *et al*'s remaining steps are explained as this chapter progresses.

The relaxed (no renewable constraint) integer programs were solved using HyperLingo® Version 5.0 on a Pentium II with a clock speed of 400MHz and 256 megabytes of memory. TSACP was run on a Pentium II with a clock speed of 300MHz and 128 megabytes of memory. Only makespans were compared for the test problems. The times reported for TSACP were all for 1000 iterations and a tabu tenure of 13. As there was potential value inherent in the initial solution heuristic itself, solution times and values were recorded for the initial heuristic as well as for TSACP.

The results are presented in Tables 16 through 19. The statistics for the 10-job problems are reported in Table 16 where the "delta" column is the raw difference between the solution obtained via the specified heuristic approach and the lower bound. The "% delta" column is the percentage away from the lower bound for the specified heuristic (i.e. (delta/lower bound)%). Table 19 contains data including minimum, maximum, and mean solution times as well as the standard deviations and confidence intervals for each size problem.

Table 16. Solution Statistics for 10-Job Problems

Job Size	Method	CPU time (seconds)	Cmax	delta	% delta	jobs not scheduled
10						
Problem 1	Lower Bound	83.00	10	--	--	0
	Relaxed Initial Soln	0.03	10	0	0%	0
	Relaxed TS	10.80	10	0	0%	0
	Fully Const. Init. Soln.	0.03	13	3	30%	0
	Fully Const. TS	14.90	11	1	10%	0
Problem 2	Lower Bound	64.00	11	--	--	0
	Relaxed Initial Soln	0.03	19	0	73%	0
	Relaxed TS	11.20	11	0	0%	0
	Fully Const. Init. Soln.	0.02	27	16	145%	0
	Fully Const. TS	11.40	15	4	36%	0
Problem 3	Lower Bound	569.00	9	--	--	0
	Relaxed Initial Soln	0.03	11	2	22%	0
	Relaxed TS	12.00	11	2	22%	0
	Fully Const. Init. Soln.	0.03	18	14	78%	0
	Fully Const. TS	11.00	18	9	78%	0
Problem 4	Lower Bound	81.00	9	--	--	0
	Relaxed Initial Soln	0.03	13	4	44%	0
	Relaxed TS	10.50	9	0	0%	0
	Fully Const. Init. Soln.	0.02	13	4	44%	0
	Fully Const. TS	10.60	9	0	0%	0
Problem 5	Lower Bound	182.00	9	--	--	0
	Relaxed Initial Soln	0.02	18	9	100%	0
	Relaxed TS	10.50	11	2	22%	0
	Fully Const. Init. Soln.	0.03	18	9	100%	0
	Fully Const. TS	12.50	12	3	33%	0
delta = heuristic solution – lower bound			% delta = (delta/lower bound)%			

Table 17. Solution Statistics for 30-Job Problems

Job Size	Method	CPU time (seconds)	Cmax	delta	% delta	jobs not scheduled
30	Lower Bound	2734.00	26	--		0
Problem 1	Relaxed Initial Soln	0.12	44	18	41%	0
	Relaxed TS	34.20	28	2	7.90%	0
	Fully Const. Init. Soln.	0.11	60	34	131%	1
	Fully Const. TS	36.80	57	31	119%	0
Problem 2	Relaxed Initial Soln	0.12	21			1
	Relaxed TS	27.00	23			0
	Fully Const. Init. Soln.	0.08	41			1
	Fully Const. TS	30.62	41			0
Problem 3	Relaxed Initial Soln	0.08	39			1
	Relaxed TS	29.50	23			0
	Fully Const. Init. Soln.	0.08	21			1
	Fully Const. TS	33.18	39			0

Table 18. Solution Statistics for 60-Job Problems

Job Size	Method	CPU time (seconds)	Cmax	jobs not scheduled
60				
Problem 1	Relaxed Initial Soln	0.86	32	0
	Relaxed TS	110.10	30	0
	Fully Const. Init. Soln.	0.67	40	0
	Fully Const. TS	137.40	40	0
Problem 2	Relaxed Initial Soln	0.26	32	0
	Relaxed TS	111.70	30	0
	Fully Const. Init. Soln.	0.26	60	0
	Fully Const. TS	130.00	32	0
Problem 3	Relaxed Initial Soln	0.26	32	0
	Relaxed TS	108.80	30	0
	Fully Const. Init. Soln.	0.26	50	0
	Fully Const. TS	111.20	50	0

Table 19. Solution Statistics for 90-Job Problems

Job Size 90	Method	CPU time (seconds)	Cmax	jobs not scheduled
Problem 1	Relaxed Initial Soln	1.57	29	2
	Relaxed TS	253.55	30	0
	Fully Const. Init. Soln.	1.64	46	3
	Fully Const. TS	264.30	46	0
Problem 2	Relaxed Initial Soln	.62	33	0
	Relaxed TS	227.90	30	0
	Fully Const. Init. Soln.	.54	44	0
	Fully Const. TS	255.60	41	0

For the 10-job problems, the relaxed TSACP achieved optimality three times, while the initial heuristic attained the relaxed optimum twice. On one occasion, the solution found by TSACP for the fully-constrained problem was equal to that of the relaxed optimum and therefore was the fully-constrained optimum. TSACP was able to improve the solution obtained with the initial heuristic three out of five times (on the other two occasions, the initial heuristic found the relaxed optimum and so TSACP could not improve its solution). TSACP improved the fully-constrained initial solution for all but one of the five 10-job test problems.

Only one lower bound was available for a 30-job problem. For this one problem, TSACP for the relaxed problem was within 8% of the optimum. In all but two of the larger problems, TSACP was able to improve the initial solution. For the cases where the initial solution heuristic failed to schedule all of the jobs, TSACP was able to find a solution where all jobs were scheduled, sometimes with a subsequent longer makespan, but since makespan is a lower priority goal, these solutions are still an improvement. All solutions were checked for feasibility.

Table 20. Solution Time Statistics

	Initial Solution	TSACP (1000 iterations)
10 Jobs*		
Min	0.02	10.4
Max	0.03	14.9
Mean	0.027	11.53
St. Deviation	0.00483	1.360596
95% Conf. Int.	(0.024, 0.029)	(10.686, 12.373)
30 Jobs*		
Min	0.08	27
Max	0.12	36.8
Mean	0.0983	31.65
St. Deviation	0.0204	3.475
95% Conf. Int.	(.082, 0.115)	(28.869, 34.431)
60 Jobs*		
Min	0.26	110.2
Max	0.861	137.4
Mean	0.429	118.2
St. Deviation	0.268	12.2727
95% Conf. Int.	(0.214, 0.643)	(108.38, 128.02)
90 Jobs*		
Min	0.541	227.9
Max	1.642	264.3
Mean	1.0934	250.338
St. Deviation	0.594	15.667
95% Conf. Int.	(0.270, 1.917)	(234.984, 265.691)
*As there was no statistical difference between the mean solution times of relaxed vs. fully constrained problems within a job size, the statistics are pooled for each job size.		

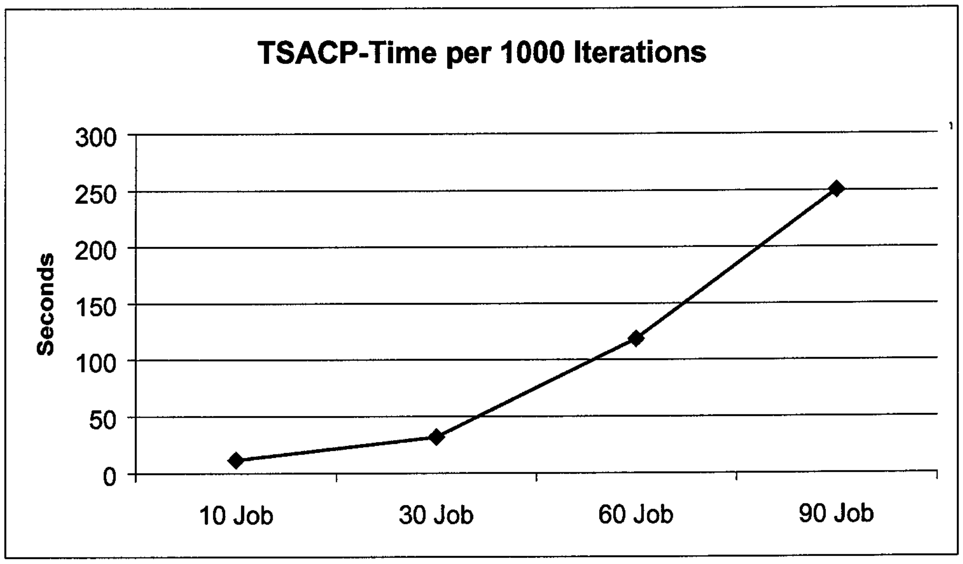


Figure 19. Graph of TSACP Time for 1000 Iterations

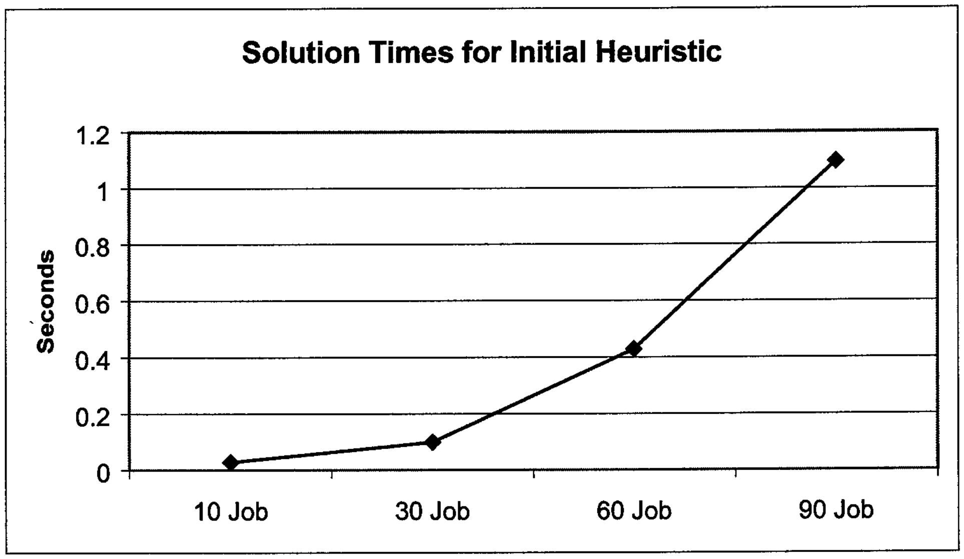


Figure 20. Graph of Solution Time for Initial Heuristic

Solution times for the test problems naturally increased as the problem size increased. Graphs of the TSACP (Figures 19 and 20) and the initial heuristic times show that they follow similar, although very different scaled, curves. The initial heuristic is very fast for the problem sizes tested. Regression analysis results are presented in Tables 20 through 23.

Table 21. Linear Fit for TSACP Solution Times (1000 Iterations)

sol time = -31.074 + 2.825 job size

Summary of Fit

RSquare	0.936
RSquare Adj	0.934
Root Mean Square Error	22.255
Mean of Response	77.587
Observations (or Sum Wgts)	26

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	175104.15	175104	353.550
Error	24	11886.60	495	Prob>F
C Total	25	186990.74		<.0001

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-31.074	7.242	-4.29	0.0003
job size	2.825	0.150	18.80	<.0001

Table 22. Quadratic Fit for TSACP Solution Times (1000 Iterations)

sol time = 7.839 – 0.009 job size + 0.030 job size^2

Summary of Fit

Rsquare	0.991
RSquare Adj	0.990
Root Mean Square Error	8.492
Mean of Response	77.587
Observations (or Sum Wgts)	26

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	185332.11	92666.1	1284.982
Error	23	1658.64	72.1	Prob>F
C Total	25	186990.74		<.0001

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	7.840	4.279	1.83	0.0799
job size	-0.009	0.245	-0.03	0.9726
job size^2	0.030	0.003	11.91	<.0001

Table 23. Linear Fit for Initial Heuristic Solution Times

Initial soltime = -0.170 + 0.012 job size

Summary of Fit

RSquare	0.652
RSquare Adj	0.637
Root Mean Square Error	0.270
Mean of Response	0.300
Observations (or Sum Wgts)	26

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	1	3.279	3.279	44.9407
Error	24	1.751	0.073	Prob>F
C Total	25	5.030		<.0001

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-0.170	0.088	-1.93	0.0651
job size	0.012	0.001	6.70	<.0001

Table 24. Quadratic Fit for Initial Heuristic Solution Times

Initial soltime = 0.051 – 0.004 job size + 0.0002 job size^2

Summary of Fit

RSquare	0.717
RSquare Adj	0.693
Root Mean Square Error	0.249
Mean of Response	0.300
Observations (or Sum Wgts)	26

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	2	3.608	1.804	29.1798
Error	23	1.422	0.062	Prob>F
C Total	25	5.030		<.0001

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.051	0.125	0.41	0.6889
job size	-0.004	0.007	-0.54	0.5964
job size^2	0.0002	0.0001	2.31	0.0304

Regression analysis shows that the tabu search solution times are linearly related (for the problems sizes tested). The adjusted r-square value given in the linear analysis of variance (ANOVA) table, .936, indicates a strong linear tendency (see Table 21). The adjusted r-square value of .990 for the quadratic ANOVA table (Table 22), however, indicates an even stronger quadratic tendency. An F-ratio comparison between the linear and the quadratic regression (353.55 vs. 1284.92, respectively) also reveal that the solution times are better modeled by a quadratic equation.

Regression analysis of the initial solution heuristic shows that these solution times also have a slightly stronger quadratic tendency than a linear one (Tables 23 and 24). A comparison of the adjusted r-square values between the linear and the quadratic regression (.637 vs. .693) and the verify this statement.

Overall, the initial solution heuristic and the tabu search both gave rapid, feasible answers to the job shop problems on which they were applied. The initial solution heuristic gives feasible answers in a very short time. The tabu search generally gives solutions with better objective function values, taking somewhat longer than the initial solution heuristic but is substantially faster than optimization. Since the solution times for both methods grow as the number of jobs increase, testing the methods against larger problems is essential.

4.3 Case Study

The case study for this research is the 100 target, 4 unit, 296 mode air campaign planning scenario described in detail in Van Hove's dissertation [Van Hove, Chapter 7, 1998] and described briefly in Appendix C. It was adapted for this research by designing and populating a database to store all of the scenario data. Queries were then written within the database, which were read and used by TSACP when operating. Additionally, all enemy air defense targets were classified as priority one targets, as were several interdiction targets. Other interdiction targets were randomly assigned priority classifications of two or three.

This problem is considerably larger than even the 90-job problems discussed in the previous section. In the 90-job problems, each target (job) could be attacked (executed in) one mode from a selection of from one to three modes, resulting in roughly 530 predecessor/successor mode combinations—the real driving force behind the size of the problem. In Van Hove's [1998] scenario, each target could be attacked by one mode from a selection of either two or four modes, resulting in 999 predecessor/successor mode combinations—almost double the size of the earlier 90-job problems.

The TSACP application actually performs more rapidly on this larger problem because of the use of candidate lists as described in Section 3.4.2. Since TSACP only operates on one priority classification at each iteration, the neighborhoods tend to be smaller than for the 90-job problems where only one target priority classification was defined. Intelligent use of candidate lists will be crucial when the scenarios achieve two to three thousand sorties as was the case during Operation Desert Storm [Cohen, 1993].

The objective function in Van Hove's model was to minimize makespan, while the objective of TSACP is threefold: maximize target coverage, minimize operational impact (for replanning/rescheduling), and minimize makespan. One of Van Hove's assumptions for his model was that adequate aircraft are available to implement the plan, and at the earlier stages of the ATO process that may certainly be the case. Optimization is valid in pre-hostility planning to shape force requirements. However, the battlespace conditions change as hostilities commence and the campaign wears on, including the availability of air-worthy fighter aircraft. By the time the process reaches detailed planning (or even after ATO dissemination) new solutions may be needed. Therefore, TSACP was applied to the scenario as it was originally designed by Van Hove, but also with varied amounts of air assets. The results of applying TSACP to the air campaign planning scenario follow.

The first application of TSACP was against a scenario with sufficient renewable and non-renewable assets as Van Hove planned. The number of aircraft in-theater and the sortie rate were set arbitrarily high, thus relaxing the resource constraints. This was done to obtain a baseline case to see how the solutions would be affected by varying the quantities of each, in turn. Table 20 shows that tightening the non-renewable resource constraint had little impact on the solution, while tightening the renewable constraint made a considerable impact on the solution. Interestingly, TSACP was unable to improve the initial solution for the non-resource constrained problem instance.

The levels of both types of resources were then set at the "threshold" level—the quantities used by Van Hove in his scenario. TSACP was applied to the scenario at this level and then at resource levels of 95%, 90%, 85%, and 80% of the threshold level of the non-renewable resource (total sorties). At the 95% level, and every subsequent level, targets were uncovered (goal deviations began to become non-zero), but in each case, TSACP was able to improve the initial heuristic solution.

Two replanning scenarios were run, one with the attrition of four aircraft from the first unit, and the other with an entire unit unable to participate in the day's battle. Recall that the fourth goal in TSACR is to minimize the impact of any changes in the overall plan. In both cases TSACR was able to reduce the number of missions needing to be replanned vs. the solution obtained by the initial heuristic. All of the results are summarized in Table 25.

Table 25. Results of Applying TSACP to the Air Campaign Planning Scenario

			Times	Goal One	Goal Two	Goal Three	Goal Four	Goal Five
Renewable	Non-renewable	Method	(Seconds)	Deviation	Deviation	Deviation	Deviation	Deviation
infinite	infinite							(makespan)
		Initial Solution	1.70	0	0	0	NA	389
		TS 1000 Iters.	152.05	0	0	0	NA	389
infinite	330							
		Initial Solution	1.64	0	0	0	NA	471
		TS 1000 Iters.	145.62	0	0	0	NA	389
120	infinite							
		Initial Solution	1.69	0	0	0	NA	728
		TS 1000 Iters.	139.73	0	0	0	NA	648
"Threshold"								
120	330							
		Initial Solution	1.77	0	0	0	NA	748
		TS 1000 Iters.	179.6	0	0	0	NA	677
120	314							
		Initial Solution	1.61	0	1	1	NA	635
		TS 1000 Iters.	180.47	0	0	4	NA	568
120	298							
		Initial Solution	1.54	0	4	2	NA	687
		TS 1000 Iters.	197	0	3	0	NA	593
120	282							
		Initial Solution	1.51	0	5	2	NA	670
		TS 1000 Iters.	203.25	0	3	3	NA	552
120	266							
		Initial Solution	1.48	0	7	4	NA	666
		TS 1000 Iters.	211.1	0	5	4	NA	518
Replan	Attrit 4 aircraft from first unit							
116	324							
		Initial Solution	1.98	0	0	1	10	748
One runway		TS 1000 Iters.	184.5	0	0	0	6	736
Lost								
84	250							
		Initial Solution	1.76	0	0	10	50	755
		TS 1000 Iters.	204.52	0	0	10	40	735

The goal of this research has been to present air combat planners with an automated planning capability. Van Hove's optimality-seeking approach is well suited for the pre-hostility phase of an air war. The methods presented in this thesis allow planners to obtain extremely fast, reasonable solutions by using the initial solution heuristic. This kind of solution would be useful when an "instant" answer is necessary (such as when air superiority has not been achieved). Planners also have the ability to use the tabu search to find good solutions in a reasonable time, when more time is available during ATO planning.

CHAPTER 5. Conclusions and Recommendations

This chapter summarizes the research in this thesis. The key points are re-emphasized, the significant contributions are outlined, and recommendations for future research are suggested.

5.1 Research

The research conducted for this thesis was pursued along three primary lines of investigation. First, the nature and scope of the problem were determined by examining how the air campaign planning process worked and what dynamics were involved in producing an Air Tasking Order. Second, the research investigated scheduling theory concepts in order to find parallels to the air campaign planning problem. The third line of investigation delved into meta-heuristics, specifically tabu search, to help develop an application that could be used by air combat planners within a dynamic, time-critical air battle environment. This research, and the resulting application, have commercial value as well—any situation that undergoes rapid change needs fast planning solutions together with options for replanning solutions.

5.2 Contributions

Several important contributions were provided through this research. The first contribution is a fast heuristic approach for the air campaign planning problem incorporating aspects of the air war such as target priority classifications and generalized precedence constraints, as well as specialized aircraft (least flexible machine) and specialized targets (least flexible job). Previous efforts within combat planning tools such as APS allow the user to define a "best" attribute to use within its greedy heuristic approach. The initial solution heuristic developed here considers more aspects of the air battle plan. This heuristic, as well as the tabu search, reads from a database similar to that used in the Aerospace Operations Center so as to more closely replicate conditions in that operational environment. Options allow specific missions to be "frozen" and priority classes of targets to be considered.

While the initial solution heuristic was designed to improve the tabu search by providing a good starting solution, as a stand-alone application it works well. Indeed, the quality of the solutions obtained by the initial solution heuristic were such that, on several of the test problems, the tabu search was unable to improve upon its starting solutions. The initial solution heuristic found the optimal solution on two of the test problems. Moreover, this heuristic works very quickly, obtaining good solutions in a fraction of a second for the smaller problems (up to job size 60) and in less than two seconds for the larger problems, including the case study. Given the dynamics of combat planning, this could be a valued asset.

Another contribution of this research is the construction of a tabu search that operates on a goal programming formulation of the air campaign planning problem. The tabu search yielded good solutions with consideration to target priority classification, and generalized precedence relationships. A list of the best *k* solutions offers options for air combat planners to use during the planning process. The tabu search worked quickly, even on the larger test problems and the case study, therefore proving its potential to operate within the AOC as a planning tool.

The methods developed in this research were applied to the problem of replanning. In many situations, a great deal of time, money, and personnel hours are spent up front generating a comprehensive plan of operation. Should the plan break down due to unforeseen circumstances, planners frequently have no option but to cobble together an *ad hoc* fix, especially when time is a critical factor. These quick fixes to the overall plan may mean extensive replanning by the commanders and managers of the individual jobs and tasks. By using TSACR, planners can generate new solutions with regard to the current situation while attempting to minimize the impact of changes throughout the system.

5.3 Recommendations for Future Work

The research contained within this thesis may be extended in a number of directions. Some of these are:

1. Improve the application by incorporating more advanced tabu search features, such as diversification and intensification phases.
2. Investigate using alternative move definitions within the tabu search.
3. More rigorous testing is necessary. Optimal solutions to the fully constrained test problems should be obtained for comparison against the tabu search solutions. Additionally, TSACP needs to be applied to a fuller range of test problems. Testing against APS and actual ATO's would be highly desirable.
4. Formulate the goal programming model in HyperLingo®, or some other matrix generator/solver for direct comparison with the goal programming solutions given by TSACP.
5. Extend TSACP to include package planning, multi-target tasking, and the ability to land at a base other than the take-off base.

6. Investigate whether tabu search could be used within an optimality-seeking framework, such as Van Hove's hybrid decomposition approach, to speed up the search for an *optimal* solution.
7. Investigate whether the initial solution heuristic could also be used within an optimality-seeking framework.
8. Rigorously test the initial solution heuristic as a stand-alone for very fast, feasible solutions to the air campaign planning problem or similar applications in industry.
9. Extend this research by applying the technique to corporate scheduling and re-scheduling problems.
10. Expand the analysis to include more constraints in the air campaign problem, including available munitions, tankers, theater air control system resources, and crew rest. Such an approach might incorporate large scale techniques.
11. Incorporate the initial solution heuristic and the tabu search within existing applications.

5.4 Summary

A method for finding good solutions, quickly, to the air campaign planning problem was developed during the course of this research. The method can be applied to other military and civilian project planning situations as well. The application was extended to incorporate aspects of replanning as it relates to operational impact.

Appendix A. LIST OF ACRONYMS

ABCCC	Airborne Command, Control, and Communications
ABP	Air Battle Plan
ACDB	Air Campaign Database
ACO	Airspace Control Order
AETACS	Airborne Elements of the Theater Air Control System
AOA	Activity on the Arc
AOC	Aerospace Operations Center
AON	Activity on the Node
APS	Advanced Planning System
AR	Aerial Refueling
ATO	Air Tasking Order
CAFMS	Computer Assisted Force Management System
CAFMS-X	Computer Assisted Force Management System for X-Windows
CAS	Close Air Support
COD	Combat Operations Division
CPD	Combat Plans Division
CPM	Critical Path Method
CTAPS	Contingency Theater Automated Planning System
CTEM	Conventional Targeting Effectiveness Model
CTL	Candidate Target List
DCA	Defensive Counter Air
DO	Duty Officer

DP	Detailed Planning
ECM	Electronic Counter-measures
ESC	Escort
FLOT	Forward Line of Own Troops
GA	Genetic Algorithm
GAT	Guidance, Apportionment, and Targeting
GIP	Goal Interval Programming
GP	Goal Programming
GTACS	Ground Theater Air Control System
IFF	Identification Friend or Foe
INT	Interdiction
IP	Integer Programming
JDBC	Java Database Connectivity
JFACC	Joint Forces Air Component Commander
JFC	Joint Forces Commander
JIPTL	Joint Integrated Prioritized Target List
JPT	JFACC Planning Tool
JTF	Joint Task Force
LFJ	Least Flexible Job
LFM	Least Flexible Machine
LGP	Lexicographic Goal Programming
LNS	Largest Number of Successors
LP	Linear Programming

MAAP	Master Air Attack Plan
MIDB	Military Intelligence Database
MMGRCPSP	Multi-Modal Generalized Precedence Resource Constrained Project Scheduling Problem
MMRCPSP	Multi-Modal Resource Constrained Project Scheduling Problem
NAF	Numbered Air Forces
NET	No Earlier Than
NLP	Non-linear Programming
NLT	No Later Than
OCA	Offensive Counter Air
ODBC	Oracle Database Connectivity
OOP	Object Oriented Programming
PD	Percent Damaged
PK	Probability of Kill
PSP	Project Scheduling Problem
RCPSP	Resource Constrained Project Scheduling Problem
RDB	Relational Database
RECCE	Reconnaissance
SAM	Surface to Air Missile
SCL	Standard Conventional Load
SD	Strategy Division
SEAD	Suppression of Enemy Air Defenses
SQL	Structured Query Language
TBMCS	Theater Battle Management Core Systems

TFT	Time off Target
TNL	Target Nomination List
TOT	Time on Target
TPW	Target Planning Worksheet
TS	Tabu Search
TSACP	Tabu Search for Air Campaign Planning
TSACR	Tabu Search for Air Campaign Replanning
TSACS	Tabu Search for Air Campaign Rescheduling
USMTF	United States Message Text Format
WOC	Wing Operations Center

Appendix B. Relational Databases and SQL

The Air Campaign Database (ACDB) is a relational database (RDB). This is an elegant and efficient way to store and manipulate massive amounts of data. There are a number of commercial RDB packages on the market—three common packages are Oracle, Sybase, and Microsoft Access. The ACDB is an Oracle database and this, together with the Military Intelligence Database (MIDB), a Sybase database, stores nearly all the data used in the Aerospace Operations Center.

A RDB allows users to manage all of the available information from a single database file. Within the file, the data is divided into separate storage containers called tables (often referred to as entities). The database applications mentioned above generally give users the ability to view, add, and update table data using online forms, find and retrieve only the data needed using queries, and analyze or print data in a specific layout using reports.

To store data in an RDB, one table is created for each type of information tracked. Relationships are defined between tables in order to bring data from multiple tables together in a query, form, or report. A query is required to find and retrieve just the data that meets specific conditions, including data from multiple tables. A query can also update or delete multiple records at the same time, and perform built-in or custom calculations on the data [Microsoft Access on-line documentation, 1996].

The Structured Query Language (SQL) is used to access relational databases. This efficient, flexible, fourth-generation language (4GL) contains features designed to manipulate and examine relational data. 4GLs describe what needs to be done, but not how to do it [Urman, p.2, 1997]. SQL is easy to learn and "English-like."

For example, assume a database contained a table called Targets. In this table there are five columns with column headers (or attributes): Target_ID, Target_Type, Target_S_Number, Target_X_Coord, and Target_Y_Coord. If a user were interested in the location of all targets of a specific type, such as hardened shelters, then a simple query to examine just those targets would look like this:

```
SELECT Target_ID,
Target_X_Coord,
Target_Y_Coord

FROM Target

WHERE Target_Type = "Hardened Shelter";
```

If the user wanted all the columns and all the rows the query would simply be:

```
SELECT * FROM Target;
```

One of the key features of using a RDB and SQL is the ability to gather information from multiple tables in a single query. This ability is engendered through the use of the relationships defined between the tables. Even where there is no specific relationship, SQL is powerful enough to create relationships between tables "on the fly."

Continuing with the example, suppose a user wished to find out the distance and estimated flying time between a unit and a target. Assume there was a table called "Unit" containing the following attributes: Unit_ID, Unit_Name, Unit_Base_Name, Unit_Aircraft_Type, Unit_Aircraft_Qy, Unit_Utilization_Rate, and Unit_Sortie_Rate. Additionally, there exists a table called "Base" with the attributes Base_ID, Base_Name, Base_X_Coord, and Base_Y_Coord and a table named "Aircraft" with the attributes Aircraft_ID, Aircraft_Name, Aircraft_Nominal_Speed, and Aircraft_Turn_Time. A query returning the Base_Name, the Unit_Name, the Target_ID, the distance from each base to each target, and the estimated straight-line travel time for each unit from base to target might look like the following:

```
SELECT DISTINCT
Base.Base_Name,
Unit.Unit_Name,
Target.Target_ID,
Sqr(POWER((Target_X_Coord - Base_X_Coord),2) +
POWER((Target_Y_Coord - Base_Y_Coord),2)) AS Distance,
Distance/Aircraft_Nominal_Speed AS Time_to_Target
FROM Target, Aircraft, Unit, Base
WHERE Aircraft.Aircraft_Name = Unit.Unit_Aircraft_Type
AND Unit.Unit_Base_Name = Base.Base_Name;
ORDER BY Base.Base_Name,
Unit.Unit_Name,
Time_to_Target
```

The preceding query employs SQL built-in functions to perform calculations within the query. The standard arithmetic operators such as + (addition), - (subtraction), * (multiplication), and / (division) are incorporated into SQL as well as various other functions. The expression, "Sqr(POWER((Target_X_Coord - Base_X_Coord),2) + POWER((Target_Y_Coord - Base_Y_Coord),2)) AS Distance" is the simple Euclidean distance function employed in elementary algebra (i.e. $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$). The SQL function Sqr(*arg*) finds the square root of *arg,* the function POWER(x,y) calculates the y^th^ power of x. The distance is divided by the nominal speed of the aircraft to determine the flying time to the target in the expression, "Distance/Aircraft_Nominal_Speed AS Time_to_Target." SQL performs the calculations in the expressions and the columns are given the aliases, "Distance" and "Time_to_Target," by the *AS* phrases. Any distance formula and duration calculation that was appropriate could be used.

A *JOIN* combines records from two tables whenever there are matching values in a common field. The WHERE clause in this example specifies that the Aircraft table and the Unit table be joined whenever the Aircraft_Type_Name from the Unit table matches the Aircraft_Name from the Aircraft table. It also joins the Unit and Base tables whenever the Unit_Base_Name from the Unit table matches the Base_Name from the Base table.

If tables in a query were not joined, either directly or indirectly, then SQL would not know which records are supposed to be associated with each other. Should this occur, SQL will return every possible combination; this is called the Cartesian product (or cross-product) between the two tables. For example, if one table had 200 records (or rows) and the other had 100, then a query with no join would return 20,000 (200 x 100) records. The problem is compounded for each additional table in the query. Such a query would take longer to run and the information might be useless [Microsoft Access on-line documentation, 1996].

The ORDER BY clause sorts the information retrieved. In the example, the records are put in alphabetical order first by Base_Name, then by Unit_Name. Then the records are put in order of increasing time to target. If the user wanted the records sorted in order of decreasing time to target, then "DESC" would be appended to the ORDER BY clause:

```
ORDER BY Base.Base_Name,
Unit.Unit_Name,
Time_to_Target DESC.
```

The RDB can store saved queries as *views* (pseudo-tables) in the database file. These views already have the user-specified calculations, relationships, filters, and sorting preferences, thus it could be said that the data is pre-processed. Making SQL calls to these views from within an application rather than making complex calls to the original tables shortens the time required for the application to perform an iteration.

Appendix C. Description of the Case Study Problem

Three components are necessary to formulate the combat planning scenario: activities, resources, and precedence relations. Van Hove's scenario consists of 40 target locations, half of them enemy air defense targets and the other half interdiction targets. Each target location includes two or three separate targets that must be attacked by separate missions. There were 100 total targets in his scenario with a corresponding total of 100 activities in the problem formulation. Either 2 or 4 aircraft must be planned for each mission, resulting in an ATO of 200-400 sorties.

The resources used in the scenario consisted of four units located at three different bases: one at base A, two at base B, and one at Base C. Each unit possesses one of two aircraft types, AC-1 and AC-2. One AC-1 unit is located at base A, the other at base B. One AC-2 unit is located at base B, the other at base C. Each unit owns a different quantity of resources. Modes and resource requirements for targets were assigned randomly; every activity (target) has either two or four execution modes.

Base and target locations, along with the aircraft's nominal airspeed dictate leg durations. Each aircraft type also has a different turn time. The average round trip distance between the bases and targets is approximately 120 distance units. A nominal airspeed of 1.0 distance units per minute makes the average flight time for a mission equal to 2 hours. This, taken with an average turn time of 1 hour, means that the average mission ties up assets for approximately 3 hours. AC-2 is given a faster nominal airspeed in exchange for a smaller turn rate and a longer turn time. Generalized precedence constraints were formulated to enforce mission timing.

Bibliography

Air Force Instruction 13-1 AOC Volume 3: Operational Procedures—Aerospace Operations Center, HQ USAF/XOCE 1 June 1999.

Air Force Manual 1-1: Basic Aerospace Doctrine of the United States Air Force, II, U.S. Government Printing Office, March 1992.

Battlestaff Training School, Joint Air Operations Staff Course (JAOSC), Course Handout, USAF Air Ground Operations School, Hurlbert Field, FL, Jan 98.

Joint Pub 3-56.1: Command and Control for Joint Air Operations, Washington: Department of Defense, November 1994.

Microsoft Access 97 On-line Help, Microsoft Corporation, (1996).

Scheduling Theory: Course notes, Deckro, Richard, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH Summer, 1999.

Theater Battle Management Core Systems (TBMCS) Joint Program Management Review (JPMR), Working Group Handout, Lockheed Martin Command and Control Systems, Colorado Springs, CO, November, 1997.

Abt, Paul E., "Air Campaign Planning," *Theater Battle Management Core Systems Newsletter*, Vol. 3 No.1 15-17 January 1997.

Ahuja, R., Magnanti, T., Orlin, J. *Network Flows: Theory, Algorithms, and Applications,* New Jersey: Prentice Hall (1993).

Barbarosoglu, G. and D. Ozgur "A Tabu Search Algorithm for the Vehicle Routing Problem," *Computers and Operations Research*, Vol. 26: 255-270 (1999).

Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C., Stewart, W.R.Jr., "Designing and Reporting on Computational Experiments with Heuristic Methods" *Journal of Heuristics*, Vol. 1 No. 1 p. 9-32, (Fall 1995).

Barnes, J.W. and W.B. Carlton "Solving the Vehicle Routing Problem with Time Windows Using Reactive Tabu Search," presented at the Fall INFORMS National Meeting in New Orleans, Louisiana (1995).

Battiti, Roberto, "Reactive Search: Toward Self-tuning Heuristics," *Modern Heuristic Search Methods*. Ed. Rayward-Smith V. et al., Chichester, UK: John Wiley and Sons, (1996).

Baykasoglu, A., "Solution of Goal Programming Models Using a Basic Taboo Search Algorithm," *Journal of the Operational Research Society,* Vol. 50 No. 9 pp. 960-973, (Sep 1999).

Boctor, Fayez F. "A New and Efficient Heuristic for Scheduling Projects with Resource Restrictions and Multiple Execution Modes," *European Journal of Operational Research, 90*(2):349-361 (April 1996).

Carlton, W. and Barnes J., "A Note on Hashing Functions and Tabu Search Algorithms," European Journal of Operational Research, Vol. 95:237-239 (1996).

Carruthers, J.A. and A. Battersby. "Advances in Critical Path Methods," *Operations Research Quarterly*, 17:359-380 (1966).

Charnes, A., W. Cooper, R. Ferguson, "Optimal estimation of Executive Compensation by Linear Programming," *Management Science,* Vol 1:138-151 (1955).

Charnes, A., Cooper, W., Harrald, J., Karwan, K., Wallace, W. "A Goal Interval Programming Model for Resource Allocation in a Marine Environmental Protection Program," *Journal of Environmental Economics and Management,* 3:347-362 (1976).

Cohen, Eliot A., *Gulf War Air Power Survey: Planning and Command and Control*, Technical Report AD-A279741, Department of the Air Force (1993).

Colletti, Bruce W., *Group Theory and Metaheuristics*, Dissertation, University of Texas, Austin, TX (May 1999).

Cotsworth, W.L., *The Conventional Targeting Effectiveness Model (CTEM) User's Manual,* AEM Services, Inc., S-93-006-DEN, (31 December 1993).

Deckro, Richard and John Hebert, "Polynomial Goal Programming: A Procedure for Modeling Preference Trade-Offs," *Journal of Operations Management,* Vol. 7, No. 4:149-164 (December 1988).

Deitel, H., and P. Deitel, *Java: How to Program,* Prentice Hall, (1998).

Dell'Amico, M. and Trubian, M. "Applying Tabu Search to the Job-shop Scheduling Problem" *Annals of Operations Research*, Vol. 41: 231-252 (1993).

Desrochers. M., J. Desrosiers, and M. Solomon "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows," *Operations Research,* Volume 40: 342-354 (1992).

Gandibleux, X.; Mezdaoui, N.; Freville, A. "A Tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problem" *Lecture notes in economics and mathematical systems*. Vol. 455: (1997).

Garcia, B.L., J.Y. Potvin, and J.M. Rousseau "A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints," *Computers and Operations Research*, Vol. 21: 1025-1033 (1994).

Glover, Fred. "Tabu Search: A Tutorial" *Interfaces,* Vol. 20:4 pp. 74-94, July-August, 1990.

Glover, Fred and Manuel Laguna. "Chapter 3: Tabu Search," *Modern Heuristic Techniques for Combinatorial Problems*, ed. Colin R. Reeves, John Wiley & Sons, Inc, (1993).

Glover, Fred and Laguna, Manuel *Tabu Search,* Boston: Kluwer Academic Publishers, (1997).

Gonzales, Daniel R. *Evolution of the Air Campaign Planning Process and the Contingency Theater Automated Planning System (CTAPS).* Technical Report MR-

618-AF, 1700 main St., P.O. box 2138, Santa Monica, CA 90407-2138: RAND, 1996.

Haas, William R., *An Operational Review of Air Campaign Planning Automation*, Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, March 1998.

Jensen, Paul A., "Integer Programming." *Integer Programming course handout.* School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, July 1994.

Jozefowska, J., Waligora, G. and Weglarz, J. "A Tabu Search Algorithm for Some Discrete-Continuous Scheduling Problems" *Modern Heuristic Search Methods*, Ed. Rayward-Smith, V., Osman, I., Reeves, C. and Smith, G. Chichester, UK: John Wiley and Sons, (1996).

Kelley, James E. "Critical-Path Planning and Scheduling: Mathematical Basis," *Operations Research*, 9(3):296-320 (May-June 1961).

Koewler, David A., An *Approach for Tasking Allocated Combat Resources to Targets*, Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, (March 1999).

Kolisch, Rainer, Sprecher, and Drexl, *Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems.* Manuskripte aus den Instituten fur Betriebswirtschaftslehere, No. 301, University of Kiel, Germany (December, 1992).

Kolisch, Rainer, Sprecher, and Drexl, "Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems," *Management Science,* 41(11):1693-1703 (1995).

Laguna, M., J. Barnes, and F. Glover, "Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Setup Costs Using Tabu Search," research report, Department of Mechanical Engineering, University of Texas-Austin (April 1989).

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

Mooney, E. and Rardin, R. "Tabu Search for a Class of Scheduling Problems" *Annals of Operations Research*, Vol. 41: 253-278 (1993).

Nowicki, E, and C. Smutnicki, "A Fast Taboo Search Algorithm for the Job Shop Problem" Management Science, Vol. 42, No. 6: 797-813 (1996).

Pinedo, Michael *Scheduling Theory, Algorithms, and Systems,* New Jersey: Prentice Hall, (1995).

Pritsker, A., L. Watters, and P. Wolfe, "Multi-Project Scheduling with Limited Resources: a Zero-One Programming Approach," *Management Science, 16*(1):93-108 (September 1969).

Rangaswamy, B., Jain, A. and Glover, F. "Tabu Search Candidate List Strategies in Scheduling" *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search.* Ed. Woodruff, D., Boston: Kluwer Academic Publishers (1998).

Romero, Carlos, *Handbook of Critical Issues in Goal Programming,* Pergamon Press, (1991).

Shtub, A., J. Bard, S. Globerson, *Project Management Engineering, Technology, and Implementation,* New Jersey: Prentice Hall, (1994).

Slowinski, Roman et al., "DSS for Multiobjective Project Scheduling," *European Journal of Operational Research*, 79:220-229 (1994).

Urman, Scott, *Oracle8 PL/SQL Programming,* McGraw-Hill, Berkeley, CA, 1997.

Van Hove, John C., *Personal E-mail,* Air Force Logistics Management Agency, USAF, Maxwell AFB, Gunter Annex, (16 March 2000).

Van Hove, John C., *An Integer Program Decomposition Approach to Combat Planning,* Dissertation, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, (October 1998).

Winston, Wayne L. *Operations Research: Applications and Algorithms (3rd edition).* Belmont, CA: Wadsworth Publishing Company, (1994).

Yost, Kirk A., "Consolidating the USAF's Conventional Munitions Models," *Military Operations Research*, Vol. 2, Number 4:53-71 (1996).