

AD-E 300 650  
**(12) LEVEL III**

DNA 5018F-1

ADA 079944

**MOST FINAL REPORT**  
Volume I

L. J. Nessler  
Science Applications, Inc.  
P.O. Box 2351  
La Jolla, California 92038

30 April 1978

Final Report for Period 5 July 1977-30 April 1978

CONTRACT No. DNA 001-77-C-0275

DDC FILE COPY

APPROVED FOR PUBLIC RELEASE;  
DISTRIBUTION UNLIMITED.

THIS WORK SPONSORED BY THE DEFENSE NUCLEAR AGENCY  
UNDER RDT&E RMSS CODE B364077464 V99QAXNL12242 H2590D.

Prepared for  
Director  
DEFENSE NUCLEAR AGENCY  
Washington, D. C. 20305

DDC  
RECEIVED  
JAN 29 1980  
B

Destroy this report when it is no longer  
needed. Do not return to sender.

PLEASE NOTIFY THE DEFENSE NUCLEAR AGENCY,  
ATTN: STTI, WASHINGTON, D.C. 20305, IF  
YOUR ADDRESS IS INCORRECT, IF YOU WISH TO  
BE DELETED FROM THE DISTRIBUTION LIST, OR  
IF THE ADDRESSEE IS NO LONGER EMPLOYED BY  
YOUR ORGANIZATION.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DNA 5018F-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MOST FINAL REPORT Volume I	5. TYPE OF REPORT & PERIOD COVERED Final Report for Period 5 July 1977 - 30 April 1978	6. PERFORMING ORG. REPORT NUMBER SAI-78-631-LJ
7. AUTHOR(s) L. J. Nessler	8. CONTRACT OR GRANT NUMBER(s) DNA 001-77-C-0275	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Science Applications, Inc. P.O. Box 2351 La Jolla, California 92038	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Subtask V99QAXNL122-42	
11. CONTROLLING OFFICE NAME AND ADDRESS Director Defense Nuclear Agency Washington, D.C. 20305	12. REPORT DATE 30 April 1978	13. NUMBER OF PAGES 34
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) DINIS, SPIT	15. SECURITY CLASS (of this report) UNCLASSIFIED	16. SECURITY CLASSIFICATION DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 26, if different from Report) Approved for public release; distribution unlimited.		
18. SUPPLEMENTARY NOTES This work sponsored by the Defense Nuclear Agency under RDT&E RMSS Code B364077464 V99QAXNL12242 H2590D.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Aimpoint Selection      Targeting DGZ                              Nuclear Poseidon                      Elementized Targets LAIR		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The MOST code (for Multiweapon Optimizer for Strategic Targets) is designed to find aimpoints for nuclear weapons assigned to complex targets such as airfields. It finds the minimum number of weapons needed to accomplish the objective of an attack and optimizes aimpoints for them. Although designed primarily for Poseidon RV's in its current form it will handle any problem in which identical yields are used.		

388 800

34

## 20. ABSTRACT (Continued)

There are two kinds of attack objectives that may be specified. The first attack objective is the achievement of at least a minimum probability of damage against each of the elements of the complex. The second objective is to achieve a required level of damage to the complex as a whole. It is assumed that each element of the complex is assigned a value against which other elements can be weighed, and that the value-weighted probability of damage to an element is the probability of damage to it multiplied by its value. This second objective, then, is to achieve at least a minimum required value-weighted average probability of damage on the entire complex (which is the sum of the value-weighted probabilities of damage of all of the elements divided by the total value of the complex). Compounded probabilities of damage, due to multiple bursts, are taken into account in the achievement of this second objective, and the value-weighted average PD is maximized during aimpoint optimization.

Colocated targets (not part of the complex) have been taken into account, and may be given a high probability of damage requirement independent of the requirements of the complex.

MOST is currently in the form of a FORTRAN-callable subroutine package implemented for the IBM-370 and DEC-10 systems. It was developed by Science Applications, Inc. under the sponsorship and direction of the Defense Nuclear Agency.

ACCESSION for		
NTIS	White Section	<input checked="" type="checkbox"/>
DGC	Exit Section	<input type="checkbox"/>
UNCLASSIFIED		<input type="checkbox"/>
U.S. SECTION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	Avail.	and/or SPECIAL
A		

## PREFACE

This report describes the project entitled "Multi-weapon Aimpoint Optimizer for Elementized Targets," conducted for the Joint Strategic Target Planning Staff (JSTPS) by Science Applications, Inc. (SAI), under the sponsorship and direction of the Defense Nuclear Agency (DNA). The result of this work is the MOST computer code (for Multiweapon Optimizer for Strategic Targets), delivered to JSTPS in December of 1977.

DNA has sponsored research into aimpoint selection at SAI since 1974, research that has led to such standard tools as DCAPS and IDES. Some of this research is outlined here, but much is omitted because MOST, in its present form, is strongly tailored to one or two specific applications.

It should, however, be emphasized that the methodology is flexible enough to support numerous useful extensions of the implementation herein described.

The MOST project owes its success to the strong participation of several key individuals. LTC Richard Walker (USAF), of JSTPS, created the opportunity to construct MOST by seeing the importance of the herein-described methodology to operational targeting problems. He also had numerous sound suggestions concerning the algorithms employed, and made sure that the results would be useful. LT Paul Foster (USAF) and LT Richard Magnan (USAF), also of JSTPS, ably performed the various IBM conversion tasks and the adaptation of MOST to existing software.

Finally, LTC Richard Edwards (USAF) of the Defense Nuclear Agency, who was the Contracting Officer's Representative for MOST, skilfully managed the three-organization project, setting priorities, allocating resources, and removing obstacles.

## TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
PREFACE . . . . .	1
1. INTRODUCTION . . . . .	3
2. METHODOLOGY . . . . .	11
2.1 PRELIMINARY . . . . .	11
2.2 DEFINITION OF TERMS AND CONCEPTS . . . . .	13
2.3 AIMPOINT ENUMERATION . . . . .	14
2.4 MUTUALLY EXCLUSIVE SETS . . . . .	16
2.5 REDUCTION TO EQUIVALENT, MINIMUM-RANK SOLUTIONS	17
2.6 TIE-BREAKING AND THE REFERENCE AIMPOINT . . . . .	20
2.7 OPTIMIZING AIMPOINTS . . . . .	21
2.8 MEETING THE APD REQUIREMENT . . . . .	25
2.9 VARIABLE PD REQUIREMENTS . . . . .	26
3. CONCLUSIONS . . . . .	29

## 1. INTRODUCTION

The problems associated with planning multiple-weapon attacks against a relatively dense complex of targets are often underestimated in both difficulty and significance. This introduction presents a few of these problems with the purpose of explaining some of the more important design features of the MOST code.

MOST (for Multiweapon Optimizer for Strategic Targets) finds the fewest number of weapons (and associated aimpoints) required to achieve acceptable probabilities of damage on a complex of targets. A "complex" is a set of targets related by function and sharing a common value system. Refineries, airfields, rail yards, and steel mills are examples of targets that are best represented as a complex of separate but related elements. Each element is usually assigned a numeric value to weigh its importance to the others in the complex, and the sum of these values is the total value of the complex.

MOST takes into account two separate criteria for finding aimpoints for a complex. The first is a simple minimum probability of damage (PD) requirement on each of the elements. The second is the assurance that at least a certain percentage of the value of the complex will be destroyed. The quantity involved is called the value-weighted (or point-weighted) average probability of damage (APD), and is calculated by compounding the probabilities of damage against each element from all weapons, producing the expected value destroyed (which is, for any element, the compounded PD times the element's value), and dividing by the complex value. Both criteria (PD and APD) are met by MOST when it optimizes aimpoints within a complex.

Occasionally an important target that is not part of a complex is located in proximity to a complex ("colocated with the complex"). MOST will handle such targets by meeting the simple PD requirement specified for them, but will tend to keep the actual PD close to this requirement in order to maximize the point-weighted average PD on the complex.

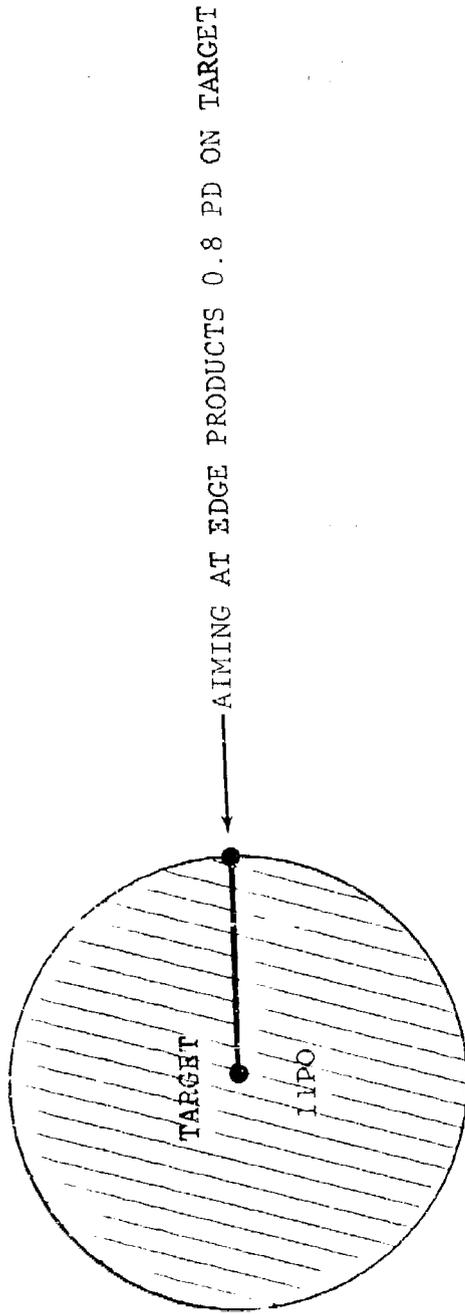
A simple PD requirements is essential because it defines what constitutes an acceptable distance from a target that an aimpoint may be offset. Targets are represented as in the TDI, and damage is calculated in routines modeled after but not identical to those of PDCALC (see JSTPS document JSTPS-TR-76-2 or later release). Because targets are either point or normally distributed, the offset distance derived from the damage routines can be rotated to form a circle in which any aimpoint (notice that an aimpoint is not the same as a point of impact) will achieve the required PD for some specific weapon. Aiming at the edge of this circle will produce exactly the required PD, aiming closer to the target center will produce a higher PD. These circles have the nick-name of "LAIR", for "Lethal Aimpoint Region", and are shown in Fig. 1.

The crucial point of a LAIR is that all aimpoints within it are equally acceptable (but not necessarily equally desirable). Therefore, the offset may be used to minimize the required number of weapons (some programs, such as DCAPS, use it to minimize collateral damage as well). As the pair of circles on Fig. 1 shows, any aimpoint in the intersection of two or more LAIRs will achieve at least the required PD's on all the targets involved in the intersection.

MOST, in its current form, does not handle targets too hard for a specific weapon to achieve the PD requirements.

PD REQUIRED: 0.8

WEAPON: 100 KT, 500' CEP, 500' HOE



AIMING IN INTERSECTION OF LAIRS DESTROYS BOTH TARGETS

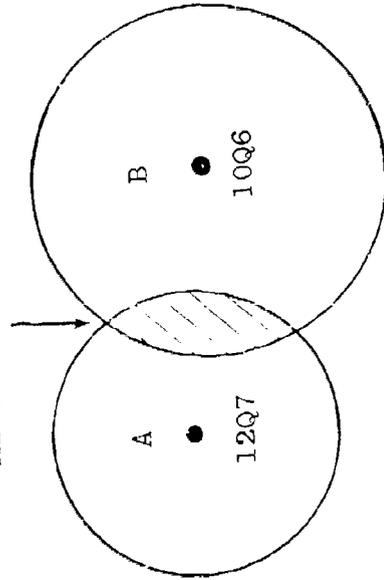


Figure 1. LAIRs.

In general terms, the procedure used by MOST to find the minimum number of aimpoints that meet the simple PD requirements (APD is handled in another step) is to generate all of the appropriate LAIRs, tabulate the regions of intersection, and find the smallest set of intersections that include all of the targets. Then, within the intersections of this set, the actual aimpoints are located which, when compounded together, maximize the value-weighted average PD of the complex. If the calculated APD meets the requirement, then the code is finished. Otherwise, additional weapons are added at aimpoints which make the largest possible increase in APD until the requirement is met.

Of course, the actual procedure is far more complex, and will be described later. However, the above description should provide enough of a description of the method to point out a few of the problems it is designed to solve.

In a target complex, it is frequently true that the best aimpoint for an attack of one weapon is not a good aimpoint for an attack of several weapons. Figure 2 shows a complex of 12 identical targets to be attacked by a weapon just large enough to achieve the required PD on the four central targets if aimed where shown between them (2a). Part 2b shows what happens if this "best" aimpoint is the first found -- namely, that the complex ends up requiring five weapons to achieve the simple-PD requirements. The third drawing (2c) shows the actual four-weapon minimum solution.

Many aimpoint optimizers fall into the trap of assuming that the best N sequentially-selected aimpoints constitutes the best aimpoints for an N-weapon attack. In other words, suppose the best single-weapon aimpoint is found first. Next, the second best aimpoint, when compounded with the first, is found, followed by the third, and so on until all requirements are met. In addition to the possible failure of algorithms of this variety to find the minimum number of required weapons, they may also exhibit unstable behavior when a large

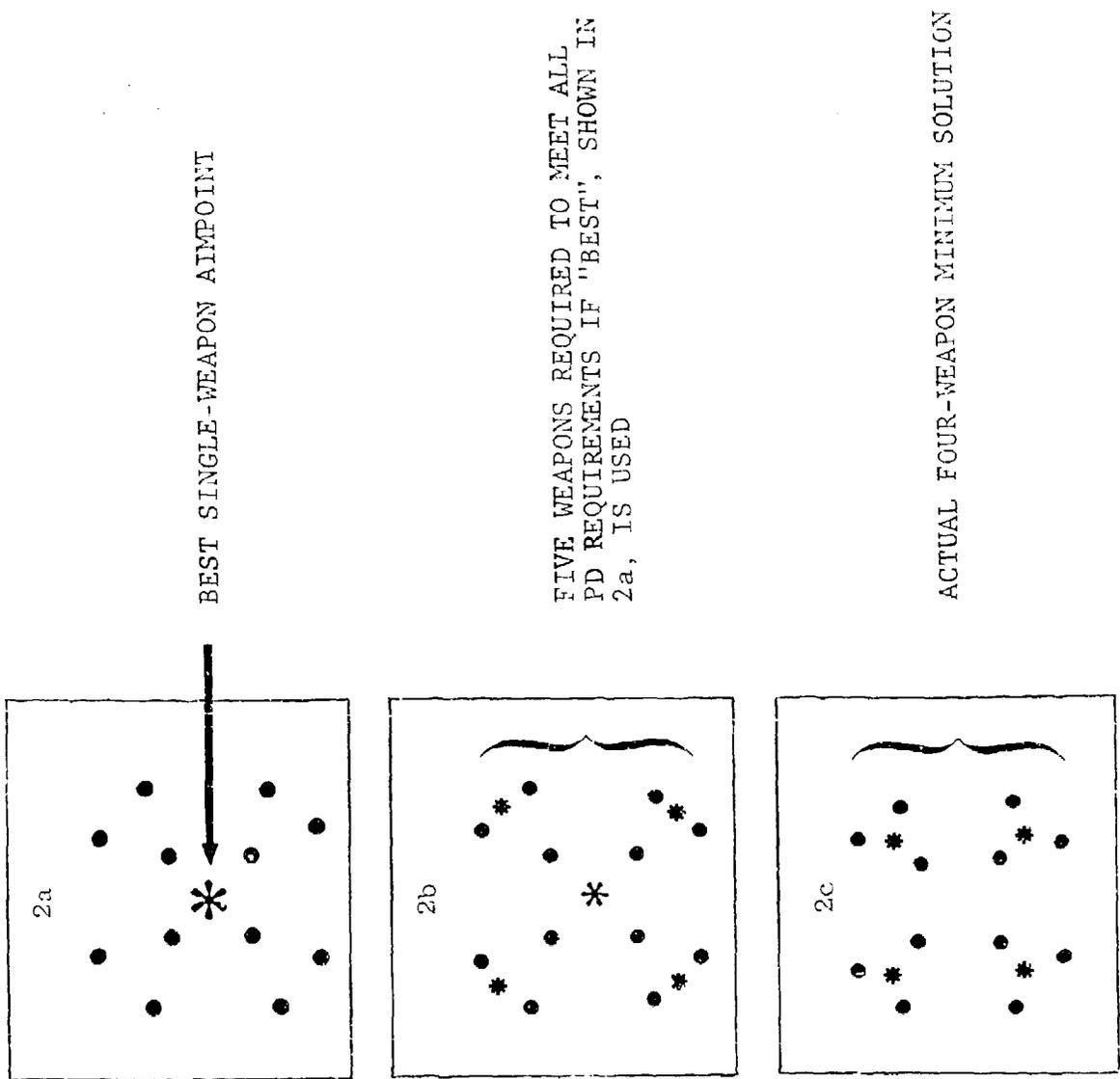


Figure 2. Effect of a bad first choice.

number of weapons are required to meet the requirements. This latter behavior is produced because any aimpoint found cannot be more effective than any found previously (that is, if  $E(N)$  is the effectiveness of the  $N$ th aimpoint to be found,  $E(N) \geq E(N+1)$  for all  $N$ ). Thus, as the algorithm approaches its solution, the aimpoints can contribute so little to the solution that the rate at which aimpoints are added becomes exponentially large. Figure 3 shows a surviving targeting space after a certain number of aimpoints have been found. It should be obvious from the shape of the area destroyed that additional weapons can work only on the ever smaller areas between the circles and, therefore, must be much less effective. The MOST algorithm, which finds all ways of employing  $N$  weapons, is not subject to these pitfalls.

Because MOST, in its present form, does not mix weapon types, it is particularly appropriate for selecting aimpoints for MIRVed weapons. Because the simple PD requirement ignores compounding, any or all weapons that arrive are likely to achieve militarily-acceptable target damage for this portion of the total attack. In addition, a methodology for using MOST concepts in targeting with weapons of different yields is known, although this has not been implemented in MOST.

A final observation of importance to this introduction to MOST is to point out that the code has been optimized for complexes containing about 12 targets, and can handle complexes as large as 20 (or more, under certain conditions). It cannot handle extremely dense and numerous targets in its present form.

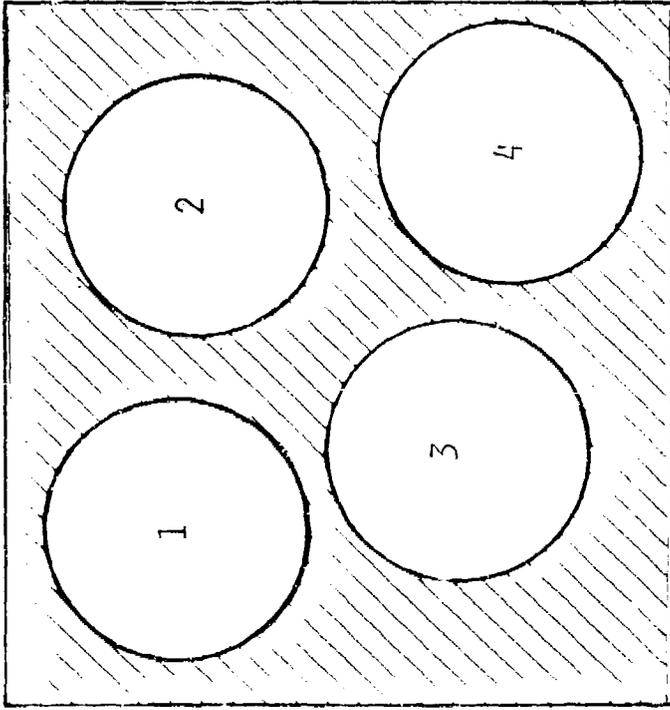


Figure 3. Targeting space after four weapons.

## 2. METHODOLOGY

The algorithms in MOST will be described in this section. They will be presented in the order in which they are applied.

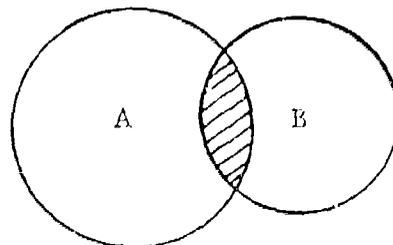
### 2.1 PRELIMINARY

MOST is given a set of installations (or targets), a weapon, and various parameters set up in driver routines. The specific details of how these drivers operate are dependent upon each user's requirements and are not part of MOST per se. A description of one driver (built for testing purposes) is given in the "User's Guide for MOST".

### 2.2 DEFINITION OF TERMS AND CONCEPTS

Throughout this report targets are denoted by a single capital letter. The LAIR of a target is the area around the target in which a specific weapon may be aimed and achieve at least a specified probability of damage higher on the target. The LAIR of a target and the target itself are denoted by the same letter. Although this terminology is ambiguous, the context of usage eliminates confusion.

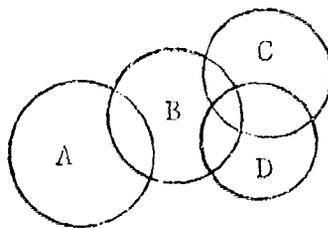
The intersection of the LAIRS of two or more targets is denoted by the concatenation of the letters denoting the respective LAIRS. In the simple figure below, the intersection of A and B is shaded.



The LAIR of B is smaller than that of A for any of several reasons (higher PD requirement, bigger VN number, bigger R-95). The intersection of A and B, or AB, is important because any aimpoint in AB meets the PD requirements on both A and B. Although this is a trivial example, it can be used to illustrate an important point. Suppose the radius of A and the radius of B are known, as well as the distance between the target centers. It can easily be proved that the intersection AB is nonempty if, and only if, the sum of the radii is greater than or equal to the distance between the target centers. Therefore, with simple calculations, MOST can know that a region exists in which any aimpoint will destroy both A and B. MOST at this stage, does not need to know any particular or specific aimpoint. So it should be easily seen that MOST can calculate the minimum number of aimpoints required to achieve the PD requirements on a group of targets without having to derive any particular aimpoints.

In fact, MOST works by treating the target LAIRs and the intersections of LAIRs as if they were "logical aimpoints", and finds actual, or "physical" aimpoints at a later stage (after the minimum number has been determined).

A set of logical aimpoints that includes every target in the group is called a "weapon set" with a rank equal to the number of logical aimpoints in the set. The figure below shows a four-target complex:



The following nine logical aimpoints can be found for this complex:

AB	BCD	CD	D
A	BC	C	
	BD		
	B		

The convention is followed that the names of the aimpoints are always given in alphabetical order.

The following six weapon sets are all that exist in the above complex with the restriction that no LAIR appears more than once:

<u>SET</u>				<u>RANK</u>
A	B	C	D	4
A	B	CD		3
A	BCD			2
A	BC		D	3
AB	C		D	3
AB	CD			2

Notice that there are two minimum rank weapon sets. These sets are equally acceptable with respect to the simple PD requirements. The tie will be broken by evaluating the sets with respect to the second criteria, value-weighted average PD.

The restriction that no LAIR appear more than once is not a limiting factor. "Redundant" sets, such as AB BCD and AB BC CD, vastly increase the number of possible sets without adding any additional capability. First, it can be proved that an enumeration that includes redundant sets cannot lead to the finding of a smaller set than an enumeration in which the redundancies are omitted. Although no rigorous proof will be presented here, it can easily be seen that every redundant set can be derived from a non-redundant set of the same rank. And AB is entirely subsumed in A. Therefore, when physical aimpoints are

formed, the aimpoint finder or "optimizer" can range over the LAIR of A, finding an aimpoint actually in AB if beneficial.

### 2.3 AIMPOINT ENUMERATION

MOST calculates the LAIRs of all the targets in the group given to it. It then generates a list of all logical aimpoints reasonable to consider.

If there are N targets, the list begins with N entries, one for each target. Each entry consists of the name of the intersection or logical aimpoint, its rank, and a physical aimpoint called the "reference" aimpoint, to be described later. Next, the logical aimpoints of rank two are added. To do this the distance between two simple targets is compared with the sum of the LAIRs. If the distance is greater, there is no intersection.

Each pair is then examined to determine the points of intersection of the two circles representing the LAIR boundaries. (For tangent LAIRs there are two identical points. For LAIRs in which one is entirely subsumed in another, the coordinates of the subsumed LAIR are used.) To determine the list of intersection of rank larger than two, the specified weapon is aimed at each of the two points. All of the installations that receive a high enough PD from one of these are collected, and if there are more than the initial pair, the existence of a logical aimpoint of rank equal to the number of targets is deduced. If it is not already in the list, it is added and the other point examined.

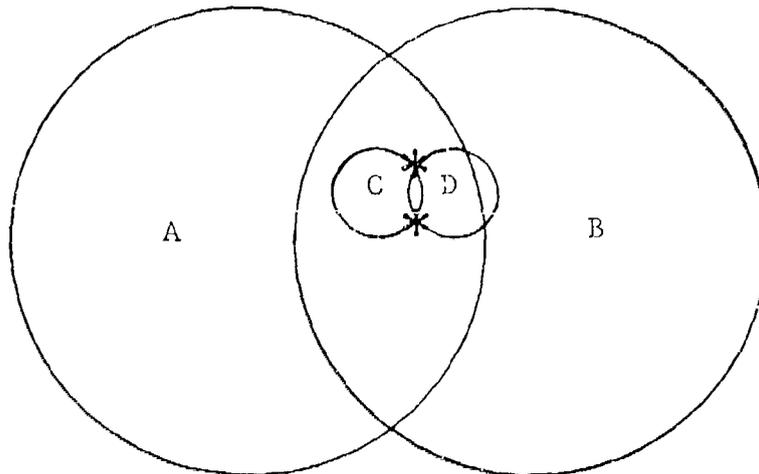
\* The term "rank" has been used to describe the number of weapons in a weapon set. The term can also be applied to an aimpoint when the rank is equal to the number of targets in the aimpoint.

All relevant aimpoints are generated in this fashion. If there are  $N$  targets and  $C_2^N$  possible logical aimpoints of rank two (where  $C_2^N$  is the combination of  $N$  things 2 at a time), then the maximum number of logical aimpoints that can exist for the entire group is given by the expression

$$\text{TOTAL} = N + C_2^N + 2C_2^N = N + 3C_2^N$$

where TOTAL is the number of installations plus the number of pairs plus the maximum number of logical aimpoints that can be generated from these pairs.

Notice that the algorithm here described produces a partial enumeration of the logical aimpoints rather than a complete enumeration. It is difficult to draw a simple example of the kind of logical aimpoint missed by this enumeration. One example is shown in the figure below:



The list of singles and pairs is as follows:

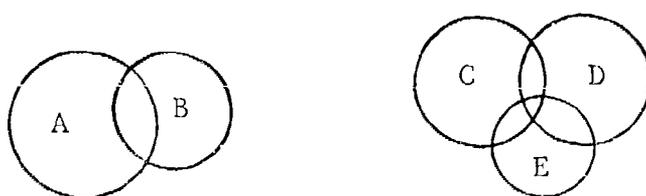
AB	BC	CD	D
AC	BD	C	
AD	B		
A			

When the two starred points are examined, they each show the existence of area ABCD. No point of intersection will generate areas ACD or CDB, or show that they are not the same. In fact, ACD, ACB, ADB and CBD (all the three-circle intersections) will fail to be enumerated.

The fact that the enumeration is partial has no bearing on the minimum weapon solution directly. It misses only those regions that are included in regions of higher rank, and the region of higher rank is always chosen in these peculiar cases. Therefore, the enumeration is not a limiting factor, and the number of entries in the list can be significantly reduced (which is why MOST uses a partial enumeration).

#### 2.4 MUTUALLY EXCLUSIVE SETS

The figure below shows five targets in two isolated groups.



It should be clear that finding the minimum number of aimpoints for one group is independent of the process of finding the minimum number for the other. Even if alternatives of the same rank

existed for one group, the solution would not depend on the results of the other group (this latter observation will be supported when the process of deciding between alternatives of minimum rank is discussed). Therefore, MOST separates all groups of disconnected targets and processes each separately, combining the results when the resolution of alternatives takes place.

The technique used starts with the first aimpoint in the list (which must be of rank one), assigning it to sublist one. All the other aimpoints are examined. If any contain the first target (as ABC contains A), the aimpoint found is also added to the sublist. When all aimpoints containing the first target have been thusly isolated, a list is made of all of the targets involved in the sublist. If the sublist contains aimpoints A, AB, AD, AEF, then the list contains A, B, D, E, and F. The main list is then checked for occurrences of B, D, E, and F, assigning all aimpoints containing one of these to the sublist. As new aimpoints are added, the list of involved targets may grow larger. The process terminates when no target included in the sublist is included in any aimpoint of the parent or main list.

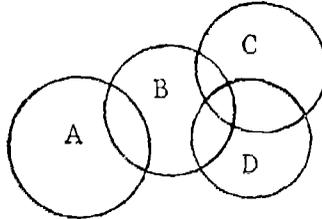
The process then repeats for the next aimpoint (of rank one) if any, remaining in the main list, until all aimpoints have been assigned to a group.

## 2.5 REDUCTION TO EQUIVALENT, MINIMUM-RANK SOLUTIONS

Once the logical aimpoints have been defined and placed into mutually-exclusive sets, each set may be examined separately with the object of finding all weapon sets of minimum rank. Exhaustive enumeration of the combinations of logical aimpoints could produce the required result in the absence of practical considerations. If there are  $N$  targets, the number of possible combinations is approximately  $C_1^N + C_2^N + \dots + C_{N-1}^N + 1$ . For 20 targets the number would be astronomical.

The enumeration problem is essentially geometric in its growth rate, and no algorithm is known that reduces this rate to arithmetic proportions. MOST, however, has dampened this growth rate considerably in its enumeration algorithm, making problems involving 25 or so targets a manageable problem size.

Suppose one group of targets are as figured below:



The list of logical aimpoints identifies nine regions:

A	AB	BCD
B	BC	
C	BD	
D	CD	

From this list four sublists are made, one for each target. The sublists are filled from left to right and contain those entries of the main list that have included the specified target and have not already been assigned to another sublist. In this case the sublists look like this:

<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>
AB	BCD	CD	D
A	BD	C	
	BC		
	B		

Note that the aimpoints are listed by rank, so that those which involve the greatest number of targets come first.

The lists are searched from left to right, beginning with the first entry of the first list, AB. The object is to obtain a weapon set that includes all targets exactly once.

Because B is in AB, the "B" sublist is skipped, and the first entry in the "C" sublist is found, CD. At this point it is noted that a complete weapon set has been found of rank two, AB CD.

The fact that it is known that a solution exists of rank two on the first pass is what makes the MOST algorithm so fast. From this point on, when any two aimpoints fail to include all targets, MOST knows to terminate the enumeration containing the two and start with another.

AB CD is saved, and the process steps back to the last chosen entry from the sublist (it is useless to go on to AB C because once the set is complete, any further examination of the last set from which an aimpoint was picked would lead to a solution of greater rank than the one just found). Therefore, the code skips over the "B" sublist (not having chosen an aimpoint from there) and tries A from the "A" sublist.

Now an entry from the "B" sublist can be examined, since the first chosen aimpoint no longer contains B, and leads to the second two-weapon solution, A BCD. This solution is saved along with the previous solution, and the process continues.

Yet now, returning to the "A" sublist, MOST discovers that there are no further entries, and the list has been exhausted. Since there are no earlier lists to try, the process terminates with two equivalent solutions of rank two, AB CD and A BCD.

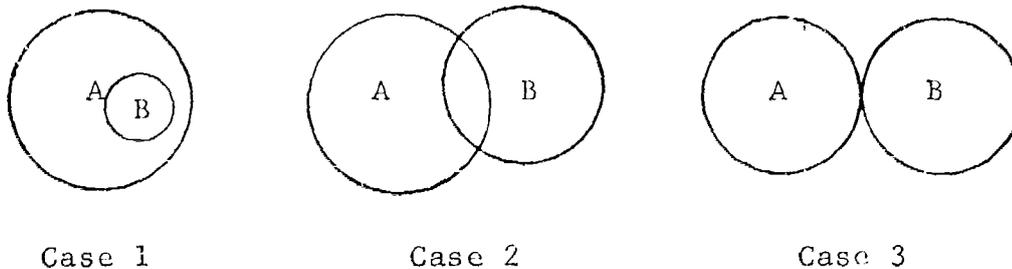
Notice that only two solutions were enumerated altogether, and both were acceptable. There was no need to generate others because the code rapidly converged on the required number of weapons, and did not need to consider enumerations that could not possibly produce two-weapon solutions. Of course, it should be repeated that this problem does not change the

fundamental, geometric character of the enumeration -- but it does allow MOST to handle worst-case problems of up to 30 targets in a practical fashion (the worst case, for the algorithm, is when the rank of the final solution is approximately half of the number of targets). And the MOST algorithm cannot overlook a solution.

## 2.6 TIE-BREAKING AND THE REFERENCE AIMPOINT

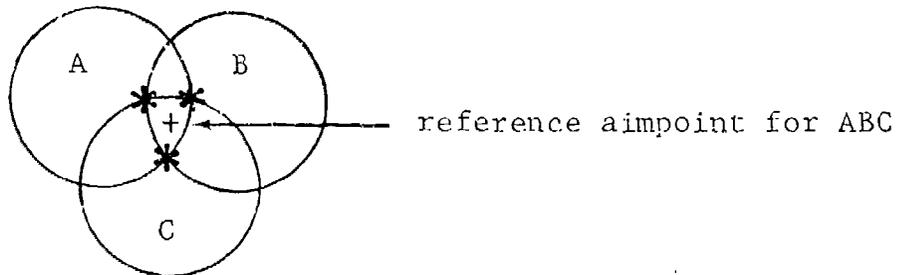
If the enumeration produces equivalent sets of minimum rank, the code picks between them on the basis of which alternatives are most likely to produce a higher value-weighted average PD on the target complex. (It will be recalled that the average PD, or APD, is the compounded PD on each element times the respective value summed together and divided by the total value of the elements).

To simplify this process, reference aimpoints are used to calculate APD. The "reference aimpoint" for an aimpoint of rank one is the center of the target generating the LAIR. For an aimpoint of rank two, there are three cases, as figured below:



In Case 1, the center of the totally-enclosed LAIR is used; in Case 2, the average of the coordinates of the intersection points of the two circles; in case 3, the tangent point (Case 3 is a degenerate Case 2).

For N circle intersections, the reference aimpoint is the average of all two-circle intersection points that lie within the intersection. Thus, for a standard three-circle intersection, as shown below, the reference aimpoint is the average of the three intersection points in the area:



Going back to the case above, where A BCD and AB CD were both solutions, the RP's would be used as physical aimpoints for each solution, probabilities of damage calculated, and the pair chosen which achieved the highest APD.

If more than a single independent group were examined, after all individual choices were made the sets would be merged into the complete answer.

This tie-breaking produces an answer likely to be best because the reference aimpoints, or RPs, are often quite good. They are not always best, however, so certain suboptimal choices can be made. A more elaborate but extremely time-consuming procedure of optimizing each candidate could be employed here, but has not for three reasons. The first is the large amount of computer time needed for optimization, the second is that any improvement to be gained in APD is likely to be quite small, and the third is that the wrong choice is seldom made.

## 2.7 OPTIMIZING AIMPOINTS

Optimization occurs after the desired weapon set has been chosen. Optimization begins with the set of reference aimpoints associated with the weapon set.

### 2.7.1 Local Optimization

Within logical aimpoint regions, physical aimpoints are optimized to increase APD over the complex. The reference aimpoint is the starting point for optimization. After a distance of movement is chosen, the code will find the direction to move the aimpoint to cause the greatest increase in APD.

The distance of movement is 90% of the distance between the current "best" aimpoint and the closest edge of the logical aimpoint region. Using this distance ensures that any aimpoint movement will stay in the region.

The direction of movement is determined by vectoral addition, where the length of each component vector is proportional to the increase in APD to be expected from a movement toward a particular target. The distance D is determined between the current "best" aimpoint and a target that is part of the complex. Distance d is the expected aimpoint movement, as defined above. The differential R is calculated as follows:

$$R = \frac{V(P(D-d) - P(D))}{d}$$

where P is the PD function and V is the value of the particular target. Note that, because  $P(D-d) \geq P(D)$ , the differential is always non-negative. The vector points toward the target and has length R.

Compounding is taken into account by function P. Thus all other aimpoints form the basis of the local optimization step.

Figure 4 shows a four-target complex in which the intersection of rank three is optimized first. The movement distance, in this figure, will be 90% of the distance between the RP and

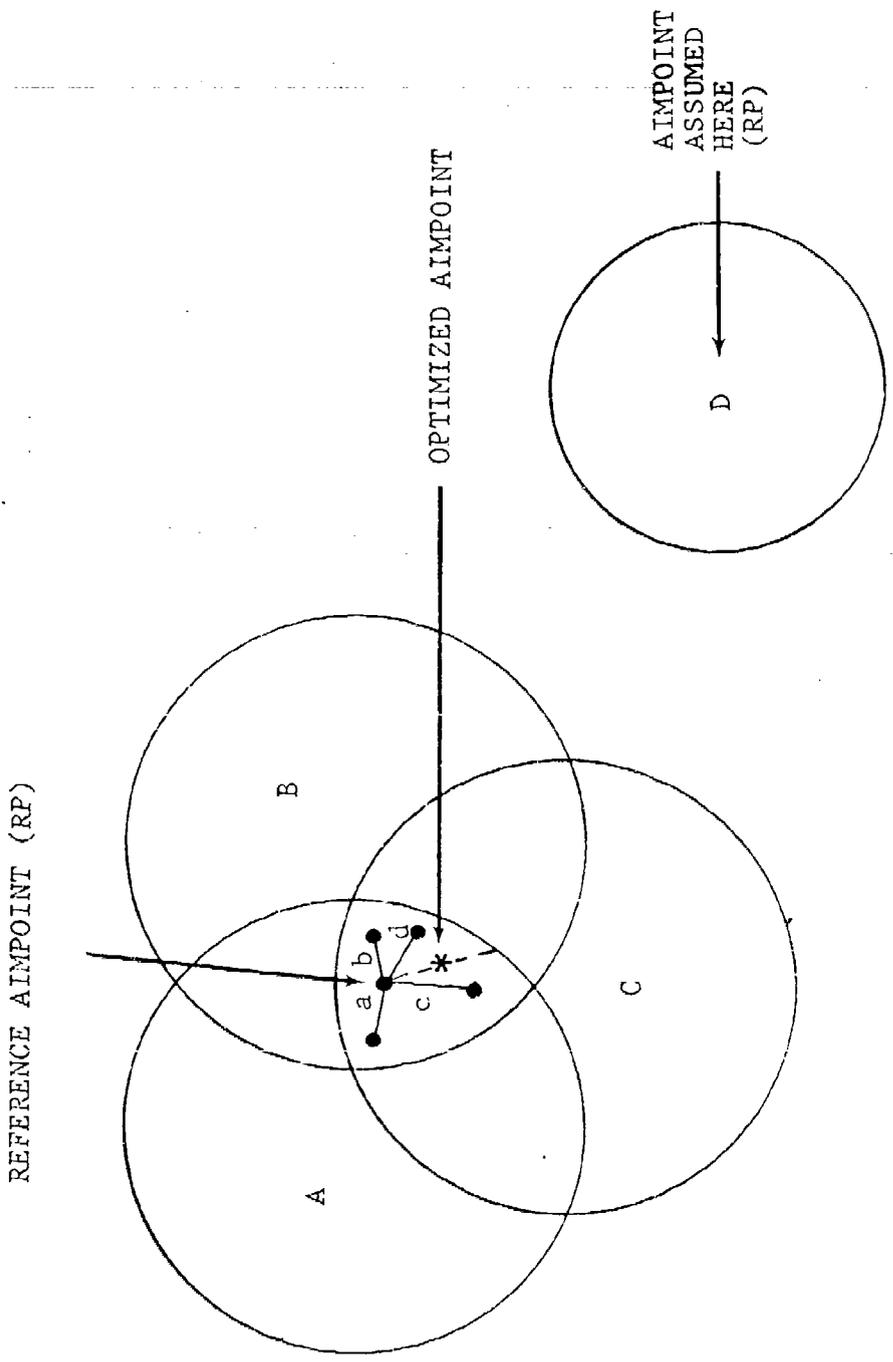


Figure 4. Four i ation complex; local optimization in AEC.

the nearest point of C. The direction of movement is determined by addition of vectors a, b, c, d, which respectively point towards targets A, B, C, and D. The new RP is denoted by a star.

The APD is calculated at the new RP and compared with the APD at the original RP. If no improvement occurred, or only a slight improvement, the process is terminated. If a large enough improvement occurred, the new physical aimpoint becomes the starting place for a repetition of the optimization algorithm. If the APD actually drops, it is assumed that the distance moved was too large. In this case the distance is halved and the process repeated until the answer improves, the movement distance becomes very small, or an insignificant increase in PD is detected.

#### 2.7.2 Global Optimization

Note that although in the example in Fig. 4 the aimpoint for a three-circle intersection was optimized the optimization included the value of a movement toward the detached target, and each PD calculation assumed an actual aimpoint at the RP of the detached target. In this way no local movement can ever decrease the total APD from the set of aimpoints previously (or subsequently) chosen. Therefore, each local step is guaranteed not to decrease the integrity of the global solution.

Also, the aimpoint of rank three was optimized before the isolated aimpoint because, in general, the higher the rank of the intersection the more restricted the range of movement for optimization. Thus, the effectiveness of the lower-ranking aimpoints is increased, in general, because their larger range makes up for their smaller general significance.

The argument against this ordering says that, because the higher-ranking intersections contribute more to APD than the lower-ranking, the lower-ranking should be optimized first. If

processed in reverse order, the effectiveness of the least significant aimpoints would be magnified without significant loss to the more significant aimpoints.

The two hypotheses were extensively tested, and the former was adopted, as it clearly took into account the dominant factors.

The optimizer can be executed again and again on a set of aimpoints until no additional movement of any aimpoint occurs. As each movement must be an improvement to the total solution, no oscillation can occur. Practical testing showed that for the current application, no such repetition is necessary.

## 2.8 MEETING THE APD REQUIREMENT

If, after selection and optimization, the APD requirement has been met, then the basic requirements of a solution have been satisfied. However, if the APD after optimization is less than required, additional weapons will be needed to bring up the APD.

The main list of all logical aimpoints is examined for the reference or optimized aimpoint which, if added to the set, most increases APD. This aimpoint may be a duplication of an aimpoint already in the set, or, more likely, an aimpoint not already used. If the APD resulting from the addition of this new weapon at the RP of the new aimpoint does not meet the requirement, the whole set of aimpoints is re-optimized as described in Section 2.7. If it still does not meet the requirement, the process of adding new weapons is repeated (with possible re-optimizations) until the requirement is finally satisfied.

## 2.9 VARIABLE PD REQUIREMENTS

Targets may be separated into two categories by the user, "critical" and "support". The functional difference is that critical targets may carry a higher PD requirement than support targets. The "critical" category is further divided into the subcategories of "reducible" and "irreducible". To the "irreducible" category are assigned those targets for which a high, simple PD is a strict requirement. Into the "reducible" category go those targets in which a high PD is desirable, but not at the expense of having to add additional weapons to the attack in order to achieve it.

Initially the aimpoints are selected as explained in the above sections. Subsequently, the solution is examined to see if there are potential gains from reducing the PD requirements of the reducible targets. The initial solution must involve more than two weapons, calculated APD must be significantly higher than what is required, and no additional weapons can have been added to the solution to bring the APD up to the requirement.

If these conditions are met, and there are critical reducible targets in the set, their PD requirement is lowered to that of support elements. Then MOST is repeated and the results compared.

A description of the behavior of MOST under these circumstances is in order. As PD requirements are dropped, the optimizer generally is able to achieve higher APD values, until one or more weapons drop out of the solution. Then there is a dramatic drop in APD. PD requirements can steadily be lowered until a single-weapon solution is found. However, what is sought in MOST is the solution that involves the fewest weapons while meeting the APD requirement on the elements of the complex, a minimum PD on all targets (the support target PD), a higher PD

on certain important targets, and as high a PD as possible on some not quite so important targets.

The algorithm drops the reducible PDs to the minimum to begin with, because the results at the least possible PDs are more important than at other PDs. If, at the lower PDs, the solution requires the same number of aimpoints as the initial solution, then the initial solution is best and we are done. If the new solution involves one or more fewer weapons than the initial solution, and still makes the APD requirement, it is better, replaces the initial, and we are done. If we have dropped one and the APD requirement has not been met, then there is no better solution than the initial. Only in the event that we have dropped more than one weapon from the initial solution and the APD results are unsatisfactory, is it necessary to try PD requirements between the initial critical and the support PDs.

In order to cope with this last eventuality, the interval between the critical PD and the support PD requirement is divided into steps beginning with the support PD. Each PD is one-fourth of the magnitude of the interval greater than the previous PD. Thus, if the support PD is 0.5 and the critical PD is 0.9, then intermediate values of 0.6, 0.7, and 0.8 are tried. For PDs of 0.4 and 0.9, the intermediates are 0.53, 0.65, and 0.78. There are always three intermediates.

The process repeats until an adequate solution is found, or until it is seen that a single weapon has been dropped from the critical solution and the APD requirement has not been met.

The new solutions can be made to be unsatisfactory if they produce a compounded PD value on any reducible target of less than a specified value. This value, called a "floor" value, can be set no lower than the support element requirement or higher than the critical requirement. If no floor value is

needed, it is set to the support requirement. Thus, it is possible to specify, for a group of targets, a minimum, simple PD, a desirable, higher PD, and a minimum compounded PD requirement.

### 3. CONCLUSIONS

MOST, in its current form, is best suited to find the minimum number of DGZ's needed by Poseidon to achieve complex damage requirements on an elementized target. However, the methodology behind MOST is much richer than this narrow application might indicate. It will work as well for aircraft-carried devices as for Poseidon.

There are many potential adaptations of MOST. For example, even though a full solution may involve  $N$  weapons, the user may not have  $N$  weapons at his disposal. The code could be modified to tell him, for any  $M \leq N$ , the best aimpoints for  $M$  weapons. Collateral damage consideration could easily be included, and complex target damage requirements.

The methodology is also known for mixing weapon types, when the rules for mixing are clearly stated. For example, if it is desirable to use two yields, and the desired limit on the number of higher-yield devices is known, the methodology can easily be extended to produce a good attack plan.

The problem of finding the fewest possible aimpoints for a number of targets by exhaustive enumeration grows geometrically in complexity with the number of targets involved, and there exists the possibility that no arithmetical  $\gamma$ -increasing solution will ever be found. However, research should continue to be directed toward the search for such an algorithm (or a proof that such an algorithm cannot exist) so that large numbers of targets, such as may be found in urban areas, might be targeted by MOST. There are, of course, ways of dividing problems into smaller ones, but at the expense of some degree of confidence in the results.

MOST represents an extension of standard methodologies to select aimpoints. The techniques used produce demonstrably better answers than can be produced through other techniques generally available to the potential user, and can be easily modified to provide results for related problems.

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE

Command & Control Technical Center  
Department of Defense

ATTN: C-343  
ATTN: C-315  
ATTN: C-332  
ATTN: C-312

U.S. European Command  
ATTN: ECJ2-1

Defense Nuclear Agency

ATTN: DDST  
ATTN: VLWS  
4 cy ATTN: TITL

Defense Technical Information Center  
12 cy ATTN: DD

Joint Chiefs of Staff  
ATTN: SAGA

Joint Strat. Tgt. Planning Staff

ATTN: JLAS  
2 cy ATTN: JLTW-2  
ATTN: JL  
ATTN: JP  
ATTN: JPS  
ATTN: JLTW

DEPARTMENT OF THE AIR FORCE

Strategic Air Command  
Department of the Air Force  
ATTN: XPFS

U.S. Air Forces in Europe  
ATTN: INT

DEPARTMENT OF DEFENSE CONTRACTORS

Science Applications, Inc.

ATTN: M. Drake  
ATTN: Document Control  
ATTN: L. Nessler  
ATTN: C. Whittenbury/W. Yengst  
ATTN: J. Warner